Theoretical Computer Science 461 (2012) 17–26

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Computing majority with triple queries☆

Gianluca De Marco[a,*], Evangelos Kranakis[b], Gábor Wiener[c]

[a] Dipartimento di Informatica, Università di Salerno, 84084 Fisciano (SA), Italy

[b] School of Computer Science, Carleton University, Ottawa, ON, K1S 5B6, Canada

[c] Department of Computer Science and Information Theory, Budapest University of Technology and Economics, H-1521, Budapest, Hungary

ARTICLE INFO

Keywords:
Search
Balls
Colors
Computation models
Queries
Pairing model
General model

ABSTRACT

Motivated from the well-known problem of establishing efficient diagnostic techniques for detecting faults in fault-tolerant computer systems we study a problem for computing majority with *restricted tests* in a set of items of two types (e.g., faulty and non-faulty). Stated in a more abstract form, consider a bin containing n balls colored with two colors. In a k-query, k balls are selected by a questioner and an oracle gives an answer which (depending on the computation model being considered) is related to the distribution of colors of the balls in this k-tuple. The oracle never reveals the colors of the individual balls. Following a number of queries and answers the questioner is said to determine *majority* if either (1) it can output a ball of the majority color provided that such a color exists, or (2) otherwise can determine that there is no majority color. We investigate the minimum number of queries required to determine majority in two computation models. We give algorithms to compute the minimum number of 3-queries which are needed so that the questioner can determine the majority color and provide tight and almost tight upper and lower bounds on the number of queries needed in each case. Our results indicate a surprising difference between the number of queries to determine majority with double versus triple queries.

1. Introduction

Suppose that $k \leq n$ are two non-negative integers. We are given a bin containing n balls colored with two colors, e.g., red and blue. At any time, we can choose any k of the balls and ask the question "do these balls have the same color?". The aim is to produce a ball of the *majority color* (meaning that the number of balls with that color is strictly greater than that of the other color), or to state that there is no majority (i.e. there is an identical number of red and blue balls) using as few questions as possible. We are interested in computing the number of queries for the worst case instance of the input to the problem, and our aim is to provide an algorithm that uses a minimum number of such queries on any input.

1.1. Model of computation

In computing the majority there are two participants: a *questioner* (denoted by **Q**) and an *oracle* (or *adversary*) denoted by **A** (see Fig. 1). The questioner asks questions on the status of the color of the balls by providing the oracle with k balls

☆ This is an extended and revised version of a paper that appeared in the proceedings of the 17th Annual International Computing and Combinatorics Conference (COCOON'11) held in Dallas, Texas, USA, August 14–16, 2011, Springer LNCS, Vol. 6842, pp. 604–615.

* Corresponding author.
E-mail addresses: demarco@dia.unisa.it (G. De Marco), kranakis@scs.carleton.ca (E. Kranakis), wiener@cs.bme.hu (G. Wiener).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.03.039

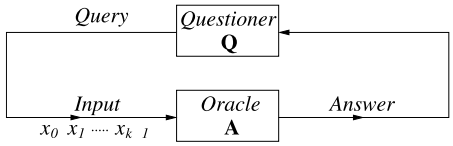

Fig. 1. A questioner (**Q**), an oracle (or adversary) (**A**) and a k-query input.

and the oracle provides a reply which, depending on the type of the oracle, is related to the distribution of colors of the balls concerned.

Although the answer of the oracle depends on the k-tuple provided by the questioner, it is otherwise *permutation-independent*, i.e., the answer is independent of the order of the balls. Moreover, the balls are equipped with distinct identities (say, integers $1, 2, \ldots, n$). Notice that in the course of querying the questioner it is never allowed to see the color of any ball but merely relies on the answers provided by the oracle.

Majority problem for k-tuple queries. Given n balls consisting of two colors, and an integer $k \leq n$.

1. Determine whether or not there is a majority color (i.e., more than half) among the balls, and
2. if indeed there is majority, to output a ball having the majority color.[1]

In addition, we would like to minimize the number of queries required.

Query models. Next we define two query models which will be considered in the sequel. In each model the input to the oracle is a set $\{x_0, x_1, \ldots, x_{k-1}\}$ of k balls.[2]

1. *Pairing Model:* The answer to a query is either YES or NO. YES means that all balls have the same color. NO means that not all the balls have the same color and to show this, (any) two balls of different color are also provided.[3]
2. *General Model:* The answer is either YES or NO. YES means that all balls have the same color, NO means that the balls in the k-tuple do not have the same color.

The majority problem could also be considered as a game between two players: the questioner (**Q**) and the adversary (**A**). The game consists of rounds in which **Q** chooses balls from the bin and **A** tells whether they have the same color or not. The goal is to determine whether or not there is majority in the minimum number of rounds. In the Pairing model introduced above if the answer of **A** is NO (that is, not all of the balls have the same color), then **A** has to show two balls of different color.

1.2. Notation

Let $q_k^p(n)$ and $q_k(n)$ denote the minimum number of queries needed to solve the majority problem for n balls colored with two colors using k-queries in the *Pairing* and *General* models, respectively. It is obvious that $q_k^p(n) \leq q_k(n)$ (assuming that these numbers exist).

1.3. Motivation and related work

Multiprocessor computer systems are vulnerable to component faults and system level diagnosis is an important technique for improving reliability and availability [15]. Motivation for studying the majority problem comes from the study of diagnostic techniques for detecting faults in fault-tolerant computer systems. Such systems are designed to be able to handle failures such as hardware-related, input/output device failures, software bugs and errors, just to mention a few. More importantly, the majority model being considered in this paper is an abstraction of employing limited (in terms of the number of components) diagnostic techniques whereby only a small fixed number of components (in our case k) can be tested at a time and it is also required to determine the number of such tests for a successful termination of the testing procedure. Moreover, the type of local test being performed indicates in a sense the capabilities of the testing procedure being employed. Thus, for example, in the *general* model the tester can only indicate whether or not all inputs are identical while in the *pairing* model and when not all inputs are identical it can also provide a pair of inputs which are different.

Our study is a natural generalization of the well-known *majority problem* previously studied in the literature [1] where at any stage *two* balls a and b are chosen and the question is "do a and b have the same color?". It is obvious that $q_2(n) = q_2^p(n)$ and Saks and Werman [16] showed that $q_2(n) = n - \nu(n)$, where $\nu(n)$ denotes the number of 1's in the binary representation

1 Note that for n odd there is always a majority color, however in this case it is also required to output a ball of the majority color.

