Discrete Mathematics and Theoretical Computer Science DMTCS vol. **14:1**, 2012, 69–86

Adaptive Identification of Sets of Vertices in Graphs

Ville Junnila[†]

Department of Mathematics, University of Turku, FI-20012 Turku, Finland

received 25th November 2008, *revised 4th April 2012*, *accepted 7th April 2012*.

In this paper, we consider a concept of adaptive identification of vertices and sets of vertices in different graphs, which was recently introduced by Ben-Haim, Gravier, Lobstein and Moncel (2008). The motivation for adaptive identification comes from applications such as sensor networks and fault detection in multiprocessor systems.

We present an optimal adaptive algorithm for identifying vertices in cycles. We also give efficient adaptive algorithms for identifying sets of vertices in different graphs such as cycles, king lattices and square lattices. Adaptive identification is also considered in Hamming spaces, which is one of the most widely studied graphs in the field of identifying codes.

Keywords: Identifying codes, Adaptive identification, Fault-detection, Sensor networks

1 Introduction

Let $G = (V, E)$ be a simple connected undirected graph with V as the set of vertices and E as the set of edges. Assume that the set V of vertices of G is finite. Then the set E of edges of G is also finite. A non-empty subset of V is called a *code*, and its elements are called *codewords*. The *distance* $d(x, y)$ is the number of edges in any shortest path between the vertices x and y. Let r be a positive integer. We say that x r*-covers* y if $d(x, y) \leq r$. Define then the r*-ball* $B_r(x)$ of radius r centered at $x \in V$ by

$$B_r(x) = \{y \in V \mid d(x, y) \leq r\}.$$

If all the r-balls in G have the same cardinality, then the cardinality of an r-ball is denoted by $V_r(G)$. For $X \subseteq V$, we denote

$$B_r(X) = \bigcup_{x \in X} B_r(x).$$

Let $C \subseteq V$ be a code and X be subset of V. An I*-set* of the set X with respect to the code C is

$$I_r(C; X) = I_r(X) = B_r(X) \cap C.$$

[†]Email: `viljun@utu.fi`

A code C is called an r-*covering* (or an r-*covering code*), if the set $I_r(C, x)$ is non-empty for all $x \in V$. In other words, each vertex in G is r-covered by at least one codeword. The minimum cardinality of an r-covering of G is denoted by $\gamma_r(G)$. A code C is called an r-*packing*, if the number of vertices in $I_r(C, x)$ is at most one for all $x \in V$. In other words, the r-balls centered at the vertices of C are all pairwise disjoint. The maximum cardinality of an r-packing of G is denoted by $c_r(G)$. If a code C is both an r-covering and an r-packing of G, then C is called an r-*perfect code*.

Definition 1.1 *Let r and ℓ be positive integers. A code $C \subseteq V$ is said to be $(r, \leq \ell)$-*identifying *in G if for all $X, Y \subseteq V$ such that $|X| \leq \ell$, $|Y| \leq \ell$ and $X \neq Y$ we have*

$$I_r(C; X) \neq I_r(C; Y).$$

*If $\ell = 1$, then we simply say that C is r-*identifying.

Remark 1.2 *Let r and ℓ be positive integers. Then there exists an $(r, \leq \ell)$-identifying code $C \subseteq V$ if and only if for all $X, Y \subseteq V$ such that $|X| \leq \ell$, $|Y| \leq \ell$ and $X \neq Y$ we have*

$$B_r(X) \neq B_r(Y).$$

*If there exists an $(r, \leq \ell)$-identifying code for a graph $G = (V, E)$, then G is said to be $(r, \leq \ell)$-*identifiable.

The original motivation for identification comes from fault detection in multiprocessor systems [15]. A multiprocessor system can be modeled as a graph, where vertices are seen as processor and edges as links between processors. A set of processors $C \subseteq V$ corresponding to an identifying code is chosen. Then each processor in C sends an alarm signal, if there exists a faulty processor in its neighborhood. Now the set of alarming processors corresponds to an I-set $I(X)$ of the identifying code, where X is the set of faulty processors. Hence, the set of faulty processors X can be located since each I-set is unique. Identifying codes can also be applied, for example, to sensor networks, where they are used in design of location and detection systems [17]. The most studied underlying graphs for identification are, e.g., square and king lattices, Hamming spaces and cycles [16].

Assume that a given graph G may contain *faulty vertices* and that we can ask whether there is a faulty vertex (or faulty vertices) in $B_r(x)$ for all $x \in V$. The *query* $Q_r : V \longrightarrow \{0, 1\}$ is equal to 1 for $x \in V$, if there is a faulty vertex in $B_r(x)$, else $Q_r(x)$ is equal to 0. We also say that a vertex $y \in V$ is r-*covered* by a query $Q_r(x)$ $(x \in V)$, if y belongs to the r-ball $B_r(x)$. Now the problem is to locate the faulty vertices using the queries $Q_r(x)$. The definition of identifying codes guarantees that if $C \subseteq V$ is an $(r, \leq \ell)$-identifying code in G, then by asking simultaneously all the queries $Q_r(c)$ for $c \in C$ we can locate in one step all the faulty vertices in G (assuming that there are at most ℓ faulty vertices in G).

The definition of identifying codes is based on the fact that all the queries have to be asked simultaneously. However, *adaptive identification*, which has been recently introduced in [2], is based on the idea that the queries can be asked one after the other, i.e. that a new query may depend on the answers given by the previous queries. In what follows, we call the identifying codes in Definition 1.1 as *regular* to distinguish them from the adaptive ones.

Let ℓ be the maximum number of faulty vertices in a graph G. The minimum cardinality of an $(r, \leq \ell)$-identifying code in G is then denoted by $i_{(r, \leq \ell)}(G)$. In adaptive identification, the corresponding value is the minimum number of queries required in the worst case to identify the (at most ℓ) faulty vertices and it is denoted by $a_{(r,\leq \ell)}(G)$. We also say that an algorithm (or a series of queries) $\mathcal{A}$ is *adaptive* $(r, \leq \ell)$*-identifying*, if it can identify the at most ℓ faulty vertices in G using only the queries $Q_r(x)$ $(x \in V)$.

In Ben-Haim *et al.* [1], [2] and [3], adaptive $(r, \leq 1)$-identification is considered in torii of square and king lattices. They suggest that further study would be needed in these torii when $\ell > 1$. This motivated the study in Sections 2.2 and 2.4. Adaptive $(r, \leq \ell)$-identification in cycles and Hamming spaces are studied in Sections 2.1 and 2.3, respectively.

In adaptive $(r, \leq \ell)$-identification (and in regular $(r, \leq \ell)$-identification), it is common that we encounter problems with large ℓ. Namely, the considered graph is no longer $(r, \leq \ell)$-identifiable with large enough ℓ. For example, square and king lattices are not $(r, \leq \ell)$-identifiable when $r > 1$ and $\ell \geq 3$. Therefore, we introduce a slightly modified version of adaptive $(r, \leq \ell)$-identification in Section 3 to enable the handling of larger ℓ, for example, in king lattices.

2 Adaptive identification

The following theorem is a generalized version of Theorem 1 in [1, 2].

Theorem 2.1 *Let r and ℓ be positive integers. Assume G is an $(r, \leq \ell)$-identifiable graph such that each r-ball in G has the same cardinality. Then we have*

$$a_{(r,\leq \ell)}(G) \geq c_r(G) - 1 + \left\lceil \log_2 \left(\sum_{i=0}^{\ell} \binom{K}{i} \right) \right\rceil ,$$

where $K = |G| - (c_r(G) - 1)V_r(G)$.

Proof: Let $G = (V, E)$ be an $(r, \leq \ell)$-identifiable graph such that each r-ball in G has the same cardinality. Consider then an algorithm $\mathcal{A}$ that is adaptive $(r, \leq \ell)$-identifying. It is clearly possible that the values given by the first $c_r(G) - 1$ queries of $\mathcal{A}$ are all equal to 0, i.e. that there are no faulty vertices in the r-balls of the first $c_r(G) - 1$ queries.

