Int. J. Appl. Math. Comput. Sci., 2012, Vol. 22, No. 3, 765–778
DOI: 10.2478/v10006-012-0057-y

amcs

THE FAN–RASPAUD CONJECTURE: A RANDOMIZED ALGORITHMIC APPROACH AND APPLICATION TO THE PAIR ASSIGNMENT PROBLEM IN CUBIC NETWORKS

PIOTR FORMANOWICZ [*,**], KRZYSZTOF TANAŚ [*]

[*] Institute of Computing Science
Poznań University of Technology, Piotrowo 2, 60-965 Poznań, Poland
Krzysztof.Tanas@cs.put.poznan.pl

[**]Institute of Bioorganic Chemistry
Polish Academy of Sciences, Z. Noskowskiego 12/14, 61-704 Poznań, Poland

It was conjectured by Fan and Raspaud (1994) that every bridgeless cubic graph contains three perfect matchings such that every edge belongs to at most two of them. We show a randomized algorithmic way of finding Fan–Raspaud colorings of a given cubic graph and, analyzing the computer results, we try to find and describe the Fan–Raspaud colorings for some selected classes of cubic graphs. The presented algorithms can then be applied to the pair assignment problem in cubic computer networks. Another possible application of the algorithms is that of being a tool for mathematicians working in the field of cubic graph theory, for discovering edge colorings with certain mathematical properties and formulating new conjectures related to the Fan–Raspaud conjecture.

Keywords: cubic graph, edge coloring, perfect matching, randomized algorithms, computer networks.

1. Introduction

We study edge colorings of cubic graphs. A proper edge coloring of a graph $G = (V, E)$ is a coloring of all edges such that, for every pair of edges $e1, e2 \in E$, if $e1$ and $e2$ are incident (share a common vertex), then $e1$ and $e2$ have different colors. The smallest number for which there exists a proper edge coloring for a given graph $G = (V, E)$ is called the *chromatic index* of the graph G and is denoted by $\chi^{'}(G)$. For cubic graphs, the chromatic index is either 3 or 4, what has been proven by Vizing (1964). Thus there exist cubic graphs which cannot be 3-edge colored in a classic (proper) way. The class of cubic graphs for which $\chi^{'} = 4$ (that is, the class of cubic graphs which do not have a proper 3-edge coloring) has a special name, i.e., *snark graphs*, or simply *snarks*.

Because of the above fact, we turn our attention to less restricted 3-edge colorings, where for every $(vertex, color)$ pair there still exists exactly one adjacent edge containing the given color but, unlike in a proper edge coloring, the edges of the graph are allowed to contain two, but not all three colors. The edges are also allowed to remain uncolored. In terms of graph theory, we are coloring cubic graphs by three perfect matchings with empty intersection.

It was conjectured by Fan and Raspaud (1994) that every bridgeless cubic graph can be colored in the above way. Thus, we call any coloring which meets the specified conditions a 'Fan–Raspaud coloring'. If the edges were allowed to contain all three colors, the coloring would be trivial—it would be enough to find just one perfect matching and represent it with all three colors.

In this paper, we propose an algorithm which constructs Fan–Raspaud colorings for a given cubic graph. The resulted colorings may represent, for example, pair assignments of computers for parallel computations in a cubic network. The colorings can also be studied to explore their mathematical properties and form new conjectures related to the Fan–Raspaud conjecture.

The edge coloring has been studied in many papers. For example, Fouquet and Vanherpe (2008) proved that the conjecture holds true for all graphs with fewer than 32 vertices. It was also proven by Holyer (1981) that the 3-edge-coloring problem of a cubic graph is NP-complete.

The paper is organized as follows. In Section 2 we

provide some basic definitions and conjectures concerning Fan–Raspaud colorings. In Section 3 some mathematical properties of these colorings are presented. In Sections 4–6 we describe the proposed algorithms. One of the possible applications of the Fan–Raspaud coloring is shortly discussed in Section 7, while in Section 8 the classes of graphs used for testing the proposed algorithms are briefly described. The paper ends with conclusions and some open problems in Section 9.

2. Basic definitions and conjectures

Definition 1. A *cubic graph* is a graph with all vertices of degree 3 (a 3-regular graph).

Definition 2. A *snark* is a cubic graph $G = (V, E)$ for which $\chi^{'} = 4$, that is, a cubic graph without a proper 3-edge coloring.