2 Note that both query models are oblivious to the order of the elements of the k-tuple.

3 To be more precise, it would enough to return (any) two indices of balls indicating this fact; however, by abuse of language we will be talking about the balls with these indices since this terminology has become standard in the literature.

of n. Other proofs of their theorem appear in [7] and [17]. Alonso et al. [8] also gave the solution for the average case (see also [5]). Aigner [3,2] introduced several variants and generalizations of the majority problem. In particular, in the (n, k)-majority game one has to show a k-majority ball z (that is, there are at least k balls colored like z), or declare there is no k-majority.

Other variations of the majority problem include the case where more than two colors are available. Fisher and Salzberg [12] studied the majority problem when the number of colors is any integer up to n. In this case the majority color is the color such that there are at least $\lfloor n/2 \rfloor + 1$ balls of that color. They solved the problem by showing that $\lceil 3n/2 \rceil - 2$ comparisons are necessary and sufficient. Another natural generalization is the plurality problem where the goal is to identify a ball of the dominant color (i.e., the one occurring more often than any other one). In [4] linear upper and lower bounds were given for 3 colors. For any number of colors $c \geq 3$, Chung et al. [9] showed that $(c - 1 - 1/c)n - 2$ comparisons suffice, while the currently best lower bound is $2cn/27 - o(n)$ [14]. For c sufficiently large, the currently best bounds are in [6]. The authors of [9] studied also the oblivious versions both for the majority and the plurality problem. In oblivious (or non/adaptive) strategies the questioner has to ask all questions in advance before getting any answer from the oracle. Finally, bounds for randomized algorithms can be found in [10,11,14].

1.4. Outline and results of the paper

In this paper we study a generalization of the well-known *majority problem*. We deal with arbitrary k-queries (instead of double queries, i.e., 2-queries, that were studied in the original version of the majority problem), and prove some general results concerning them. We also determine the function $q_3^p(n)$ and give almost matching lower and upper bounds for the case $q_3(n)$. In outline, Section 2 discusses the problem of existence of a solution to the majority problem in the models proposed. In Section 3 we are considering the General model for $k = 3$ and give lower and upper bounds for $q_3(n)$ whose difference is either 1 or 2, depending on the residue of n modulo 4. The main result in Theorem 2 shows that for $n = 4m + r$ for some $r \in \{0, 1, 2, 3\}$ and $m \geq 1$,

$$
\begin{aligned}
&n - 1 \leq q_3(4m) \leq n;\\
&n - 2 \leq q_3(4m + 1) \leq n - 1;\\
&n - 1 \leq q_3(4m + 2) \leq n + 1;\\
&n - 2 \leq q_3(4m + 3) \leq n.
\end{aligned}
$$

Section 4 investigates the Pairing model. Here we give a general lower bound for $q_k^p(n)$ and compute the precise value of $q_3^p(n)$, namely

$$
q_3^p(n) = \begin{cases} n/2 + 1 & \text{if } n \text{ is even, and} \\ \lfloor n/2 \rfloor & \text{if } n \text{ is odd.} \end{cases}
$$

This result is quite surprising: complexity functions that are not *strictly* increasing are known, but a complexity function that is not even increasing is quite unusual in search theory.

2. Existence of solutions

Before trying to compute $q_k^p(n)$ and $q_k(n)$ we should discuss whether these numbers exist at all, since it may happen that asking all possible queries is not enough for **Q** to solve the problem. It is clear that n should be at least k and if $n = k$, then the only possible query is enough in both games for $k = 2$ and in the General model for $k = 3$, but not in the other cases. For $k \geq 3$ we prove that $q_k(n)$ and $q_k^p(n)$ exist if and only if $n \geq 2k - 2$ and $n \geq 2k - 3$, respectively.

Theorem 1. *Let $k \geq 3$.*

1. *The number $q_k(n)$ exists if and only if $n \geq 2k - 2$.*
2. *The number $q_k^p(n)$ exists if and only if $n \geq 2k - 3$.*

Proof. Consider k-tuple inputs. We have to show that **Q** can solve the majority problem by asking all possible queries if and only if $n \geq 2k - 2$ in the General model and if and only if $n \geq 2k - 3$ in the Pairing model. It is easy to see for both problems that if **A** gives a positive answer (i.e., declares that balls in a set S of size k have the same color), then **Q** can learn which balls have the same color as the balls in S, thus solving the problem. The questioner **Q** just has to ask every query containing the balls of S', where S' is an arbitrary subset of S having size $k - 1$.

Thus we may assume that **A** never gives a positive answer. Now the first part of the theorem is easy to prove. If $n \geq 2k - 1$, then **A** must give at least a positive answer, since there are k balls having the same color, while if $n = 2k - 2$ and there are no k balls of the same color, then there cannot be majority (both colors appear $k - 1$ times). On the other hand, if $n \leq 2k - 3$ then it is possible that there are exactly $k - 1$ red balls and $n - k + 1 \leq k - 2$ blue balls, in which case the answer is always negative and the problem cannot be solved, since there is a majority but no ball can be named as one in majority.

Since **Q** is given more information in the Pairing model, it is obvious that the problem can be solved if $n \geq 2k - 2$. To show that in the case $n = 2k - 3$ **Q** can also solve the problem, but in the cases $n \leq 2k - 4$ cannot, we use graphs to describe

the knowledge of **Q**. In this model to every negative answer of **A** there corresponds a pair of balls having different colors. Let these pairs be the edges of a graph G, whose vertices are the balls.

It is straightforward that there always exists a coloring of the balls (with colors blue and red), such that there is no edge between two balls of the same color; therefore the previously defined graph G is bipartite. It is also obvious that **Q** can show a ball of majority color if and only if there is a vertex x of G that always appears in the greater part in every bipartition of G and **Q** can declare that there is no majority if and only if in every bipartition of G the two parts have the same size.

Let us consider now the case $n \leq 2k-4$. Let G be a graph on $n \leq 2k-4$ vertices with an edge set consisting of $\lfloor n/2 \rfloor - 1$ independent edges. It is easy to see that any set of k vertices spans at least one edge, so the answers of **A** will be all negative and to every answer **A** can show a pair of balls that are neighbours in G.

Assume now to the contrary that **Q** can solve the problem, that is in every bipartition of G the two parts have the same size or there exists a vertex x of G that always appears in the greater part in every bipartition of G.

The first case is clearly impossible since there exists an isolated vertex in G. For the second case we prove a useful lemma.