After the first $c_r(G) - 1$ queries there are still at least $|G| - (c_r(G) - 1)V_r(G)$ vertices that are not r-covered by any of the previously asked queries. Furthermore, we know that among these uncovered vertices there exist from 0 to ℓ faulty vertices. Therefore, the number of different possibilities for these faulty vertices to be among these uncovered vertices is $\sum_{i=0}^{\ell} \binom{K}{i}$, where $K = |G| - (c_r(G) - 1)V_r(G)$. Hence, we need at least $\lceil \log_2(\sum_{i=0}^{\ell} \binom{K}{i}) \rceil$ queries to locate these faulty vertices or to conclude that there are none. Thus, the following lower bound follows:

$$a_{(r,\leq \ell)}(G) \geq c_r(G) - 1 + \left\lceil \log_2 \left(\sum_{i=0}^{\ell} \binom{K}{i} \right) \right\rceil .$$

□

Notice that $|G| - (c_r(G) - 1)V_r(G) \geq V_r(G)$ since the value $c_r(G)$ is the maximum number of vertices in an r-packing of G. Thus, the slightly weaker lower bound of Theorem 1 in [1, 2] immediately follows from the previous result:

$$a_{(r,\leq 1)}(G) \geq c_r(G) - 1 + \lceil \log_2(V_r(G) + 1) \rceil .$$

2.1 Adaptive identification in cycles

Let G be a cycle of length n, i.e. a cycle with n vertices. Regular identification in cycles have been studied, for example, in [5], [10] and [18]. The following theorem provides the accurate value for the number of queries $a_{(r,\leq 1)}(G)$ needed in the adaptive $(r, \leq 1)$-identification of G, when $n > 2r+1$. Notice also that if $n \leq 2r+1$, then the cycle G is not $(r, \leq 1)$-identifiable.

Theorem 2.2 *Let G be a cycle of length n. If $n = 2r+1+k$ with $1 \leq k \leq 2r$, then we have*

$$a_{(r,\leq 1)}(G) = \left\lfloor \frac{2r+1}{k} \right\rfloor + \left\lceil \log_2 \left(2r+1-k\left(\left\lfloor \frac{2r+1}{k} \right\rfloor - 1 \right) \right) \right\rceil. \tag{1}$$

If $n \geq 2(2r+1)$, then we have

$$a_{(r,\leq 1)}(G) = \left\lfloor \frac{n}{2r+1} \right\rfloor - 1 + \left\lceil \log_2 \left(n - \left(\left\lfloor \frac{n}{2r+1} \right\rfloor - 1 \right)(2r+1) + 1 \right) \right\rceil. \tag{2}$$

Proof: Let $G = (V, E)$ be a cycle of length n and $V = \{x_1, x_2, \ldots, x_n\}$. Consider first the result (1) for the cycles of length $n = 2r+1+k$ with $1 \leq k \leq 2r$. Notice that now each query outputting 1 tell us that one of the vertices covered by the query is faulty and also that none of the k uncovered vertices is faulty.

Let us begin by showing a lower bound on $a_{(r,\leq 1)}(G)$ when $n = 2r+1+k$ with $1 \leq k \leq 2r$. We can assume that the first query asked outputs value 1. Therefore, the faulty vertex is one of the $2r+1$ vertices covered by the first query. We can assume that the next $\lfloor (2r+1)/k \rfloor - 1$ queries also output value 1. By the considerations in the first paragraph above, there still exists at least $2r+1-k(\lfloor (2r+1)/k \rfloor - 1)$ vertices such that exactly one of them is faulty. Hence, we still need at least $\lceil \log_2(2r+1-k(\lfloor (2r+1)/k \rfloor - 1)) \rceil$ queries to locate the faulty vertex. Thus, at least $\lfloor (2r+1)/k \rfloor + \lceil \log_2(2r+1-k(\lfloor (2r+1)/k \rfloor - 1)) \rceil$ queries is needed to identify the faulty vertex.

Now the following algorithm is adaptive $(r, \leq 1)$-identifying:

1. Begin by asking the query $Q_r(x_{r+1})$. If $Q_r(x_{r+1}) = 0$, then there does not exist any faulty vertex in $B_r(x_{r+1})$. Hence, the possible faulty vertex belongs to the remaining k vertices, which are consecutive ones in a cycle. Therefore, since $k \leq 2(2r+1)+1$, we obtain using dichotomic (or binary) search (as in the step 3) that at most $\lceil \log_2(k+1) \rceil$ queries are needed to locate the faulty vertex among the remaining k vertices or to conclude there is none. It can be easily verified that in this case we use at most $\lfloor (2r+1)/k \rfloor + \lceil \log_2(2r+1-k(\lfloor (2r+1)/k \rfloor - 1)) \rceil$ queries.

2. Assume that $Q_r(x_{r+1}) = 1$. Now for $i = 1, 2, \ldots, \lfloor (2r+1)/k \rfloor - 1$ ask the query $Q_r(x_{r+1+ik})$. If the query $Q_r(x_{r+1+jk}) = 0$ for some j, then the faulty vertex belongs to the set $\{x_{(j-1)k+1}, x_{(j-1)k+2}, \ldots, x_{jk}\}$, which consists of k consecutive vertices. Now the faulty vertex can be located in this set as in the step 3. Again it is easy to verify that also in this case we need at most $\lfloor (2r+1)/k \rfloor + \lceil \log_2(2r+1-k(\lfloor (2r+1)/k \rfloor - 1)) \rceil$ queries.

3. Assume now that all the $\lfloor (2r+1)/k \rfloor$ queries asked in the previous steps outputted the value 1. Then the faulty vertex is one of the remaining $2r+1-k(\lfloor (2r+1)/k \rfloor - 1)$ vertices in the set

$$\{x_{k(\lfloor (2r+1)/k \rfloor - 1)+1}, x_{k(\lfloor (2r+1)/k \rfloor - 1)+2}, \ldots, x_{2r+1}\}.$$

Therefore, the faulty vertex can now be located by dichotomic search using at most $\lceil \log_2(2r+1-k(\lfloor(2r+1)/k\rfloor-1)\rceil$ queries. By the considerations in the first paragraph, the dichotomic search can now, indeed, be used since $2r+1-k(\lfloor(2r+1)/k\rfloor-1)<2k$ and there are $k\lfloor(2r+1)/k\rfloor$ consecutive vertices that are known to be not faulty (the set of uncovered vertices can now always be divided into two roughly equal halves).

Thus, we need at most $\lfloor(2r+1)/k\rfloor + \lceil \log_2(2r+1-k(\lfloor(2r+1)/k\rfloor-1))\rceil$ queries to locate the faulty vertex or to conclude there is none. In conclusion, the first result (1) of the claim holds.

Consider then the result (2) for the cycles with length $n \geq 2(2r+1)$. Denote first $n = q_1(2r+1)+q_0$, where $0 \leq q_0 < 2r+1$. Then consider the upper bound on $a_{(r,\leq 1)}(G)$. The following algorithm is adaptive $(r, \leq 1)$-identifying:

1. For $i = 1, \ldots, q_1 - 1$ ask the query $Q_r(x_{i(2r+1)-r})$. If $Q_r(x_{j(2r+1)-r}) = 1$ for some $1 \leq j \leq q_1 - 1$, then there exists a faulty vertex in $B_r(x_{j(2r+1)-r})$. The vertices in the ball are clearly consecutive and, therefore, we can locate the faulty vertex by dichotomic (or binary) search as in step 2. Thus, the number of queries needed in this case is bounded above by the value given in the equation (2).