Definition 3. A *perfect matching* of a graph $G = (V, E)$ on $n = 2k$ vertices, $k \in \mathbb{N}_+$, is a set of edges $\{(a_1, b_1), \ldots, (a_k, b_k)\}$ where $a_1, \ldots, a_k, b_1, \ldots, b_k$ are all pairwise distinct vertices.

Definition 4. A *girth* of a graph $G = (V, E)$ is the length of the shortest cycle in the graph G. If a graph contains no cycles, then its girth is infinity.

Conjecture 1. (Fan and Raspaud, 1994) *For every bridgeless cubic graph $G = (V, E)$ there exist three perfect matchings M_1, M_2, M_3 such that $M_1 \cap M_2 \cap M_3 = \emptyset$.*

Definition 5. A *Fan–Raspaud coloring* of a cubic graph $G = (V, E)$ is a set of three perfect matchings $\{M_1, M_2, M_3\}$ satisfying the Fan–Raspaud conjecture, that is, for every $e \in E$ there exists $i \in \{1, 2, 3\}$ such that $e \notin M_i$.

Fact 1. *The Fan–Raspaud coloring is a generalized 3-edge coloring. It is obvious that every proper 3-edge coloring of any bridgeless cubic graph is a Fan–Raspaud coloring. In a Fan–Raspaud coloring, however, a single edge can have assigned two (but not three) colors.*

3. Mathematical properties of Fan–Raspaud colorings

In this section we describe the studied mathematical properties of the Fan–Raspaud colorings, such as the bicolored cycle structure and the Fan–Raspaud disorder, and prove some facts connected with those properties.

3.1. Bicolored (alternating) cycle structure.

Lemma 1. *For every Fan–Raspaud coloring $\{M_1, M_2, M_3\}$ of any cubic graph G and for every pair (a, b), $a, b \in \{1, 2, 3\}, a \neq b$, there exists a set of vertex-disjoint cycles $C_{a,b}$ of even length which covers all vertices of the graph G, except for vertices having an incident edge containing both the colors a and b (we call these edges 'sticks' and describe them later in the paper), and for every cycle in $C_{a,b}$ edges are alternating with colors a and b.*

Proof. Let v_1 be an arbitrarily chosen vertex, $v_1 \in V(G)$, and let $a, b \in \{1, 2, 3\}$, $a \neq b$ be any two colors used in the coloring. From the definition of the Fan–Raspaud coloring there exists an edge adjacent to the vertex v_1 belonging to the perfect matching M_a (that is, painted with color a). Let it be the edge (v_1, v_2). From the definition there also exists an edge adjacent to v_2 belonging to M_b. Let it be the edge (v_2, v_3). Now, we can repeat the procedure for each visited vertex with alternating colors a and b. Since the graph G is cubic, which means G has no dead ends (degree 1 vertices), we can always continue the procedure and visit the next vertex on our way, and since $n = |V(G)|$ is finite, we will always return to an already visited vertex (v_1) after a finite number of steps. Since every cycle of odd length requires at least three colors for a proper edge coloring, and our procedure allows travelling through edges of only two colors while the coloring of our graph is a proper edge coloring, the cycle we have constructed has even length. ■

The set of cycles produced by our procedure forms a *bicolored cycle structure* of the cubic graph G.

Fact 2. *Some of the bicolored cycles of a bicolored cycle structure of a Fan–Raspaud coloring may not be normal cycles, but* 'sticks' *(single edges with some two colors a and b).*

When counting the bicolored cycle structure of a given cubic graph G, we assume that a stick is a cycle of length 2 *for the pair of colors it contains.*

Lemma 2. *For every snark G, every Fan–Raspaud coloring $\{M_1, M_2, M_3\}$ contains at least one stick for every pair of colors (a, b), $a, b \in \{1, 2, 3\}$, $a \neq b$.*

Proof. It is derived by contradiction. Let us assume that G is a snark and for some Fan–Raspaud coloring $\{M_1, M_2, M_3\}$ there exists a pair of colors a, b with no edge having both the colors a and b. That implies that the bicolored cycle structure of the snark G, Fan–Raspaud coloring M_1, M_2, M_3 and colors a, b consists of standard cycles only. From this, we know that the whole graph G (all vertices) can be covered by a set of vertex-disjoint standard bicolored cycles (we say that a vertex is covered when for every of the three colors there exists an adjacent edge containing that color). Thus, every vertex of the graph has exactly two adjacent colored edges, both single-colored.