Lemma 1. *Let G be a bipartite graph on n vertices. If there exists a vertex x of G that always appears in the greater part in every bipartition of G, then G has at least $\lfloor n/2 \rfloor$ edges.*

Proof of Lemma 1. Let C be the part of a bipartition of G that contains x. It is easy to see that $C \setminus \{x\}$ contains at least $\lfloor n/2 \rfloor$ non-isolated vertices, otherwise moving the isolated vertices of $C \setminus \{x\}$ to the other part of the bipartition would leave x in the smaller part, which is impossible. Notice that x itself may be an isolated vertex. Since there is no edge between two vertices of C, there must be at least $\lfloor n/2 \rfloor$ edges in G. This completes the proof of Lemma 1. □

By Lemma 1 the second case also leads to a contradiction, which shows that for $n \leq 2k-4$ the questioner **Q** cannot solve the problem.

Now let us turn our attention to the only remaining case, $n = 2k-3$. We show that **Q** can always solve the problem in this case. Since **A** must always answer NO, every set of k vertices of G spans at least one edge, in other words $\alpha(G)$, the maximum size of an independent vertex set of G is at most $k-1$. It is well-known that $\alpha(G) + \tau(G) = n$, where $\tau(G)$ is the minimum size of a vertex cover of G, thus $\tau(G) \geq n - k + 1$. G is a bipartite graph, so by the theorem of Kőnig [13] $\tau(G) = \nu(G)$, where $\nu(G)$ is the maximum size of an independent edge set of G. Thus there exists an independent edge set X of size $n - k + 1 = k - 2 = (n-1)/2$ in G. Now the ball corresponding to the only vertex that is not covered by X must be in majority, thus finishing the proof of Theorem 1. □

3. General model

It might be worth observing that in the General model while a NO answer on a pair of balls tells us that these balls have different colors and consequently they may be discarded by the questioner, when the number of balls is greater than two then a NO answer only tells us that there are (at least) two balls of different color. Therefore, if on the one hand it seems more advantageous comparing more balls at a time, on the other hand it is more challenging for the questioner to exploit a less informative NO answer.

First we give upper bounds and then we discuss lower bounds on the number of queries. We conclude with some examples.

3.1. The upper bound

In this subsection we give an algorithm for solving the majority problem. We start with a straightforward extension of a result of Saks and Werman [16].

Lemma 2. *For every odd value of n, we have $q_k(n) \leq q_k(n-1)$.*

Proof. The proof is basically the same as the proof of Saks and Werman for $k = 2$; we include it here for completeness. Let S be the set of balls. Let us remove a ball x from S. By definition we can solve the majority problem on $S \setminus \{x\}$ using $q_k(n-1)$ queries. Now there are two cases to consider. If $S \setminus \{x\}$ has no majority, then the number of blue and red balls in the set $S \setminus \{x\}$ is the same and therefore x has the majority color in the set S. On the other hand if $S \setminus \{x\}$ has majority, then since $n-1$ is even the number of balls of the majority color is at least $(n-1)/2 + 2$, therefore the majority color of S must be the same as the majority color for $S \setminus \{x\}$, proving $q_k(n) \leq q_k(n-1)$ for n odd. □

Let us now describe our algorithm.

Algorithm MAJORITY$_3$. Let $m = \lfloor n/4 \rfloor$. Consider an arbitrary partition of the n balls in m groups $G_1, G_2, \ldots, G_m$ of size 4 each, let R be the group of the remaining balls, and let $r = |R|$.

For $i = 1, 2, \ldots, m$, we make all $\binom{4}{3} = 4$ possible triple queries on the four balls of G_i until we possibly get a YES answer. There are two cases:

(a) for some j we get a YES answer for at least one of the 4 triple queries involving balls of G_j;
(b) we get always NO answers for all the $4m$ comparisons.

In case (a), we discard all balls contained in $G_1, \ldots, G_{j-1}$ (the NO answers on $G_1, \ldots, G_{j-1}$ imply an equal number of red and blue balls on the discarded balls). Let a, b, and c be the three balls in G_j that have the same color. From now on, all the remaining $n - 4j$ balls are compared, one at a time, with two balls of identical color, e.g. a and b.

In case (b), we discard all balls contained in $G_1, \ldots, G_m$ and concentrate on the remaining r balls in set R. The aim is to solve the majority problem for every $r \in \{0, 1, 2, 3\}$. We limit our analysis to $r = 0$ and $r = 2$ as, in view of Lemma 2, we have $q_3(4m + 1) \leq q_3(4m)$ and $q_3(4m + 3) \leq q_3(4m + 2)$.

If $r = 0$, the algorithm states that there is no majority.

If $r = 2$, let x and y be the two remaining balls. We compare x and y with three arbitrary balls a, b, c (one at a time) from G_1. Namely, we perform the following queries: $\{x, y, a\}$, $\{x, y, b\}$ and $\{x, y, c\}$. If we obtain a YES the algorithm shows either x or y as the majority ball, otherwise we always obtain NO answers and the algorithm states that there is no majority.

Lemma 3. *Let $n = 4m + r$ for some $r \in \{0, 1, 2, 3\}$ and $m \geq 1$.*

$q_3(4m) \leq 4m = n$;
$q_3(4m + 1) \leq 4m = n - 1$;
$q_3(4m + 2) \leq 4m + 3 = n + 1$;
$q_3(4m + 3) \leq 4m + 3 = n$.

Proof. We have to analyze both cases (a) and (b). We are allowed to limit our analysis to cases $r = 0$ and $r = 2$ as, in view of Lemma 2, a solution for $r = 0$ implies a solution for $r = 1$ and a solution for $r = 2$ implies a solution for $r = 3$ (i.e., $q_3(4m + 1) \leq q_3(4m)$ and $q_3(4m + 3) \leq q_3(4m + 2)$).

In case (a), we got always NO answers on all comparisons done on $G_1, \ldots, G_{j-1}$ and a YES answer on G_j, for some $1 \leq j \leq m$. The NO answers on $G_1, \ldots, G_{j-1}$ imply an equal number of red and blue balls among the $4(j - 1)$ balls contained in $G_1 \cup \cdots \cup G_{j-1}$. Therefore, we can safely discard all these balls as they have no effect for determining the majority. Let a, b, and c be the three balls in G_j that have the same color. Note that, in view of the 4 comparisons on G_j, we also know whether or not the color of the fourth ball d in G_j is the same as the color of the other three balls. The number of queries required up to this point is $4j$. From now on, all the remaining $n - 4j$ balls are compared, one at a time, with two balls of identical color, e.g. a and b. It is clear that this way we can count the number of balls colored like a and b and the number of balls colored like a different ball e, if it exists. This will allow us to solve the problem by using a total number of at most $4j + (n - 4j) = n$ queries.