2. Assume then that all the vertices r-covered by the queries in step 1 are not faulty. Now there are still $2r+1+q_0$ vertices that are not r-covered by any of the queries from step 1. Since the number of uncovered vertices is $2r+1+q_0 < 2(2r+1)$, the possible faulty vertex can be located by dichotomic search. Hence, we need at most $\lceil \log_2(2r+1+q_0+1)\rceil$ queries in this step.

Thus, we have $a_{(r,\leq 1)}(G) \leq q_1 - 1 + \lceil \log_2(2r+1+q_0+1)\rceil$.

By Theorem 2.1, we have $a_{(r,\leq 1)}(G) \geq q_1 - 1 + \lceil \log_2(2r+1+q_0+1)\rceil$. Hence, the claim (2) immediately follows from the upper and lower bounds. □

Consider then adaptive $(r, \leq 2)$-identification in a cycle G of length n. By Theorem 2.1, we have

$$a_{(r,\leq 2)}(G) \geq \left\lfloor \frac{n}{2r+1} \right\rfloor - 1 + \left\lceil \log_2 \left(\sum_{i=0}^{2} \binom{K}{i} \right) \right\rceil, \tag{3}$$

where $K = n - (\lfloor n/(2r+1)\rfloor - 1)(2r+1)$. The following theorem provides an upper bound on $a_{(r,\leq 2)}(G)$.

Theorem 2.3 *Let G be a cycle of length n. If $n \geq 3(2r+1)$, then we have*

$$a_{(r,\leq 2)}(G) \leq \left\lfloor \frac{n}{2r+1} \right\rfloor + 1 + 2\lceil \log_2(2r+1)\rceil. \tag{4}$$

Proof: Let $G = (V, E)$ be a cycle of length n and $V = \{x_1, x_2, \ldots, x_n\}$. Denote also $n = q_1(2r+1)+q_0$, where $0 \leq q_0 < 2r+1$. The following algorithm is adaptive $(r, \leq 2)$-identifying:

1. For $i = 1, \ldots, q_1$, we first ask the queries $Q_r(x_{i(2r+1)-r})$. If two of these queries give value 1, then we proceed as in step 3. Assuming this is not the case, we need to ask one additional (carefully chosen) query. If the first query $Q_r(x_{r+1}) = 1$, then the additional query is $Q_r(x_{n-r})$, else it is $Q_r(x_{(q_1+1)(2r+1)-r})$, where the index of x is subtracted by n if it is larger than n. Notice that each vertex in G has now been r-covered by a query and that if two queries output value 1, then there exist two faulty vertices in G (one in each query giving value 1).

2. If all the queries in step 1 output value 0, then there clearly exist no faulty vertices in G.

3. If two of the queries in step 1 give value 1, then there are exactly two faulty vertices in G. Namely, one faulty vertex in each r-ball corresponding to the queries outputting 1. Denote these r-balls containing a faulty vertex by B_1 and B_2. Note that the vertices in B_1 and B_2 are consecutive ones. Hence, the faulty vertex in B_1 (or B_2) can be located by dichotomic search and, in particular, using queries that do not r-cover vertices from B_2 (or B_1). (Here we use the assumption that $n \geq 3(2r+1)$.) Therefore, the faulty vertices can be located by dichotomic search using at most $2\lceil \log_2(2r+1) \rceil$ queries.

4. If there is exactly one query giving value 1, then we know that this query is r-covering 1 or 2 faulty vertices. Denote the considered r-ball by $B = \{y_1, y_2, \ldots, y_{2r+1}\}$ and assume that the two vertices adjacent to y_1 are y_0 and y_2. The faulty vertices in B can then be located as follows:

   (*i*) First the r-ball B is divided into two halves $B_1 = \{y_1, \ldots, y_{\lfloor (2r+1)/2 \rfloor}\}$ and $B_2 = \{y_{\lfloor (2r+1)/2 \rfloor + 1}, \ldots, y_{2r+1}\}$. Then we find out whether there is a faulty vertex in B_1 or B_2 using the queries $Q_r(y_0)$ and $Q_r(y_{2r+1})$.

   (*ii*) If the halves B_1 and B_2 both contain a faulty vertex, then the faulty vertices from B_1 and B_2 can be separately located by dichotomic search. Otherwise, we know that only either B_1 or B_2 contain faulty vertices. For simplicity, assume that B_1 contains faulty vertices. Then we proceed as in step (*i*), but we replace the set B by B_1.

   Thus, in step 4 we locate the faulty vertices using at most $2\lceil \log_2(2r+1) \rceil$ queries.

In conclusion, the algorithm locates the faulty vertices using at most $q_1 + 1 + 2\lceil \log_2(2r+1) \rceil$ queries. Thus, the claim follows. □

Notice that the difference between the lower bound (3) and the upper bound (4) is at most 4 queries for any r and n. Indeed, this can be concluded by estimating the term with logarithm in the lower bound when the term with logarithm in the upper bound gives a fixed value. Hence, we conclude that the upper and lower bounds differ only by a constant (for any radius r).

Let $G = (V, E)$ be a cycle of length n with $V = \{x_1, x_2, \ldots, x_n\}$. Since $B_r(\{x_1, x_3\}) = B_r(\{x_1, x_2, x_3\})$ for any (positive) r, the cycle G is not $(r, \leq \ell)$-identifiable when $\ell \geq 3$.

2.2 Adaptive identification in torii of king lattice

Let p and q be positive integers. The graph $T^k_{p,q} = (V, E)$ is a $p \times q$ *torus in the king lattice*, if the vertex set is
$$V = \{(i, j) \mid 0 \leq i \leq p-1, 0 \leq j \leq q-1\}$$
and the edge set is
$$\begin{aligned} E = \{&((i,j),(i,j+1)), ((i,j),(i+1,j)) \mid 0 \leq i \leq p-1, 0 \leq j \leq q-1\} \\ \cup \{&((i,j),(i+1,j+1)), ((i,j),(i+1,j-1)) \mid 0 \leq i \leq p-1, 0 \leq j \leq q-1\}, \end{aligned}$$
where the first coordinate is calculated modulo p and the second coordinate is calculated modulo q (see Figure 1).

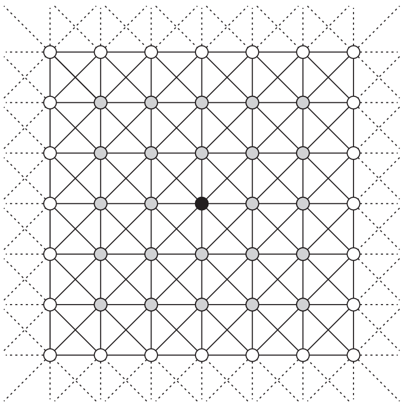

Fig. 1: A 2-ball centered at the black vertex is illustrated in the torus $T^k_{7,7}$. Notice that the vertices in the border are wrapped around as suggested by the definition of the edge set E.

Consider then an r-ball in $T^k_{p,q}$. The r-ball $B_r((x,y))$, which is shortened as $B_r(x,y)$, can be seen as a $(2r+1)\times(2r+1)$ square in the king lattice since

$$B_r(x,y) = \{(i,j) \mid |x-i| \leq r, |y-j| \leq r\}.$$

Now it is easy to see that there exists an r-perfect code $C \subseteq T^k_{p,q}$ if and only if both p and q are dividable by $2r+1$.

Regular $(r, \leq \ell)$-identification in king lattice (or grid) have been studied, for example, in [6] and [11]. Consider then adaptive $(r, \leq \ell)$-identification in $T^k_{p,q}$. If $\ell \geq 3$, then it can be easily seen that $T^k_{p,q}$ is not $(r, \leq \ell)$-identifiable for any r. The case with $\ell = 1$ is considered in [1, 3]. Thus, we concentrate here on the case with $\ell = 2$. By Theorem 2.1, we have

$$a_{(r,\leq 2)}(T^k_{p,q}) \geq \frac{pq}{(2r+1)^2} - 1 + \left\lceil \log_2 \left(\sum_{i=0}^{2} \binom{(2r+1)^2}{i} \right) \right\rceil, \tag{5}$$

since $c_r(T^k_{p,q}) = pq/(2r+1)^2$. The following theorem provides an upper bound for $a_{(r,\leq 2)}(T^k_{p,q})$.