Since G is cubic, every vertex has exactly one not yet colored adjacent edge, so the set of all uncolored edges

forms a perfect matching. Coloring it with the third color would complete a proper 3-edge coloring of the graph G. So, G has a proper 3-edge coloring. Therefore G is not a snark. ■

3.2. Fan–Raspaud cycle structure: The 'disorder'.

Definition 6. Every Fan–Raspaud-coloring $A = \{M_1, M_2, M_3\}$ which is not a proper 3-edge coloring contains a set of edges colored in a way not allowed in a standard coloring, that is, non-single-colored ones. These edges have 0 or 2 colors.

Let us call the set of non-single-colored edges in the coloring A a *Fan–Raspaud disorder* (or just *disorder*) of the coloring A.

Lemma 3. *For every Fan–Raspaud coloring the disorder is a set of vertex-disjoint cycles of even length.*

Proof. It is analogical to the bicolored cycle structure proof, using the same procedure by going through double-colored and empty edges, although in this case the cycles will not cover the entire graph. ■

4. Randomized algorithm approach and its advantages

Since we know that the proper 3-edge coloring problem for cubic graphs is NP-complete, we suspect that the Fan–Raspaud coloring problem for cubic graphs is also NP-complete. If this were true, this would mean there did not exist a deterministic polynomial time algorithm for this problem, unless $P = NP$. We then turn our attention to randomized methods and construct algorithms which try to find a Fan–Raspaud coloring for a given cubic graph by coloring the edges of the graph in a way determined by a series of random draws.

As the Fan–Raspaud conjecture is already solved for small graphs (Fouquet and Vanherpe, 2008), the algorithm we construct must be able to effectively handle larger cubic graphs, too. The algorithms we construct belong to the *Monte-Carlo class* of randomized methods.

4.1. Basic algorithms components. The constructed algorithms are based on the idea of edge-coloring of a given cubic graph with recursive place nonplace scanning technique performed every time after adding a color to an edge. When the recursions have finished (and the result of the attempt to find a Fan–Raspaud coloring is undetermined yet), we perform a random draw which will decide which edge to color next. The algorithms are arranged in such a way that it is never needed to make a random draw with more than two possible numbers (except the selection of the starting vertex, but in this case it is chosen by the user (or by default, we start from the vertex 0), rather than being randomly drawn).

We have constructed and tested two algorithms. The first one is a 'simple' version. In this algorithm the *place-nonplace scanning procedure* (described later in Section 5.2) does not check the number of empty (without any assigned colors yet) edges adjacent to the scanned vertex, and we do not make a distinction whether the adjacent edge is empty or single-colored (although double-colored edges are always considered 'blocked', since the Fan–Raspaud conjecture forbids any edge of the graph to have three colors).

The second algorithm is the 'minimum disorder' algorithm (or we can call it the 'max-single-colored-edge' algorithm). In this algorithm the place/nonplace scanning procedure's decision is different for various numbers of (still) empty edges adjacent to the scanned vertex. In this algorithm we seek to create as many normal (single-colored) edges as possible, creating double-colored edges only when there is no place to fit some color for the scanned vertex without creating a new double-colored edge.

4.2. Definitions, procedures and functions used in both algorithms.

• *Assigning colors to vertices.* Obviously, in a Fan–Raspaud coloring we color edges, not vertices, of a given cubic graph. However, for technical reasons, since the graph $G = (V, E)$ is described (in the input file) as a set of triples $[v.s[0], v.s[1], v.s[2]]$, $v \in V(G)$, where $v.s[i], i \in \{0, 1, 2\}$ is the number of v's neighbor on position i, the current state of the graph is implemented as an array of vertices, each of which has stored information about the neighbor numbers and positions of each of the three colors. From the definition of perfect matching, we know that no vertex may have more than one adjacent edge containing the same color, and in a complete coloring for every pair (v, k), $v \in G$, where k is a color, there exists exactly one $i \in \{0, 1, 2\}$ such that the edge $(v, v.s[i])$ contains the color k. Therefore it is safe to store information about the current state of the coloring in the above way.