In case (b) we get always NO answers on all the $4m$ comparisons. This implies an equal number of red and blue balls among the $4m$ balls in $G_1 \cup G_2 \cup \cdots \cup G_m$. Therefore, in this case, the majority is determined by the remaining r balls in set R. Hence, we have to determine, for every $r \in \{0, 1, 2, 3\}$, the number $q_3(4m + r)$.

If $r = 0$, the algorithm can state at this point that there is no majority. If $r = 2$, the problem reduces to ascertain whether these two remaining balls have the same color (in which case any of them is in the majority) or not (which implies that there is no majority). In order to do so, the algorithm compares the 2 remaining balls (call them x and y) with three arbitrary balls a, b, c (one at a time) from G_1 (queries $\{x, y, a\}$, $\{x, y, b\}$ and $\{x, y, c\}$). If x and y are identically colored, we obtain a YES (recall that, since we are in case (b), there must be two different balls among a, b, and c); otherwise we always obtain NO answers. Therefore, in both cases we can solve the problem using at most 3 additional queries. □

In the next section we will see that this bound is tight, that is $q_3(6) = 7$.

Lemma 4. *For every even value of n, we have $q_3(n) \leq q_3(n - 3) + 4$.*

Proof. Let S be the set of n balls. Take 3 arbitrary balls a, b, and c from S and let S' be the set of the remaining $n - 3$ balls. Let $n - 3 = 4m + r$ for some $m \geq 0$ and $r \in \{0, 1, 2, 3\}$. In the following we will refer to cases (a) and (b) as defined in algorithm MAJORITY$_3$. On the $n - 3$ balls in S' we apply an algorithm A which differs from algorithm MAJORITY$_3$ only in case (b) when for $r = 3$ there is no majority among the $4m$ balls. Instead of determining the majority on the remaining $r = 3$ balls, in this case algorithm A terminates without solving the majority problem, therefore saving the last 3 queries, i.e. using $q_3(n - 3) - 3$ queries. Let us refer to this case as the *different case*, while the *identical case* corresponds to all remaining situations in which the two algorithms behave the same way, i.e. case (a) and case (b) for $r = 0, 1, 2$ and for $r = 3$ when there is a majority among the first $4m$ balls (actually, since $n - 3$ is odd, $r = 1$ or $r = 3$).

We will analyze the two cases separately. The aim is to show that in both cases we can solve the majority problem on the whole set S by using at most $q_3(n - 3) + 4$ queries.

Identical case. In this case algorithm A solves the majority problem on the $n - 3$ balls in S' by using $q_3(n - 3)$ queries. We can observe that in this case, at the end of its execution, the algorithm is also able to output a number i such that there are i balls of one color and $n - i$ of the other color, such that $i > n - i$ (it suffices to note in the proof of Lemma 3 that the only case when algorithm MAJORITY$_3$ is not able to produce such a number is in case (b) when for $r = 3$ there is no majority on the first $4m$ balls). It is obvious that $i > (n - 4)/2$.

Let x be the majority ball provided by the algorithm on the $n - 3$ balls. Notice that x exists as $n - 3$ is odd. We have to show that with at most 4 more queries we can solve the majority problem on the whole set S of n balls. First we perform the query $\{a, b, c\}$.

Assume first that we got a YES answer on $\{a, b, c\}$. Now we perform query $\{a, b, x\}$. If we get a YES on $\{a, b, x\}$ we infer that x is a majority ball for S. If $\{a, b, x\}$ gives a NO answer, then three cases are possible:

if $i < \frac{n}{2}$ then any of a, b, and c can be shown as a majority ball;
if $i = \frac{n}{2}$ then there is no majority ball;
if $i > \frac{n}{2}$ then x is a majority ball.

If on the other hand we got a NO answer on $\{a, b, c\}$, we can solve the majority problem by using the following three additional queries: $\{a, b, x\}$, $\{a, c, x\}$ and $\{b, c, x\}$. If at least one of the three answers is YES then we can infer that x is a majority ball. If we get NO on all three queries, then we infer that there is no majority among a, b, c and x. Two cases are possible:

if $i = \frac{n-4}{2} + 1$, then there is no majority;
if $i > \frac{n-4}{2} + 1$, then x is a majority ball.

This proves that with a total of $q_3(n-3)+4$ queries we can solve the majority problem on the whole set S. This completes the proof for the identical case.

Different case. In this case, algorithm A discovers that there is no majority among the $4m$ balls and terminates after $q_3(n-3)-3$ queries, without analyzing the remaining $r = 3$ balls in S'. To complete the proof it will suffice to show that with at most 7 additional queries we can solve the majority problem on the original set S. Let a', b' and c' be the $r = 3$ remaining balls in S' that have not been analyzed by A. Since there is no majority among the $4m$ balls, all we have to do is to solve the majority problem on the 6 balls a, b, c, a', b' and c'. By Lemma 3 when $n = 6$, we know that this requires at most 7 queries. This completes the proof of the lemma. □

3.2. The lower bound

In the sequel, a *coloring* is a partition (R, B) of the set of balls into two sets R and B, where R is the set of red balls and B the set of blue balls.

In this section we give a lower bound on the number of queries needed to determine majority. Our aim will be to construct a worst case input coloring for any unknown correct algorithm that solves the majority problem. We use the well-known adversary lower bound technique.

We say that an adversary's strategy *admits a coloring* if such a coloring is consistent with all the answers provided by the adversary. As long as the strategy devised by the adversary admits alternative possible colorings that are consistent with at least two different possible solutions for the majority problem, the algorithm cannot correctly give its output. The goal of the adversary is to maximize, with its strategy of answers, the number of rounds until the algorithm can correctly give its output.

We will first consider the case of an even number n of balls. Given a sequence of queries $\mathcal{S} = (Q_1, \ldots, Q_t)$ and a positive integer $i \leq t$, let us define the following property:

$\mathcal{P}(i)$: for all $X \subseteq [n]$, with $|X| = m$, there exists $j \leq i$ such that one of the following conditions hold:
(a) $Q_j \subseteq X$;
(b) $Q_j \cap X = \emptyset$.