Theorem 2.4 *Let p and q be positive integers dividable by $2r+1$. If $p \geq 3(2r+1)$ and $q \geq 3(2r+1)$, then we have*

$$a_{(r,\leq 2)}(T^k_{p,q}) \leq \frac{pq}{(2r+1)^2} + 4\left\lceil \log_2(2r+1) \right\rceil. \tag{6}$$

Proof: Assume p and q are positive integers dividable by $2r+1$. Let $T^k_{p,q}$ be a torus in the king lattice with pq vertices. Now there exists an r-perfect covering C of $T^k_{p,q}$. The following algorithm is adaptive $(r, \leq 2)$-identifying:

1. For every $c \in C$ ask the query $Q_r(c)$.

2. If all the queries output value 0, then there clearly exist no faulty vertices in $T^k_{p,q}$.

3. If two queries output value 1, then there are two queries each r-covering exactly one faulty vertex. The faulty vertex r-covered by a query can now be found as follows: we first locate by dichotomic search the column containing a faulty vertex and then from this column we identify the faulty vertex again using dichotomic search. (Notice that suitable queries used in dichotomic searches can be found even if the r-balls of the two queries are next to each other. Here we actually use the assumption that $p \geq 3(2r+1)$ and $q \geq 3(2r+1)$.) In conclusion, we need at most $4\lceil \log_2(2r+1) \rceil$ queries in this step.

4. If there is exactly one query outputting value 1, then this query is r-covering 1 or 2 faulty vertices. Using similar ideas as in the proof of Theorem 2.3, we can locate the columns (or a column) containing faulty vertices using at most $2\lceil \log_2(2r+1) \rceil$ queries (consider columns in the king lattice as the vertices in cycles) and then from these columns (or a column) identify the faulty vertices using at most $2\lceil \log_2(2r+1) \rceil$ queries. Thus, we need at most $4\lceil \log_2(2r+1) \rceil$ queries in this step.

In conclusion, the previous algorithm identifies the faulty vertices using at most $pq/(2r+1)^2+4\lceil \log_2(2r+1) \rceil$ queries. □

Using similar arguments as in the case of cycles, we conclude that the difference between the lower bound (5) and the upper bound (6) is at most 5 queries for any r.

2.3 Adaptive identification in Hamming spaces

In this subsection, we consider adaptive $(1, \leq \ell)$-identification in binary Hamming spaces. Let n be a positive integer. The binary *Hamming space* $\mathbb{F}^n$ is the n-fold Cartesian product of the binary field $\mathbb{F} = \{0, 1\}$. The *Hamming distance* $d(\mathbf{x}, \mathbf{y})$ between words $\mathbf{x}, \mathbf{y} \in \mathbb{F}^n$ is the number of coordinate places in which they differ. The following simple lemma, which is needed in the sequel, can be easily proven.

Lemma 2.5 *Let* $\mathbf{x}, \mathbf{y} \in \mathbb{F}^n$. *Then*

$$|B_1(\mathbf{x}) \cap B_1(\mathbf{y})| = \begin{cases} n+1 & \textit{if } \mathbf{x} = \mathbf{y}, \\ 2 & \textit{if } 1 \leq d(\mathbf{x}, \mathbf{y}) \leq 2, \\ 0 & \textit{otherwise.} \end{cases}$$

We begin by considering adaptive $(1, \leq 1)$-identification in Hamming spaces. The following lemma is needed in the proof of Theorem 2.7, which provides a lower bound for $a_{(1,\leq 1)}(\mathbb{F}^n)$.

Lemma 2.6 *Let X be a non-empty subset of $\mathbb{F}^n$. Assume that there is 0 or 1 faulty words in X. Then an adaptive $(1, \leq 1)$-identifying algorithm needs at least*

$$\left\lceil \sqrt{\frac{|X|}{2}} \right\rceil$$

queries, which are centered at a word in $\mathbb{F}^n$, to locate the faulty word in X or to conclude that there is none.

Proof: Let X be a non-empty subset of $\mathbb{F}^n$ and let $\mathcal{A}$ be an algorithm that identifies the faulty word in X using queries from $\mathbb{F}^n$. Define then k as the maximum number of words in X that are 1-covered by a

1-ball of $\mathbb{F}^n$, i.e.

$$k = \max_{\mathbf{x} \in \mathbb{F}^n} |B_1(\mathbf{x}) \cap X|.$$

Now we have two approaches for the lower bound on the number of queries used in $\mathcal{A}$:

(1) By the previous definition, a query $Q_1(\mathbf{x})$ with $\mathbf{x} \in \mathbb{F}^n$ can 1-cover at most k words of X. Assume that the first $\lfloor |X|/k \rfloor - 1$ queries of $\mathcal{A}$ output value 0. Now there still exist at least k words that are not 1-covered by any of the previous queries, and we need at least $\lceil \log_2(k+1) \rceil$ queries to locate the faulty word among these uncovered words or to conclude that there is none. Thus, the number of queries used in $\mathcal{A}$ is at least $\lfloor |X|/k \rfloor - 1 + \lceil \log_2(k+1) \rceil$. If $k = 1$, then the claim clearly follows. Otherwise, we need at least $|X|/k$ queries in the algorithm $\mathcal{A}$.

(2) On the other hand, we know by Lemma 2.5 that the number of words in the intersection of two different 1-balls of $\mathbb{F}^n$ is at most 2. Let then $\mathbf{x} \in \mathbb{F}^n$ be a word such that the number of words in $B_1(\mathbf{x}) \cap X$ is equal to k. Assume that there exists a faulty word in $B_1(\mathbf{x}) \cap X$. Using similar arguments as in the first lower bound from (1), we obtain that the number of queries used in $\mathcal{A}$ is at least

$$\begin{cases} \lfloor k/2 \rfloor - 1 + \lceil \log_2 2 \rceil & \text{if } 2 \mid k, \\ \lfloor k/2 \rfloor - 1 + \lceil \log_2 3 \rceil & \text{if } 2 \nmid k. \end{cases}$$

Therefore, the number of queries needed is at least $k/2$.

By the considerations above, the number of queries needed in $\mathcal{A}$ is at least $\max\{|X|/k, k/2\}$. Therefore, by straightforward analysis, it can be concluded that (with any choice of k) the number of queries needed is at least

$$\left\lceil \sqrt{\frac{|X|}{2}} \right\rceil.$$

□

The following theorem provides a lower bound for $a_{(1,\leq 1)}(\mathbb{F}^n)$.

Theorem 2.7 *We have*

$$a_{(1,\leq 1)}(\mathbb{F}^n) \geq c_1(\mathbb{F}^n) + \left\lfloor \frac{n+1}{8} \right\rfloor.$$

Proof: Let algorithm $\mathcal{A}$ be an adaptive $(1, \leq 1)$-identifying code (in $\mathbb{F}^n$). (Notice that the size of a ball of radius 1 in $\mathbb{F}^n$ is equal to $n+1$.) Assume then that the first $c_1(\mathbb{F}^n) - \lceil (n+1)/8 \rceil$ queries of $\mathcal{A}$ output value 0. (By simple analysis, it can be shown that the number of queries $c_1(\mathbb{F}^n) - \lceil (n+1)/8 \rceil$ is chosen in such a way that it gives the best possible lower bound using this approach.) Then the number of words that are not 1-covered by the previous queries is at least $\lceil (n+1)/8 \rceil (n+1)$. Therefore, by Lemma 2.6, the number of queries used in $\mathcal{A}$ is at least

$$\begin{aligned} & c_1(\mathbb{F}^n) - \left\lceil \frac{n+1}{8} \right\rceil + \left\lceil \sqrt{\frac{\lceil (n+1)/8 \rceil (n+1)}{2}} \right\rceil \\ & \geq c_1(\mathbb{F}^n) + \left\lceil \frac{n+1}{4} \right\rceil - \left\lceil \frac{n+1}{8} \right\rceil \geq c_1(\mathbb{F}^n) + \left\lfloor \frac{n+1}{8} \right\rfloor, \end{aligned}$$

where the second inequality is obtained, for example, by considering two cases depending on whether $n+1$ is dividable by 4 or not. □

The following theorem provides an upper bound for $a_{(1,\leq 1)}(\mathbb{F}^n)$.