• *Starting vertex.* For both algorithms, we need to choose a vertex to start the algorithm from. The starting vertex is chosen by the user (or by default, we start from the vertex 0). The number of the starting vertex is denoted as sv. Initially, we color all the edges incident to the starting vertex with one color each. Permutation of the colors is irrelevant, so we assume that the edge connecting the neighbor of the starting vertex in position 0 has the first color, in position 1—the second color, and in position 2—the third color. In the sample results shown later in the article, the first color is dark gray, the second—middle gray and the the third—light gray. The empty (uncolored) edges are painted with dashed lines in the examples.

Definition 7. For our algorithm, we say that *a color k is assigned to a vertex $v \in V(G)$ on position $i, i \in \{0,1,2\}$* if and only if the edge $(v, v.s[i])$ contains the color k. In a more general way, we can say that *a color k is assigned to a vertex $v \in V(G)$* if and only if *any* edge adjacent to v (already) contains the color k.

In the presented algorithm, positions of the colors for each vertex $v \in V(G)$, at the current state of the coloring, are stored as $[v.kol[0], v.kol[1], v.kol[2]]$. If the vertex v has not been assigned the color k yet, then $v.kol[k] = 255$. Initially for every $v \in V(G)$, $k \in \{0,1,2\}$, $v.kol[k] = 255$, except for values which correspond to the edges adjacent to the starting vertex.

• *Partially Done Vertices (PDV) list.* During the progress of the algorithm we need to keep information about the vertices which were already visited (and assigned any color(s)). If a series of recursive colorings finishes but the result of the coloring (success or failure) is undetermined yet, we need to know which vertex (and color) we should start from now making a random draw. For this purpose, we construct the PDV (Partially Done Vertices) list. From now on, we will refer to the element of the list in the position x as *PDV[x]*. The first element is *PDV[0]*. There is also a function *PDVL* to get the current length of the *PDV* list. Initially, for both versions of the algorithm, the PDV list contains three vertices—the neighbors of the starting vertex.

• *Binary code.* To store information about the results of all random draws made during the progress of the algorithm, the graph is assigned a $CODE$ array. In general, *CODE[i]* is a result of the $i+1$-th random draw (we start from *CODE[0]*). There is also a *CODEL* function to get the length of the code at present. Initially, $CODEL = 0$ and $CODE = \emptyset$. Due to specification of the algorithms, there are always exactly two possible edges to add the color of the draw to the vertex assigned for the draw, hence every random draw performed in the algorithm is a binary draw, and thus $CODE[i] \in \{0,1\}$ for every $i \in \{0, \ldots, CODEL - 1\}$.

• *Places and nonplaces.*

Definition 8. A *place* of a vertex v_1 adjacent to v is a *place* for color k if and only if it is legal to add the color k to the edge (v, v_1).

Thus, for the Fan–Raspaud coloring algorithm, a vertex v_1 is a place if and only if v_1 does not have (already) assigned color k (because in any perfect matching there cannot be two same-color edges sharing a common vertex), and the (v, v_1) edge does not already have two assigned colors (because the Fan–Raspaud conjecture does not allow any edge to be three-colored). We can always assume that the vertex v has not been assigned the color k yet, because both the presented algorithms perform the PNS procedure only for vertices which are still without the scanned color. If a vertex is not a place for a given color, it is a *nonplace*.

• *Prefix tree.* A *prefix tree* is a structure which stores the data about all the possible binary codes for the given cubic graph and specified starting vertex. Every leaf of the prefix tree has an associated binary code and a Boolean value determining whether the coloring was successful or not (so the prefix tree has 'good' and 'bad' leaves). By counting the number of good leaves among all the leaves and their binary code lengths, we can determine the probability of finding a complete Fan–Raspaud coloring for any given cubic graph and a given starting vertex.

To construct a prefix tree, we can modify any of the presented algorithms in such a way that we do not make random draws at all, but every time a draw would be made in the unmodified algorithm, split the graph G into two graphs G and G' instead, keeping G's current coloring state for both newly formed graphs. Then add 0, and 1 to the binary codes of G and G', respectively, and make colorings such as if the added digits were the results of a random draw in the unmodified algorithm. Then we can continue the algorithm recursively for G and G'.