The adversary strategy. The strategy followed by the adversary is defined as follows. To every query Q_j, the adversary replies NO as far as $\mathcal{P}(j)$ does not hold. Let i be the first index for which $\mathcal{P}(i)$ holds, if it exists. Then there exists a set $X \subseteq [n]$, with $|X| = m$, such that $Q_j \not\subseteq X$ and $Q_j \not\subseteq \bar{X} = [n] - X$ for $j = 1, 2, \ldots, i-1$ (this is because i is the smallest index such that $\mathcal{P}(i)$ holds), and either $Q_i \subseteq X$ or $Q_i \subseteq \bar{X}$ (these are conditions (a) and (b) on $\mathcal{P}(i)$). From this point forth, the adversary replies YES to Q_i and all subsequent queries $\{a, b, c\}$ if all three elements a, b, c belong to the same set X or $\bar{X}$, and replies NO to all the other queries.

Lemma 5. *For all n even we have $q_3(n) \geq n - 1$.*

Proof. Let $n = 2m$ for some $m > 1$. Assume that $Q_1, \ldots, Q_t$ is the sequence of queries that the algorithm produces before giving its output. We distinguish two main cases: either $\mathcal{P}(t)$ does not hold or $\mathcal{P}(t)$ holds.

Case 1. *$\mathcal{P}(t)$ does not hold.* In this case there exists $X \subseteq [n]$, with $|X| = m$, such that for all queries $Q_1, \ldots, Q_t$ neither (a), nor (b) holds. According to its strategy, the adversary replies always NO in this case.

If the algorithm states that there is a majority, the algorithm actually has to show a ball in X or in $[n] \setminus X$. The adversary's strategy admits the following coloring: let red be all the balls in X and blue all the remaining balls. Indeed, such a coloring is consistent with all the NO answers provided by the adversary, as since property $\mathcal{P}(t)$ does not hold, there is no query entirely contained in X or in $\bar{X}$. This means that there is no majority, which contradicts the algorithm.

Let us assume the algorithm states that there is no majority. Suppose that there exists a ball x in X such that $Q_i \not\subseteq \bar{X} \cup \{x\}$ for every $i = 1, 2, \ldots, t$. In this case, the adversary's strategy allows us to produce the following coloring: color red all the balls in $\bar{X} \cup \{x\}$ and blue the others. Indeed, it is easy to observe that, since $Q_i \not\subseteq \bar{X} \cup \{x\}$ for every $i = 1, 2, \ldots, t$, such a

coloring is consistent with the NO answers provided by the adversary to each Q_i for $i = 1, 2, \ldots, t$. This implies that there is a majority: once again the claim of the algorithm is contradicted. The case in which there exists a ball x in $\bar{X}$ such that $Q_i \not\subseteq X \cup \{x\}$ for every $i = 1, 2, \ldots, t$, is similar. It remains to analyze the case when both the following conditions hold:

(1) for every $x \in X$ there exists a query Q_i for some $i \in \{1, 2, \ldots, t\}$, such that $Q_i \subseteq \bar{X} \cup \{x\}$;

(2) for every $x \in \bar{X}$ there exists a query Q_i for some $i \in \{1, 2, \ldots, t\}$, such that $Q_i \subseteq X \cup \{x\}$.

Condition (1) implies that the sequence of queries $Q_1, Q_2, \ldots, Q_t$ must contain a query $\{x, x_1, x_2\}$ for every $x \in X$ and for some $x_1, x_2 \in \bar{X}$. Analogously, condition (2) implies that there must be a query $\{x', x'_1, x'_2\}$ for every $x' \in \bar{X}$ and for some $x'_1, x'_2 \in X$. Since all these queries are clearly distinct, $t \geq 2m = n$.

Case 2. $\mathscr{P}(t)$ *holds*. In this case, the adversary replies NO to all the queries Q_j, as far as $\mathscr{P}(j)$ does not hold, and replies YES to the first query Q_i for which $\mathscr{P}(i)$ holds. Since $\mathscr{P}(i)$ holds, while for $j = 1, 2, \ldots, i-1$ $\mathscr{P}(j)$ does not hold, there exists $X \subseteq [n]$, $|X| = m$, such that either $Q_i \subseteq X$ or $Q_i \subseteq \bar{X}$, and

$$Q_j \not\subseteq X \quad \text{and} \quad Q_j \not\subseteq \bar{X} \text{ for } j = 1, 2, \ldots, i-1. \tag{1}$$

In the following, we assume without loss of generality that $Q_i \subseteq X$. Notice that this time the adversary's strategy admits any coloring consistent with all the NO answers given to $Q_1, Q_2, \ldots, Q_{i-1}$ and the YES answer to Q_i.

If the algorithm concludes that there is a majority, it can be easily contradicted by observing that the adversary's strategy admits a coloring such that all the balls in X are red and all the others are blue.

Therefore, suppose the algorithm states that there is no majority. If there exists a ball $x \in X \setminus Q_i$ (resp. $y \in \bar{X}$) such that for every $j = 1, 2, \ldots, i$, $Q_j \not\subseteq \bar{X} \cup \{x\}$ (resp. $Q_j \not\subseteq X \cup \{y\}$), the adversary's strategy admits the following coloring. Color red all the balls in $X \setminus \{x\}$ (resp. $\bar{X} \setminus \{y\}$) and blue the balls in $\bar{X} \cup \{x\}$ (resp. $X \cup \{y\}$), so to have a majority in the latter set. This contradicts the algorithm's claim.

It remains to analyze only the case when both the following conditions hold:

(1*) for every $x \in X \setminus Q_i$ there is a query Q_j for some $j \in \{1, 2, \ldots, i-1\}$, such that $Q_j \subseteq \bar{X} \cup \{x\}$;

(2*) for every $y \in \bar{X}$ there is a query Q_j for some $j \in \{1, 2, \ldots, i-1\}$, such that $Q_j \subseteq X \cup \{y\}$.

Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be the set of queries necessary for (1*) and (2*) to be satisfied, respectively. Condition (1*) implies that the sequence of queries $Q_1, Q_2, \ldots, Q_{i-1}$ must contain at least a query $\{x, y, z\}$ for every $x \in X \setminus Q_i$ and for some $y, z \in \bar{X}$. Condition (2*) implies that there must be also at least a query $\{x', y', z'\}$ for every $x' \in \bar{X}$ and for some $y', z' \in X$. Therefore,

$$|\mathcal{F}_1| + |\mathcal{F}_2| \geq 2m - 3. \tag{2}$$

In order to estimate the total number of queries, we need also to consider the set $\mathcal{F}_3$ of queries that involve balls in $Q_i = \{a, b, c\}$ and are not considered in $\mathcal{F}_1$ and $\mathcal{F}_2$. The query Q_i clearly belongs to $\mathcal{F}_3$, so

$$|\mathcal{F}_3| \geq 1. \tag{3}$$

Now we have to distinguish two cases.