Theorem 2.8 *We have*

$$a_{(1,\leq 1)}(\mathbb{F}^n) \leq \gamma_1(\mathbb{F}^n) + \left\lceil \frac{n+1}{2} \right\rceil .$$

Proof: Let $C = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{|C|}\}$ be a 1-covering code of $\mathbb{F}^n$ attaining $\gamma_1(\mathbb{F}^n)$. Denote then by $\mathbf{e}_i$ the word in $\mathbb{F}^n$ that has value 1 in the ith coordinate place and value 0 in all other places. Now the following algorithm is adaptive $(1, \leq 1)$-identifying:

1. For $i = 1, \ldots, |C|-1$ ask the query $Q_1(\mathbf{x}_i)$. If $Q_1(\mathbf{x}_i) = 1$ for any $i = 1, \ldots, |C|-1$, then the faulty word in $B_1(\mathbf{x}_i)$ can be located as in the following step 2.
2. Assume then that all the previous queries output value 0, meaning that none of these queries do not 1-cover any faulty word. Now we can assume without loss of generality that $\mathbf{x}_{|C|} = \mathbf{0}$. For $i = 1, \ldots, \lceil n/2 \rceil - 1$ ask the query $Q_1(\mathbf{e}_{2i-1} + \mathbf{e}_{2i})$. If now for any i we have $Q_1(\mathbf{e}_{2i-1} + \mathbf{e}_{2i}) = 1$, then the faulty word can be located using one more suitably chosen query. Hence, assume that none of the previous queries 1-cover any faulty word. Now it can be easily seen that we only need two more queries to locate the faulty word in the remaining words or to conclude that there are none.

In conclusion, the previous algorithm uses at most $\gamma_1(\mathbb{F}^n) + \lceil n/2 \rceil + 1 = \gamma_1(\mathbb{F}^n) + \lceil (n+1)/2 \rceil$ queries. □

Let then $\mathbb{F}^n$ be a Hamming space with integers $n = 2^s - 1$ and $s \geq 3$. By [7], we know that there now exists a 1-perfect covering of $\mathbb{F}^n$. Hence, we have $c_1(\mathbb{F}^n) = \gamma_1(\mathbb{F}^n) = 2^n/(n+1)$. Therefore, for the previous lengths, Theorems 2.7 and 2.8 can be written as follows:

$$c_1(\mathbb{F}^n) + \frac{n+1}{8} \leq a_{(1,\leq 1)}(\mathbb{F}^n) \leq c_1(\mathbb{F}^n) + \frac{n+1}{2}.$$

As above, assume that $n = 2^s - 1$ and $s \geq 3$. Consider then adaptive $(1, \leq \ell)$-identification in $\mathbb{F}^n$ with $\ell > 1$. Theorem 2.7 naturally provides a lower bound also for $a_{(1,\leq \ell)}(\mathbb{F}^n)$. The following theorem gives us then an upper bound on $a_{(1,\leq \ell)}(\mathbb{F}^n)$.

Theorem 2.9 *Let n and s be integers such that $n = 2^s - 1$ and $s \geq 3$. If now $\ell < n/6 + 1$, then we have*

$$a_{(1,\leq \ell)} \leq c_1(\mathbb{F}^n) + \ell \cdot \frac{n+3}{2}.$$

Proof: Let C be a 1-perfect code of $\mathbb{F}^n$ with $n = 2^s - 1$ and $s \geq 3$. The number of words in C is equal to $\gamma_1(\mathbb{F}^n) = c_1(\mathbb{F}^n)$. Denote then $n = 2m+1$, where $m = 2^{s-1} - 1$. Using $c_1(\mathbb{F}^n)$ queries, we can now locate the 1-balls centered at the words of C which contain faulty words. Assume then that the 1-balls $B_1(\mathbf{x}_1), B_1(\mathbf{x}_2), \ldots, B_1(\mathbf{x}_k)$ with $1 \leq k \leq \ell$ are the ones containing faulty words. Denote the set consisting of these 1-balls by S.

Consider then locating the faulty words inside $B_1(\mathbf{x}_1)$. We can assume without loss of generality that $\mathbf{x}_1 = \mathbf{0}$. We would now like to 1-cover $B_1(\mathbf{0})$ using m words of weight two and 1 word of weight one (similarly to the proof of Theorem 2.8). However, here we cannot use such words of weight one and two which 1-cover words from other 1-balls containing faulty words. The choice of m words of weight two and 1 word of weight one that 1-cover $B_1(\mathbf{0})$ is called a *partition* of $B_1(\mathbf{0})$. Notice that if $\mathbf{x}$ and $\mathbf{y}$ are two different words in a partition of $B_1(\mathbf{0})$, then the intersection of the sets $B_1(\mathbf{x}) \cap B_1(\mathbf{0})$ and $B_1(\mathbf{y}) \cap B_1(\mathbf{0})$ is empty. A partition is called *unavailable*, if some of the words in the partition 1-cover words in other 1-balls containing faulty words. In what follows, we show that all partitions are not unavailable.

The number of all different partitions is equal to

$$\frac{\binom{n}{2,2,\ldots,2,1}}{m!} = \frac{n!}{m! \cdot 2^m}, \tag{7}$$

where the numerator is the usual multinomial coefficient and the number of 2's in it is equal to m. Notice that all the 1-balls of S that make partitions unavailable are centered at words of weights three or four, since words that are of weight at most two are not included in the 1-perfect code C and words of weight at least five cannot clearly make partitions unavailable. Hence, we consider the number of partitions that are made unavailable by a 1-ball of S centered at a word of weight three or four.

Each 1-ball centered at a word of weight three that contains faulty words produces at most

$$\frac{\binom{3}{2}\binom{n-2}{2,2,\ldots,2,1}}{(m-1)!} + \frac{\binom{3}{1}\binom{n-1}{2,2,\ldots,2}}{m!} = \frac{3(n-2)!}{(m-1)! \cdot 2^{m-1}} + \frac{3(n-1)!}{m! \cdot 2^m} = \frac{6(n-1)!}{m! \cdot 2^m} \tag{8}$$

unavailable partitions. Each 1-ball centered at a word of weight four that contains faulty words produces at most

$$\binom{4}{2}\frac{\binom{n-2}{2,2,\ldots,2,1}}{(m-1)!} = \frac{6(n-2)!}{(m-1)! \cdot 2^{m-1}} = \frac{6(n-1)!}{m! \cdot 2^m} \tag{9}$$

unavailable partitions. Hence, by the equations (7), (8) and (9), the number of available partitions is at least

$$\frac{n!}{m! \cdot 2^m} - (k-1) \cdot \frac{6(n-1)!}{m! \cdot 2^m}.$$

Since $k \le \ell < n/6 + 1$, the number of available partitions is positive. Thus, there exists a partition that is not unavailable. In conclusion, we need at most $m + 1 = (n+1)/2$ queries to locate which of the m pairs of two words of weight one contain faulty words and whether there exist faulty words in the pair containing the last word of weight 1 and the word $\mathbf{0}$.