A sample prefix tree looks like the set of binary prefix codes below, for the first Celmins–Swart snark (Fig. 1) used as an example:

```
FIRST_CELMINS-SWART_SNARK
starting vertex = 8
minimum disorder (md) version
00 N
010 N
0110 N
0111 N
1000 N
1001 N
101 N
110 Y
111 N
```

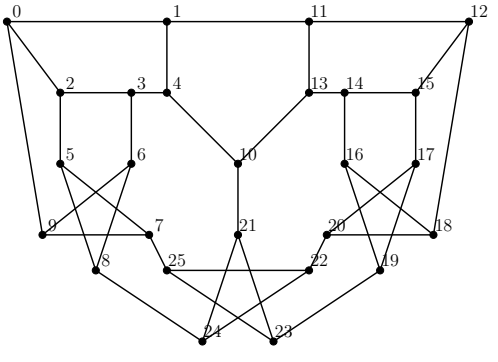

Fig. 1. First Celmins–Swart snark.

• *Explanation of the listed prefix tree.* The prefix tree listed above shows results of the algorithm in the minimum disorder (md) version for the first Celmins–Swart snark with vertices numbered as shown in Fig. 1. We have chosen the vertex number 8 as the starting vertex, and colored each of its incident edges with one color. Permutation of colors is irrelevant, so we assume that the colors are assigned as follows: $color0 = darkgray$, $color1 = middlegray$, $color2 = lightgray$. We have obtained a coloring position as shown in Fig. 2.

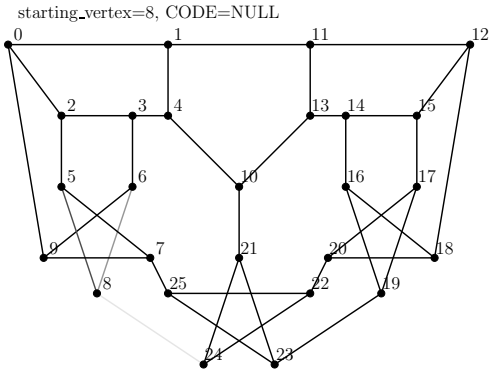

Fig. 2. Starting position for a given example.

In the above coloring position there is neither a certain edge to add any color, nor has the coloring ended (either in a success or failure). The first random draw (vertex, color) pair found is $(5, middlegray)$. The neighbors of the vertex 5 are $2, 7, 8$. The nonplace for the pair $(5, middlegray)$ is the vertex number 8, so the position of the nonplace is 2. We draw a random bit. If a 0 is drawn, we add the middle gray color in position $2 +_3 1 = 0$, and if a 1 is drawn—in position $2 +_3 2 = 1$. Therefore we add the gray color to the edge $(5, 2)$ or $(5, 7)$ depending on the result of the random draw performed.

Let us now show a situation after the last draw before conclusion of the coloring, where one result of the draw leads to a successful Fan–Raspaud coloring and the other to a failure.

In the above situation the algorithm has already drawn two 1's in earlier stages of the coloring. We have reached a position without a certain $(edge, color)$ pair, and the $(vertex, color)$ pair for a draw was $(24, darkgray)$. The cases when a 0 and a 1 were drawn, thus adding the dark gray color to the $(24, 21)$ or the $(24, 22)$ edge, are shown on the left and the right, respectively. In this example, the 0-case will eventually form a successful Fan–Raspaud coloring, while the 1-case will end in a failure. Let us show final coloring states for both specified cases Fig. 4:

The left part shows a successful Fan–Raspaud coloring for the given example, represented by the 110 code

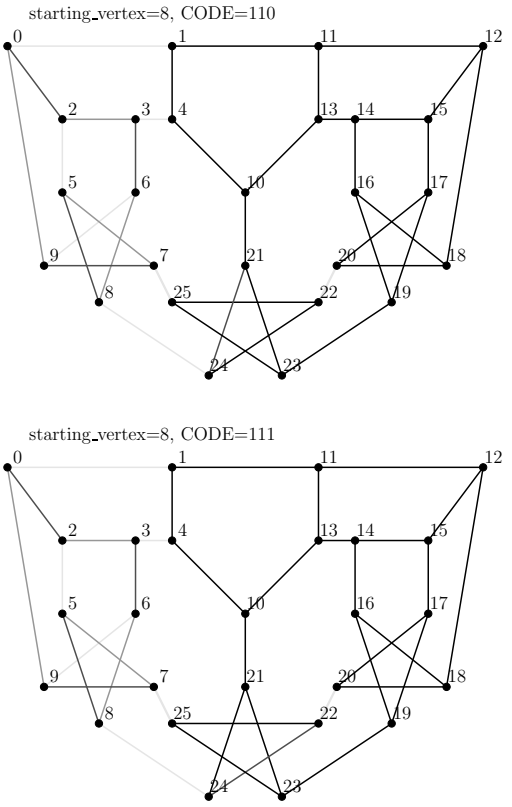

Fig. 3. Positions after the last draw, one leading to a successful Fan–Raspaud coloring and the other ending in a failure.