Case (i). There is no query $Q' = \{x, y, z\}$, such that $x \in \bar{X}$ and $y, z \in Q_i$. In this case, $\mathcal{F}_3$ must contain at least another query Q that contains some element from $\{a, b, c\}$ and two elements from $\bar{X}$, otherwise the adversary's strategy would admit a coloring where all the balls in $\bar{X} \cup \{a, b, c\}$ are red, which would imply the existence of a majority in this set. Hence, $\mathcal{F}_3 \geq 2$ and by (2) the total number of queries is $|\mathcal{F}_1| + |\mathcal{F}_2| + |\mathcal{F}_3| \geq 2m - 1$.

Case (ii). There exists a query $Q' = \{x, y, z\}$ such that $x \in \bar{X}$ and $y, z \in Q_i$. Let s be the number of queries involving one ball from Q_i and two balls from $\bar{X}$. We may assume that $s < 2$, otherwise $|\mathcal{F}_3| \geq s + 1 \geq 2$ and the proof would be complete. If $s < 2$, we must have that

(*) there exist two balls a and b in Q_i such that there is no query involving a or b with two balls from $\bar{X}$.

Let us consider $x, y, z \in Q'$ and recall that $x \in \bar{X}$ and $y, z \in Q_i$. By (*) we have that

(**) there exists a ball $v \in \{y, z\} \subseteq Q_i$ such that there is no query $\{v, w_1, w_2\}$ with $w_1, w_2 \in \bar{X}$.

Moreover, since we are assuming that $|\mathcal{F}_1| + |\mathcal{F}_2| = 2m - 3$, condition (2*) implies that

(***) for every $x \in \bar{X}$ there is *exactly one* query $Q_x = \{x, y_x, z_x\}$ that includes x and some $y_x, z_x \in X$.

Let $Y = X \setminus \{v\} \cup \{x\}$. Next we will show that there is no query among $Q_1, Q_2, \ldots, Q_i$ that is entirely contained in Y or $\bar{Y}$, which contradicts the hypothesis that $\mathscr{P}(t)$ holds with i being the smallest index such that for all $X \subseteq [n]$, with $|X| = m$, there exists $j \leq i$ such that either $Q_j \subseteq X$ or $Q_j \subseteq \bar{X}$.

Since now $x \in Y$ and $v \in \bar{Y}$, we can easily verify that $Q_i \not\subseteq Y$ and $Q_i \not\subseteq \bar{Y}$. It remains to show that no query among $Q_1, Q_2, \ldots, Q_{i-1}$ is entirely contained in Y or $\bar{Y}$.

As a consequence of (**), there is no query involving v entirely contained in $\bar{Y}$. Moreover, by (***) Q' must be the only query involving $x \in \bar{X}$ and two balls in X. Hence, there is no query involving x entirely contained in Y. Let $\mathcal{Q}$ be the set of remaining queries to analyze, i.e. the queries not involving v nor x. By the definition of Y and $\mathcal{Q}$, it is straightforward to observe that for all $Q \in \mathcal{Q}$, $Q \subseteq Y$ (resp. $Q \subseteq \bar{Y}$) if and only if $Q \subseteq X$ (resp. $Q \subseteq \bar{X}$). Therefore, since i is the smallest index such that $\mathscr{P}(i)$ holds, there is no query in $\mathcal{Q}$ entirely included in Y or in $\bar{Y}$. This is a contradiction and the proof is now complete. □

Lemma 6. *For all n odd we have* $q_3(n) \geq n - 2$.

Proof. Let us consider a set S of m balls with m even. By Lemma 4, we have $q_3(m) \leq q_3(m-3)+4$. Recalling that for m even, we know by Lemma 5 that $q_3(m) \geq m-1$, we have $q_3(m-3)+4 \geq q_3(m) \geq m-1$, which implies $q_3(m-3) \geq m-5$. The lemma follows by letting $n = m-3$. □

Now we can state the main result concerning the General model.

Theorem 2. *Let $n = 4m + r$ for some $r \in \{0, 1, 2, 3\}$ and $m \geq 1$. We have*

$n - 1 \leq q_3(4m) \leq n$;
$n - 2 \leq q_3(4m + 1) \leq n - 1$;
$n - 1 \leq q_3(4m + 2) \leq n + 1$;
$n - 2 \leq q_3(4m + 3) \leq n$.

Proof. The proof follows immediately by combining Lemma 3 with Lemmas 5 and 6. □

Tightening the bounds of Theorem 2 so as to compute the exact value of $q_3(n)$ does not seem to be easy, but we provide the exact values of $q_3(n)$ for $n \leq 6$ in Section 3.3.

3.3. Number of queries in the general model, for $n \leq 6$

Determing the precise value of the number of queries in the general model appears to be a rather demanding problem even when the number of balls is small. In the sequel we compute the number of queries when the number of balls is small ($n \leq 6$).

First of all observe that if $n \leq 3$ then the Questioner cannot solve the problem.

Lemma 7. $q_3(4) = 4$.

Proof. By Theorem 2 4 queries are enough (actually it is easy to see it without the theorem). On the other hand, 3 queries are not enough, since if the answers are always NO, we cannot decide whether there is no majority or the balls in that triple that was not asked have the same color and the fourth ball has a different color. □

Lemma 8. $q_3(5) = 4$.

Proof. By Theorem 2 4 queries are enough (it is also easy to see it without the theorem). Assume now to the contrary that asking 3 triples are enough. The adversary always answers NOand now those colorings where there are 3 red balls and 2 blue balls are all consistent with the answers, except those ones where the 3 red balls form one of the 3 triples that were asked. Since from 5 balls $\binom{5}{3} = 10$ different triples can be formed, we have 7 possibilities for the set of the red balls. This means that there is no ball which appears in all 7 such triples (since there are only $\binom{4}{2} = 6$ triples in which a certain ball appears), thus though there must be a majority type ball, the Questioner cannot name any ball as one in majority, finishing the proof. □

Lemma 9. $q_3(6) = 7$.