Assume now that a pair $\{\mathbf{e}_{i_1}, \mathbf{e}_{i_2}\}$ contains faulty words, where $i_1, i_2 \in \{1, 2, \ldots, n\}$ ($\mathbf{e}_i$ defined as in the proof of Theorem 2.8). In order to conclude whether $\mathbf{e}_{i_1}$ is a faulty word, we need a word $\mathbf{e}_{i_1} + \mathbf{e}_j$ of weight two such that $B_1(\mathbf{e}_j) \setminus \{\mathbf{e}_{i_1}\}$ contains no faulty vertices, i.e. $\mathbf{e}_j$ is not included in any of the pairs containing faulty words in $B_1(\mathbf{x}_1)$ and $B_1(\mathbf{e}_j)$ does not intersect with any of the 1-balls of S other than $B_1(\mathbf{x}_1)$. Now we have that the number of such words of weight two is at least

$$n - 2(\ell - k) - \binom{4}{2}(k-1).$$

Since $\ell < n/6 + 1$, the previous number of words is positive. Hence, there exists a word of weight two satisfying the previous conditions. Assume then that the pair $\{\mathbf{0}, \mathbf{e}_i\}$ $(i \in \{1, 2, \ldots, n\})$ contain faulty words. Using similar counting arguments as before, we can show that the faulty words can be found in this pair by at most two queries. In conclusion, we need at most two queries to locate the faulty words inside each pair containing them.

If $k = \ell$, then, by the previous considerations and by the fact that now we need only one query to identify the faulty word from a pair $\{\mathbf{e}_{i_1}, \mathbf{e}_{i_2}\}$, we need at most

$$c_1(\mathbb{F}^n) + \ell \cdot \frac{n+1}{2} + \ell = c_1(\mathbb{F}^n) + \ell \cdot \frac{n+3}{2}$$

queries to locate the faulty words in $\mathbb{F}^n$. If $1 \leq k \leq \ell - 1$, then we also need at most

$$c_1(\mathbb{F}^n) + (\ell - 1) \cdot \frac{n+1}{2} + 2\ell \leq c_1(\mathbb{F}^n) + \ell \cdot \frac{n+3}{2}$$

queries. Thus, the claim follows. □

Remark 2.10 *It should be noted that the upper bound of the previous theorem can be sharpened to* $c_1(\mathbb{F}^n) + \ell \cdot (n+1)/2$. *This slight improvement is obtained by more detailed considerations in the last paragraph of the proof (as pointed out by an anonymous referee).*

2.4 Adaptive identification in square lattices

Let p and q be positive integers. The graph $T_{p,q} = (V, E)$ is a $p \times q$ *torus in the square lattice*, if the vertex set is

$$V = \{(i, j) \mid 0 \leq i \leq p-1, 0 \leq j \leq q-1\}$$

and the edge set is

$$E = \{((i, j), (i, j+1)), ((i, j), (i+1, j)) \mid 0 \leq i \leq p-1, 0 \leq j \leq q-1\},$$

where the first coordinate is calculated modulo p and the second coordinate is calculated modulo q (see Figure 2).

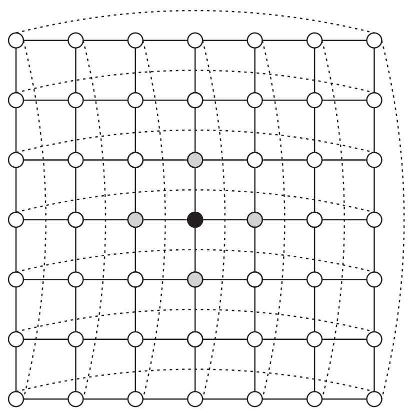

Fig. 2: A 1-ball centered at the black vertex is illustrated in the torus $T_{7,7}$.

Regular $(r, \leq \ell)$-identification in square lattice (or grid) have been studied, for example, in [4], [13] and [14]. Consider then adaptive $(1, \leq \ell)$-identification of $T_{p,q}$. The case with $\ell = 1$ is considered in [1, 2]. If, on the other hand, $\ell \geq 4$, then it is easy to see that $T_{p,q}$ is not $(1, \leq \ell)$-identifiable. Hence, we concentrate on this subsection to the cases with $\ell = 2$ and $\ell = 3$.

The size of the 1-ball in $T_{p,q}$ is clearly 5, i.e. $V_1(T_{p,q}) = 5$ (see Figure 2). By [8] and [9], we know that there exists a 1-perfect code of $T_{p,q}$, if both p and q are dividable by 5.

Consider then adaptive $(1, \leq 2)$-identification in $T_{p,q}$. By Theorem 2.1, we have the lower bound

$$a_{(1,\leq 2)}(T_{p,q}) \geq \frac{pq}{5} + 3. \tag{10}$$

The following theorem provides an upper bound for $a_{(1,\leq 2)}(T_{p,q})$.

Theorem 2.11 *Let p and q be positive integers dividable by 5. If $p \geq 10$ and $q \geq 10$, then we have*

$$a_{(1,\leq 2)}(T_{p,q}) \leq \frac{pq}{5} + 6. \tag{11}$$

Proof: The proof is similar to and even easier than the proof of Theorem 2.12. Therefore, the proof is omitted here. □

Consider then adaptive $(1, \leq 3)$-identification in $T_{p,q}$. Again by Theorem 2.1, we have the lower bound

$$a_{(1,\leq 3)}(T_{p,q}) \geq \frac{pq}{5} + 4. \tag{12}$$

The following theorem provides an upper bound for $a_{(1,\leq 3)}(T_{p,q})$.

Theorem 2.12 *Let p and q be positive integers dividable by 5. If $p \geq 15$ and $q \geq 15$, then we have*

$$a_{(1,\leq 3)}(T_{p,q}) \leq \frac{pq}{5} + 9. \tag{13}$$

Proof: Assume p and q are positive integers dividable by 5. Let $T_{p,q}$ be a torus in the square lattice with pq vertices. Now there exists a 1-perfect covering C of $T_{p,q}$. In what follows, we present a sketch of an adaptive $(1, \leq 3)$-identifying algorithm:

1. For every $c \in C$ ask the query $Q_1(c)$.
2. If all the queries output value 0, then there clearly exist no faulty vertices in $T_{p,q}$.
3. If exactly one query output value 1, then this query 1-covers 1, 2 or 3 faulty vertices. Assume that this query is centered at the vertex (x, y). Then ask the queries $Q_1(x, y+2)$, $Q_1(x+2, y)$, $Q_1(x, y-2)$ and $Q_1(x-2, y)$ (the first coordinate is calculated modulo p and the second is calculated modulo q). Depending on the answers of these queries, we need at most one auxiliary query to locate all the (from 1 to 3) faulty vertices in $B_1(x, y)$. Thus, we need at most 5 queries in this step.
4. Assume then that there exist two queries outputting 1. Assume first that the 1-balls of these queries are next to each other, i.e. assume that the queries outputting 1 are $Q_1(x, y)$ and $Q_1(x + 2, y + 1)$ (other cases are symmetrical). Now we need to locate the 1 or 2 faulty vertices in $B_1(x, y)$ without

using queries that 1-cover vertices in $B_1(x+2, y+1)$. We start by asking the query $Q_1(x, y-1)$. If $Q_1(x, y-1) = 0$, then the faulty vertices can be found using queries $Q_1(x-1, y-1)$, $Q_1(x, y+2)$ and $Q_1(x+1, y-1)$ as illustrated in Figure 3(a) (totally at most 4 queries).