in the sample prefix tree. The right part shows a failed coloring represented by the code 111. We know that the coloring ended in a failure, because there is no legal place for the $(15, lightgray)$ pair. All three positions for the vertex number 15 and the light gray color are nonplaces.

Fact 3. *The prefix tree can be used to compute the efficiency of the algorithm.*

• *The OP[ ] array.* *OP[ ]* is a temporary globally declared array containing the list of operations on the PDV list created during each recursive series of colorings (temporary list of vertices to add to the PDV list after the recursion has finished). It is implemented for technical reasons. The present length of the OP array is stored in the variable OPL.

4.3. Procedures and functions used in the algorithm.

- int $NCOL(v)$: Returns the number of colors already assigned to the vertex v.

- bool $COL2(v, p)$: Returns $TRUE$ if and only if the edge adjacent to the vertex v on position p ($p \in \{0, 1, 2\}$) has (already) assigned two colors.

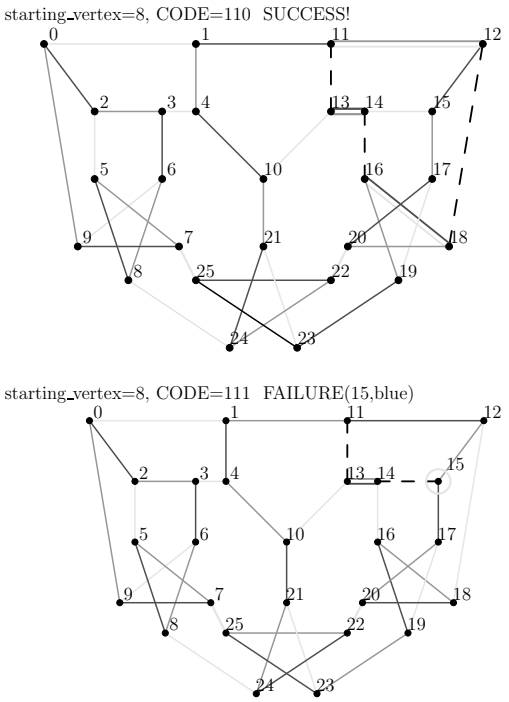

Fig. 4. Successful Fan–Raspaud coloring (left) and a failed one (right).

- int $POZ(v_1, v_2)$: Returns the position ($\in \{0, 1, 2\}$) of the vertex v_2 among the neighbors of v_1 (or 255 if the vertices v_1 and v_2 are not adjacent).

Algorithm 1 $PNS(v, k)$

```
pl = 0; // the number of places
p = −1; // position of the place, relevant only when
there is exactly one place
for (i = 0, 2) do
    if ((COL2(v, i) == false)
    and (v[v.s[i]].kol[k] == 255)) then
        pl = pl + 1; p = i;
    end if
end for
if (pl == 0) then
    PRINT_RESULT(FAILURE); STOP;
end if
if (pl == 1) then
    COLOR(v, v.s[p], k);
end if
```

Place/Nonplace Scanning (PNS) procedure. A *place/nonplace scanning* is the checking of possible places (among the neighbors of a given vertex) where it is possible to legally add the scanned color to this vertex. In the simple version any of the scanned vertex's neighbors is a place for the color k if and only if the color k has not been assigned there yet, and the edge connecting the scanned vertex and neighbor is not blocked (i.e., it is not double-colored). If the procedure finds that there are no places for the scanned color, we already know that we will not be able to complete the Fan–Raspaud coloring. This time the randomized algorithm has failed. If there is exactly one place, we are sure where to fit the color k for the vertex v and then we proceed to the (recursive) coloring procedure. If there are two or more possible places to add the color k to the vertex v, we cannot determine the color's location yet.