Proof. Lemma 3, when $n = 6$, tells us that 7 queries are sufficient for 6 balls. Now we show that 6 queries are not enough for 6 balls. Suppose the contrary. Since not all 10 pairs of complementary triples are asked, the 3 red, 3 blue situation is always possible if all the answers are NO. We show that if the triples are not in a special setting (described later), then the 4 blue, 2 red situation is also possible if all the answers are NO.

If the 4-2 situation is not possible with just NO answers, then every 4-tuple contains a triple we asked. In other words, the complements of our triples contain every possible pair of balls. So we would like to cover the edges of the complete graph on 6 vertices with 6 (or less) triangles. Since a triangle covers 2 edges of the same end-vertex (i.e. triangle 123 covers 12 and 13) and every vertex has degree 5 in K_6 (which is odd), this is possible if and only if three edges forming a perfect matching are doubly covered and all the others are simply covered (6 triangles have 18 edges and we have 15 edges (to see that 5 triangle is not enough is even easier)). Let the doubly covered edges be 12, 34, 56.

Now we give an adversary strategy: the first five questions are answered NO, the sixth is NO, if the above setting is not satisfied, otherwise it is YES. In the first case both the 3-3 and the 4-2 situations are possible. We show the same for the second case. Now all triples contain either 12, 34, or 56 (since we have six triples and these edges are doubly covered), so we may suppose that the last triple was 456 (that is, the last question was 123). We know that 1, 2, and 3 are of the same color, but the other balls may all have the other color, since the only question (answered NO) that is not consistent with this is the question 456 and it is easy to see that this cannot be a question, because in this case we have the triples 123 and 456 and all other four triples have just one ball in common with one of them, so there are four doubly covered edges, which is impossible. It remains to show that the 4-2 situation is also possible. We know that 123 is a question, let the other question whose complement covers 56 be $1234 \setminus \{x\}$, where x is 1, 2, or 3. Let y be an element from 1, 2, 3 different from x. Then the answers are consistent with the situation that 4, 5, 6, and y are blue, the others are red. This finishes the proof. □

4. Pairing model

In this section we focus on the Pairing model. First we prove a lower bound on k-tuples that does not depend on k and then show that the bound is tight for $k = 3$.

4.1. A general lower bound

First we prove a lemma.

Lemma 10. *Let $n \geq 2k-3$, $k \geq 3$. Then*

$$q_k^p(n) \geq \begin{cases} n/2+1 & \text{if } n \text{ is even, and} \\ \lfloor n/2 \rfloor & \text{if } n \text{ is odd.} \end{cases}$$

Proof. Before proceeding with the main proof, observe that without loss of generality we may allow the adversary to color the balls as long as the coloring is consistent with its answers. This simple observation does not alter the number of queries required and at the same simplifies the proof of correctness on the number of queries required.

First we describe a strategy for the Adversary which guarantees that the number of queries the questioner should use is at least $\lfloor n/2 \rfloor$ if n is odd. The strategy is quite simple, **A** always answers NO whenever it is possible, that is, if it is not known before the query that all k balls asked have the same color (in which case the answer is YES, of course, but such a question will never be asked). **A** also has to show two balls of different color; these can be any pair of balls that may have different color before the query.

To see that **Q** has to use at least $\lfloor n/2 \rfloor$ queries even against this simple strategy, we use graphs to describe the knowledge of **Q** (this is the same graph we use in the proof of Theorem 1). In this Pairing model to every negative answer of **A** there corresponds a pair of balls having different colors. Let G be a graph such that the balls represent the vertices and the edges correspond to these pairs of balls of different colors.

By the strategy of **A** there exists a coloring of the balls (with colors blue and red) such that there is no edge between two balls of the same color; therefore the graph G is bipartite. Suppose now that **Q** can show a ball of majority color. This is possible if and only if there is a vertex x of G that always appears in the greater part in every bipartition of G. (Notice that now there is a majority color, since n is odd.) Now it is easy to see that G has at least $\lfloor n/2 \rfloor$ edges, thus the number of queries is also at least $\lfloor n/2 \rfloor$.

Now we slightly modify the above strategy to obtain the lower bound for n even. **Q** can declare that there is no majority if and only if in every bipartition of G the two parts have the same size, which is impossible if there is an isolated vertex in G.

Suppose that **A** answers the first $n/2-1$ questions the same way as above. We consider two cases.

Case 1. The edges of G after the $n/2-1$ queries are not independent. Now the answer to the next query is also answered the same way as above. Since there is an isolated vertex in G (we have $n/2$ edges that are not independent), **Q** cannot declare that there is no majority. Moreover, no vertex x can always appear in the greater part in every bipartition of G, because there are only $n/2$ edges in G (in the part where such an x would appear, there would be at least $n/2+1$ non-isolated vertices). This completes the proof in Case 1.

Case 2. The edges of G after the $n/2-1$ queries are independent. Now **A** has to be careful, since an edge between the two remaining isolated vertices would guarantee that there is no majority. So the edge is drawn somewhere else, which is possible, because $k \geq 3$ and there is no cycle in G (that is, **A** may draw the edge between any two of the k vertices of the query). Now the situation is the same as in Case 1. This completes the proof. □

4.2. Determining $q_3^p(n)$

Now we prove that the bound we have just proved in Lemma 10 is tight for $k=3$. This is interesting because of several reasons. The strategy of **A** that gave the lower bound is quite simple and the lower bound is much smaller than the value of $q_2(n)$ (as well as $q_3(n)$). However, the most surprising is that the function q_3^p is not increasing, since $q_3^p(2n) = n+1$ and $q_3^p(2n+1) = n$. Actually, we have seen that q_k is not strictly increasing and the same proof shows that the same is true for q_k^p: $q_k^p(2n+1) \leq q_k^p(2n)$ and $q_k(2n+1) \leq q_k(2n)$. Nevertheless, a complexity function that is not even increasing is quite unusual in search theory.

The main theorem of this section is the following.

Theorem 3. *Let $n \geq 3$. Then*

$$q_3^p(n) = \begin{cases} n/2+1 & \text{if } n \text{ is even, and} \\ \lfloor n/2 \rfloor & \text{if } n \text{ is odd.} \end{cases}$$

Proof. We just have to show that the bounds in Lemma 10 are tight, that is, give an algorithm for the questioner **Q** that uses $n/2+1$ queries if n is even and $\lfloor n/2 \rfloor$ queries if n is odd.

The algorithm consists of two main parts of which the first is the formation of bins with balls and is similar to the one in [16]. Initially place all the n balls in n different bins and at each step do the following.