Assume then $Q_1(x, y-1) = 1$. Ask the query $Q_1(x-1, y+1)$. If $Q_1(x-1, y+1) = 1$, then the faulty vertices can be easily found using totally at most 4 queries. Assume then $Q_1(x-1, y+1) = 0$. Ask the query $Q_1(x-1, y)$ (see Figure 3(b)). If $Q_1(x-1, y) = 1$, then the vertex (x, y) is faulty and we proceed by asking the query $Q_1(x+1, y-1)$. If $Q_1(x+1, y-1) = 0$, then the vertex (x, y) is the only faulty one in $B_1(x, y)$ (totally 4 queries used). Otherwise, the second faulty vertex in $B_1(x, y)$ can be found by asking the query $Q_1(x-1, y-1)$ (totally 5 queries used).

We have still not considered the somewhat more problematic case when $Q_1(x-1, y) = 0$. Now we know that the vertex $(x, y-1)$ is faulty and that the vertices (x, y), $(x-1, y)$ and $(x, y+1)$ are not faulty (see Figure 3(c)). However, there exists no fourth query telling whether the vertex $(x+1, y)$ is faulty, since queries 1-covering vertices in $B_1(x+2, y+1)$ cannot be used. Therefore, we proceed next by locating the faulty vertices in the 1-ball $B_1(x+2, y+1)$ using similar queries as with $B_1(x, y)$. (Indeed, the faulty vertices can be located from $B_1(x+2, y+1)$ using at most 5 queries, since we know that the vertices (x, y) and $(x, y+1)$ are not faulty.) After locating the faulty vertices in $B_1(x+2, y+1)$, we can also conclude whether the vertex $(x+1, y)$ is faulty or not. Hence, we use in this case at most 9 queries to locate all the faulty vertices in $T_{p,q}$.

In other cases, we also proceed by finding the faulty vertices (or a faulty vertex) in $B_1(x+2, y+1)$ using similar queries as with $B_1(x, y)$. Notice that if there is only one faulty vertex in a 1-ball, then it can be located using 4 queries, and if there are two faulty vertices, then they can be located using 5 queries. Hence, in all the cases we need at most 9 queries to locate all the faulty vertices in $T_{p,q}$.

It should also be noted that if the two queries outputting 1 after step 1 are not next to each other we can still use similar techniques to locate the faulty vertices using at most 9 queries.

5. Assume then that three queries in step 1 output value 1. Now each of these queries 1-cover exactly one faulty vertex. There are again several cases, but here we consider only one case as an example (others are analogous).

   Assume that the queries outputting 1 are $Q_1(x, y)$, $Q_1(x-1, y+2)$ and $Q_1(x+2, y+1)$. Ask then the query $Q_1(x-1, y-1)$ (see Figure 3(d)). If $Q_1(x-1, y-1) = 1$, the faulty vertex can be found by the query $Q_1(x-2, y)$. Otherwise the faulty vertex can be located by asking the query $Q_1(x+1, y-1)$ and depending on the answer we might need one auxiliary query. In conclusion, we need at most 3 queries to locate the faulty vertex in $B_1(x, y)$. Notice that these queries are chosen in such a way that they do not intersect with the balls $B_1(x-1, y+2)$ and $B_1(x+2, y+1)$. Therefore, we need at most 9 queries to locate the faulty vertices in $T_{p,q}$.

   With any other choice of the queries outputting 1 in step 1, we can always found the faulty vertices using at most 9 queries.

In conclusion, the previous algorithm shows that

$$a_{(1,\leq 3)}(T_{p,q}) \leq \frac{pq}{5} + 9.$$

□

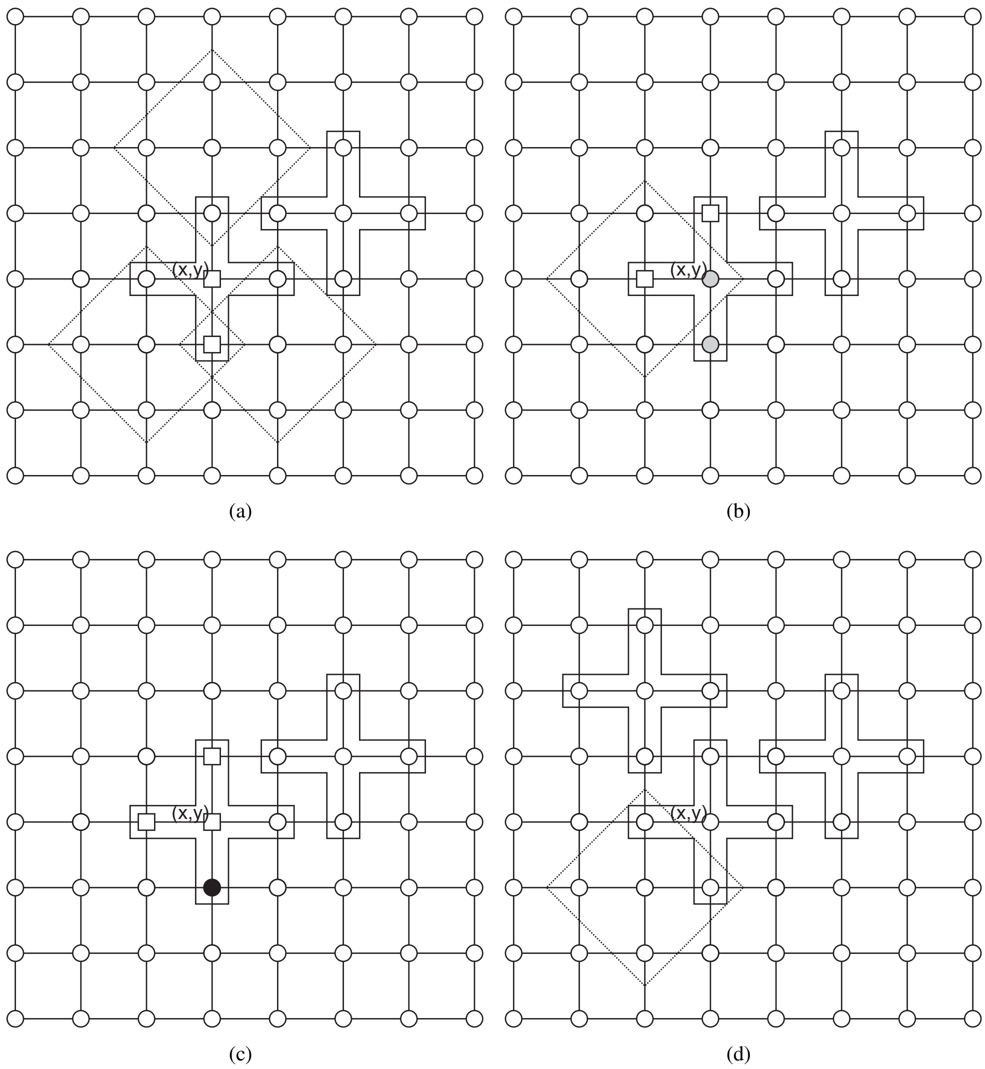

Fig. 3: The proof of Theorem 2.12 illustrated. The black and squared vertices respectively represent faulty and non-faulty vertices. Moreover, at least one of the grey vertices is faulty. Furthermore, the dashed squares represent the asked queries.

3 Adaptive weak identification

In this section, we introduce a concept of adaptive weak $(r, \leq \ell)$-identification, which enables the handling of larger ℓ. Similar concept has also been considered in the case of regular $(r, \leq \ell)$-identification. These weakly $(r, \leq \ell)$-identifying codes are considered, for example, in [12].

Assume that during the execution of an algorithm we can fix a faulty vertex after one is found and then proceed with the algorithm. We call such an algorithm as *adaptive weakly* $(r, \leq \ell)$*-identifying*, if it can fix all the (at most ℓ) faulty vertices. The minimum of the maximum number of queries needed in an adaptive weakly $(r, \leq \ell)$-identifying algorithm is denoted by $a^W_{(r,\leq \ell)}(G)$, where G is the underlying graph.

In what follows, we consider adaptive weak $(r, \leq \ell)$-identification in torii of king lattice $T^k_{p,q}$. The following theorem provides an upper bound for $a^W_{(r,\leq \ell)}(T^k_{p,q})$, when p and q are dividable by $2r+1$.