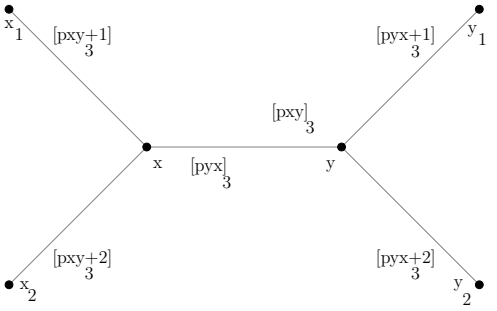

Fig. 5. Vertices x and y and their neighbors with marked position numbers.

Recursive coloring procedure. Every time the algorithm reaches the beginning of this procedure, a color k is added to the edge (x, y).

Now, for both other neighbors of the vertices x and y (let us name them x_1, x_2 and y_1, y_2, respectively) we check if the vertex is still free for the color k and the edge connecting the scanned vertex and neighbor is not blocked (already double-colored), and if so, we will go to the place/nonplace scanning procedure. Also, for the vertices x and y, check if the vertex has exactly two (already) assigned colors, and if so, go to the PNS procedure for this vertex and the only lacking color (let us call this color l). The pair of vertices (x, y) and their neighbors with marked position numbers *(mod 3)*, where $pxy = POZ(x, y)$ and $pyx = POZ(y, x)$, are shown in the Fig. 5.

5. Algorithms

5.1. Basic algorithm. In the basic algorithm we try to create random Fan–Raspaud colorings, without giving extra constraints on the coloring. This way we can use the algorithm to seek Fan–Raspaud colorings, and discover the ones with interesting properties, without limiting the set of possible colorings to any specific subset.

Fan–Raspaud coloring: The basic algorithm—code. In the basic algorithm we do not check the number of empty edges when performing a place/nonplace scanning or before the random draw step. This algorithm generally

Algorithm 2 $COLOR(x, y, k)$

for (a={x,y}) **do**
  if $(NCOL(a) == 0)$ **then**
    $OPL = OPL + 1$; $OP[OPL - 1] = a$;
  end if
end for
$pxy = POZ(x, y)$; $pyx = POZ(y, x)$;
$x.kol[k] = pxy$; $y.kol[k] = pyx$;
$x_1 = v[x].s[(pxy + 1) \bmod 3]$;
$x_2 = v[x].s[(pxy + 2) \bmod 3]$;
$y_1 = v[y].s[(pyx + 1) \bmod 3]$;
$y_2 = v[y].s[(pyx + 2) \bmod 3]$;
for $(\{a, b, c\} = \{x, pxy + 1, x_1\}, \{x, pxy + 2, x_2\}, \{y, pyx + 1, y_1\}, \{y, pyx + 2, y_2\})$ **do**
  if $((COL2(a, (b \bmod 3)) == false)$ **and** $(c.kol[k] == 255))$ **then**
    $PNS(c, k)$;
  end if
end for
for $(a = \{x, y\})$ **do**
  if $(NCOL(a) == 2)$ **then**
    $PNS(a, l)$; // l is the only lacking color for the vertex x or y, respectively
  end if
end for

Algorithm 3 $FAN - RASPAUD\_STARTUP(G, sv)$

Graph G**:** $\{v[n] : s[3], kol[3]\}$
$CODEL = 0$; $CODE = \emptyset$; $PDVL = 3$;
for $(i = 0, 2)$ **do**
  $v[sv].kol[i] = i$;
  $(v[sv].s[i]).kol[i] = POZ(v[sv].s[i], sv)$
  $PDV[i] = v[sv].s[i]$;
end for
// elsewhere $kol[i] = 255$;
$FAN\_RASPAUD\_COLORING(G)$;

gives colorings with more non-single-colored edges than the minimum disorder algorithm, but is more likely to find a complete Fan–Raspaud coloring (with greater probability of success).

5.2. 'Minimum disorder' algorithm. In this algorithm we try to create a Fan–Raspaud coloring of a given cubic graph with as small Fan–Raspaud disorder as possible. Thus, we try to maximize the number of 'classic' (single-colored) edges in the coloring. As many steps of the md algorithm are performed in the same way as in the basic algorithm, we will limit the description of the md algorithm to the steps and procedures which work differently.