1. Select any 3 equal size bins and pick one representative from each of them and query these 3 balls.
2. If the answer is YES, then merge the bins into one new bin.
3. If the answer is NO, then **A** specifies two balls of different colors. Remove the two corresponding bins.
4. Iterate the process until there are no 3 bins of equal size.

It is obvious that all remaining bins have size a power of 3. The process stops because we can no longer find 3 bins of equal size, that is, each bin size occurs at most twice. It is easy to check that to build a bin of size 3^c the questioner **Q** needs $(3^c - 1)/2$ queries, while to remove two bins of size 3^c each, the questioner **Q** needs exactly 3^c queries. Thus if the total number of removed balls is $2t$, then **Q** used t queries in order to get rid of them.

This means that after the first main step of the algorithm, **Q** has $n - 2t$ balls in bins whose sizes are powers of 3, bin sizes occur at most twice and **Q** used t queries to remove the $2t$ balls and $(3^c - 1)/2$ queries to build a bin of size 3^c. In other words, if the bin size 3^i occurs a_i times ($i = 0, 1, 2$) then the sequence $a_s, a_{s-1}, \ldots, a_0$ is the ternary expansion of $n - 2t$ (where $s = \lceil \log_3(n - 2t) \rceil$) and **Q** used $t + \sum_{i=0}^{s} a_i(3^i - 1)/2)$ queries altogether.

It is obvious that if $a_s = 0$ (which occurs iff $n - 2t = 0$), then there is no majority, if $a_s = 1$, then the corresponding bin contains majority type balls, while if $a_s = 2$, then **Q** cannot show a majority type ball, neither can prove that there is no majority at the moment, so we may suppose that $a_s = 2$ (otherwise we are done, since $2t + \sum_{i=0}^{s} a_i(3^i - 1) \leq n$, so for the number of queries we have that $t + \sum_{i=0}^{s} a_i(3^i - 1)/2) \leq n/2$).

Now starts the second part of the algorithm, where **Q** tries to eliminate the remaining bins. If $s > 0$, then **Q** chooses one ball from both bins of size 3^s and a third ball from one of these bins and queries this triple. Given the answer, **Q** learns if the two biggest bins have balls of the same color or not. If the answer is YES, then all balls in these two bins are clearly in majority, if the answer is NO, then **Q** removes the two bins. Iterate this process if necessary (if the greatest bin size occurs twice), until it is possible, that is, while $s > 0$. To remove two bins of size 3^c each, **Q** still needs exactly 3^c queries, therefore it is clear that at the end of the second part **Q** can either solve the problem (that is, find a majority ball or show that there is no majority) using at most $n/2$ queries or **Q** has just two bins of size 1 and used exactly $(n - 2)/2$ queries to reach this position (which is possible only if n is even). Now to finish the algorithm and the whole proof let the two remaining balls be a and b and let c and c' be two balls of different color (such balls exist and are known to **Q**, since $n \geq 3$ and n is even). Now **Q** queries the balls a, b, c. If the answer is YESor the answer is NOand the two differently colored balls shown by **A** is a and b, then **Q** can solve the problem using $n/2$ queries altogether. If the answer is NOand the two different balls shown by **A** are not a and b, then **Q** queries the balls a, b, c', thus solving the problem using $n/2 + 1$ queries altogether. This completes the proof of Theorem 3. □

5. Conclusion

In this paper we studied the minimum number of triple queries needed to determine the majority color in a set of n balls colored with two colors in two models, depending on the type of answers of the oracle. In addition to tightening the bounds for the majority problem in the General model, several interesting questions remain open for further investigation, including computing majority for a) k-tuple queries, for some fixed k, b) bins with balls colored with more than two colors, as well as c) other natural computation models, e.g. where some bounded number of lies of the oracle is allowed.

Acknowledgments

We would like to thank the anonymous referee for numerous useful suggestions that significantly improved the overall structure and presentation of the main ideas of the paper. The second author was supported in part by NSERC and MITACS grants. The third author was supported in part by the Hungarian National Research Fund and by the National Office for Research and Technology (Grant Number OTKA 67651).

References

[1] M. Aigner, Combinatorial Search, John Wiley & Sons, Inc, 1988.
[2] M. Aigner, Variants of the majority problem, Discrete Applied Mathematics 137 (1) (2004) 3–25.
[3] M. Aigner, Two colors and more, in: Entropy, Search, Complexity, 2007, pp. 9–26.
[4] M. Aigner, G. De Marco, M. Montangero, The plurality problem with three colors, in: STACS 2004, 2004, pp. 513–521.
[5] L. Alonso, P. Chassaing, EM Reingold, R. Schott, The chip problem, preprint. Available at, http://emr.cs.uiuc.edu/~reingold/chips.ps.
[6] L. Alonso, E.M. Reingold, Determining plurality, ACM Transactions on Algorithms 4 (3) (2008) 1–19.
[7] L. Alonso, E.M. Reingold, R. Schott, Determining the majority, Information Processing Letters 47 (5) (1993) 253–255.
[8] L. Alonso, E.M. Reingold, R. Schott, The average-case complexity of determining the majority, SIAM Journal on Computing 26 (1997) 1–14.
[9] F. Chung, R. Graham, J. Mao, A. Yao, Oblivious and adaptive strategies for the majority and plurality problems, Algorithmica 48 (2007) 147–157.
[10] G. De Marco, A. Pelc, Randomized algorithms for determining the majority on graphs, in: Mathematical Foundations of Computer Science 2003, 2003, pp. 368–377.
[11] Z. Dvořák, V. Jelínek, D. Král, J. Kynčl, M. Saks, Probabilistic strategies for the partition and plurality problems, Random Structures & Algorithms 30 (1–2) (2007) 63–77.
[12] M. Fisher, S. Salzberg, Finding a majority among n votes, Journal of Algorithms 3 (1982) 375–379.
[13] D. König, Graphen und matrizen, Mat. Fiz. Lapok 38 (1931) 116–119.
[14] D. Král, J. Sgall, T. Tichy, Randomized strategies for the plurality problem, Discrete Applied Mathematics 156 (2008) 3305–3311.
[15] F.P. Preparata, G. Metze, R.T. Chien, On the connection assignment problem of diagnosable systems, Electronic Computers, IEEE Transactions on 16 (12) (1967) 848–854.
[16] M.E. Saks, M. Werman, On computing majority by comparisons, Combinatorica 11 (4) (1991) 383–387.
[17] G. Wiener, Search for a majority element, Journal of Statistical Planning and Inference 100 (2) (2002) 313–318.