Theorem 3.1 *Let p and q be positive integers dividable by $2r+1$. If $\ell < \min\{p/(2r+1), q/(2r+1)\}$, then we have*

$$a^W_{(r,\leq \ell)}(T^k_{p,q}) \leq \frac{pq}{(2r+1)^2} + \ell\left(2\left\lceil \log_2(2r+1)\right\rceil + 1\right).$$

Proof: Assume p and q are positive integers dividable by $2r+1$. Let $T^k_{p,q}$ be a torus in the king lattice with pq vertices. Now there exists an r-perfect code C of $T^k_{p,q}$. The following algorithm is adaptive weakly $(r, \leq 2)$-identifying:

1. For every $c \in C$ ask the query $Q_r(c)$.

2. If all the queries output value 0, then there clearly exist no faulty vertices in $T^k_{p,q}$. Let then the queries outputting value 1 be centered at $v_1, v_2, \ldots, v_m$ with $1 \leq m \leq \ell$. In what follows, the first coordinate of a vertex is calculated modulo p and the second one is calculated modulo q. Choose then $v_j = (x_j, y_j)$ to be such that there do not exist faulty vertices in $B_r(x_j - 2r, y_j)$, $B_r(x_j-2r, y_j+2r)$ and $B_r(x_j, y_j+2r)$. Indeed, such a vertex always exists since $\ell < \min\{p/(2r+1), q/(2r+1)\}$. Now using some of the queries $Q_r(x_j-2r, y_j), Q_r(x_j-2r+1, y_j), \ldots, Q_r(x_j-1, y_j)$ we can locate a column containing a faulty vertex by dichotomic search. Assume that this column is formed by the vertices $\{(x_j+k, y_j-r), (x_j+k, y_j-r+1), \ldots, (x_j+k, y_j+r)\}$ with $-r \leq k \leq r$. The dichotomic search ensures that the found column is the leftmost of the columns containing faulty vertices in $B_r(v_j)$. Therefore, we can locate the faulty vertex in this column by dichotomic search using some of the queries $Q_r(x_j+k-r, y_j+2r), Q_r(x_j+k-r, y_j+2r-1), \ldots, Q_r(x_j+k-r, y_j+1)$. In conclusion, one faulty vertex in $B_r(v_j)$ can be identified using at most $2\lceil \log_2(2r+1)\rceil$ queries.

3. After locating the faulty vertex, we fix it and ask again the query $Q_r(v_j)$. Then we proceed as in the step 2.

In conclusion, the previous algorithm locates the faulty vertices using at most

$$\gamma_r(T^k_{p,q}) + \ell\left(2\left\lceil \log_2(2r+1)\right\rceil + 1\right)$$

queries. Thus, the claim follows. □

As in the previous theorem, let p and q be positive integers dividable by $2r+1$. Consider then adaptive weak $(r, \leq \ell)$-identification in $T^k_{p,q}$ with $\ell = p/(2r+1)$. Let then the following sets be two patterns of faulty vertices in $T^k_{p,q}$:

$$\begin{aligned} X_1 &= \{(i \cdot p/(2r+1), 0) \mid i = 0, \ldots, \ell - 1\}, \\ X_2 &= \{(1 + i \cdot p/(2r+1), 0) \mid i = 0, \ldots, \ell - 1\}. \end{aligned}$$

Clearly, if $Q_r(v) = 1$ ($v \in T^k_{p,q}$) when the faulty vertices of $T^k_{p,q}$ are X_1, then $Q_r(v) = 1$ when the faulty vertices of $T^k_{p,q}$ are X_2. The same also holds for the other direction. Hence, $Q_r(v) = 1$ ($v \in T^k_{p,q}$) with fault pattern X_1 if and only if $Q_r(v) = 1$ with fault pattern X_2. Thus, we cannot locate any faulty vertices in $T^k_{p,q}$. Similar considerations also apply when $\ell = q/(2r+1)$. In conclusion, there does not exist any adaptive weakly $(r, \leq \ell)$-identifying code, if $\ell \geq p/(2r+1)$ or $\ell \geq q/(2r+1)$. Therefore, the bound on ℓ is the best possible in the previous theorem.

Acknowledgements

I wish to thank an anonymous referee for the insightful comments and suggestions. In particular, the sharpened result of Remark 2.10 is based on the referee's observation.

References

[1] Y. Ben-Haim, S. Gravier, A. Lobstein, and J. Moncel. Adaptive identification in graphs. Technical report, Rapport interne Telecom Paris-2007D012, September 2007.

[2] Y. Ben-Haim, S. Gravier, A. Lobstein, and J. Moncel. Adaptive identification in graphs. *J. Comb. Theory Ser. A*, 115(7):1114–1126, 2008.

[3] Y. Ben-Haim, S. Gravier, A. Lobstein, and J. Moncel. Adaptive identification in tori in the king lattice. *Electron. J. Combin.*, 18(1):Paper 116, 13, 2011.

[4] Y. Ben-Haim and S. Litsyn. Exact minimum density of codes identifying vertices in the square grid. *SIAM J. Discrete Math.*, 19(1):69–82, 2005.

[5] N. Bertrand, I. Charon, O. Hudry, and A. Lobstein. Identifying and locating-dominating codes on chains and cycles. *European J. Combin.*, 25(7):969–987, 2004.

[6] I. Charon, I. Honkala, O. Hudry, and A. Lobstein. The minimum density of an identifying code in the king lattice. *Discrete Math.*, 276(1-3):95–109, 2004. 6th International Conference on Graph Theory.

[7] G. Cohen, I. Honkala, S. Litsyn, and A. Lobstein. *Covering codes*, volume 54 of *North-Holland Mathematical Library*. North-Holland Publishing Co., Amsterdam, 1997.

[8] S. W. Golomb and L. R. Welch. Algebraic coding and the Lee metric. In *Error Correcting Codes (Proc. Sympos. Math. Res. Center, Madison, Wis., 1968)*, pages 175–194. John Wiley, New York, 1968.

[9] S. W. Golomb and L. R. Welch. Perfect codes in the Lee metric and the packing of polyominoes. *SIAM J. Appl. Math.*, 18:302–317, 1970.

[10] S. Gravier, J. Moncel, and A. Semri. Identifying codes of cycles. *European J. Combin.*, 27(5):767–776, 2006.

[11] I. Honkala and T. Laihonen. Codes for identification in the king lattice. *Graphs Combin.*, 19(4):505–516, 2003.

[12] I. Honkala and T. Laihonen. On the identification of sets of points in the square lattice. *Discrete Comput. Geom.*, 29(1):139–152, 2003.

[13] I. Honkala and T. Laihonen. On identifying codes in the triangular and square grids. *SIAM J. Comput.*, 33(2):304–312, 2004.

[14] I. Honkala and A. Lobstein. On the density of identifying codes in the square lattice. *J. Combin. Theory Ser. B*, 85(2):297–306, 2002.

[15] M. G. Karpovsky, K. Chakrabarty, and L. B. Levitin. On a new class of codes for identifying vertices in graphs. *IEEE Trans. Inform. Theory*, 44(2):599–611, 1998.

[16] A. Lobstein. Watching systems, identifying, locating-dominating and discrminating codes in graphs, a bibliography. Published electronically at `http://perso.enst.fr/~lobstein/debutBIBidetlocdom.pdf`.

[17] S. Ray, D. Starobinski, A. Trachtenberg, and R. Ungrangsi. Robust location detection with sensor networks. *IEEE Journal on Selected Areas in Communications*, 22(6):1016–1025, August 2004.

[18] D. L. Roberts and F. S. Roberts. Locating sensors in paths and cycles: The case of 2-identifying codes. *European J. Combin.*, 29(1):72–82, 2008.