- *Place-nonplace scanning procedure.* The PNS procedure in the minimum disorder algorithm works quite differently than in the basic algorithm. The definitions of place and nonplace remain the same, but there is a priority for scanning for places among those neighbors for which the edge connected to the scanned vertex is

Algorithm 4 $FAN - RASPAUD\_COLORING(G)$

if $(PDVL > 0)$ **then**
  $rdf = false$; // random draw found? $(true/false)$
  $lk, lw, lw(0), lw(1); p = PDV[0]$;
  if $(NCOL(p) == 3)$ **then** // the fully-collored vertex case
    for $(i = 1, PDVL - 1)$ **do**
      $PDV[i - 1] = PDV[i]$;
    end for
    $PDVL = PDVL - 1$;
    Let emp be the number of empty edges adjacent to p
    if $(emp == 1)$ **then**
      Let a be the position of the only empty edge for p
      $q = v[p].s[a]$; $kt = POZ(q, p)$;
      $x_1 = v[q].s[(kt + 1) \bmod 3]$;
      $x_2 = v[q].s[(kt + 2) \bmod 3]$;
      if $(NCOL(x_1) + NCOL(x_2) == 0)$ **then**
        // A safeguard against locking by an
        // all-empty-edge edge cut.
        $lk = 0$; $lw = q$; $lw(0) = x_1$;
        $lw(1) = x_2$; $rdf = true$;
      end if
    end if
  else // the non-fully-colored vertex case
    Let lk be the lowest-numbered color for which vertex p has exactly 2 places.
    Let nm be the position of the only nonplace for the pair (p, k).
    $lw = p$; $lw(0) = v[p].s[(nm + 1) \bmod 3]$;
    $lw(1) = v[p].s[(nm + 2) \bmod 3]$;
  end if
  if $(rdf == true)$ **then**
    // the random draw
    $x = random, \quad x \in \{0, 1\}$.
    $CODEL = CODEL + 1$;
    $CODE[CODEL - 1] = x$;
    $COLOR(lw, lw(x), lk)$;
    // recursive coloring sequence
    $MERGE\_OP$;
    // merge OP array with the end of the PDV list
  end if
  $FAN - RASPAUD\_COLORING(G)$;
  // for the modified graph, after coloring sequence
else // if the PDV list is empty
  $PRINT\_RESULT(SUCCESS)$; $STOP$;
end if

empty. If there is exactly one place among the empty-edge neighbors, we proceed directly to the recursive scanning procedure for that place and the current color. We scan the other neighbors only if there are no places among the empty-edge neighbors. If there are at least two empty-edge places or no empty-edge places and two non-empty-edge places, we cannot determine the location of the current color yet. If all three neighbors are nonplaces, the algorithm returns a 'failure' result as in the basic algorithm.

• *Non-fully colored vertex case.* The non-fully colored vertex case step is also different in this algorithm. To determine the position of the nonplace for the random draw step we consider two cases, depending on the number of empty edges adjacent to the vertex p:

(i) One empty adjacent edge: In this case we know that the vertex p is lacking only one color, both the non-empty-edge neighbors are places and the empty-edge neighbor is a nonplace (otherwise, the algorithm would have found a sure coloring or a failure earlier, before reaching this step).

(ii) Two empty adjacent edges: In this case we have two empty-edge places and a non-empty-edge nonplace (otherwise the algorithm would have found a sure coloring or a failure earlier, before reaching this step).

Thus, in both cases the position of the nonplace for the next random draw (let us call it $\mathbf{nm}$), $nm \in \{0, 1, 2\}$, is the position of the edge with a *distinct state of emptiness* (the empty edge in Case (*i*) or the non-empty edge in Case (*ii*)).

Fact 4. *The minimum disorder algorithm does not always find the Fan–Raspaud coloring with minimum possible disorder for the given cubic graph. If this version succeeds, hovewer, then the coloring found is usually a good approximation.*

The above fact is confirmed by results shown in the efficiency tables, constructed by running the mimimum disorder algorithm for the tested snarks, constructing a prefix tree for every possible starting vertex and counting the total number of the Fan–Raspaud colorings for each possible length of the Fan–Raspaud disorder. For the tested graphs the average length of the Fan–Raspaud disorder in successful colorings is less than 9 and most of the found colorings have disorder of length 6, except for the Goldberg snark graphs, for which most of Fan–Raspaud colorings found have length 8.

6. Properties and analysis of the algorithms

In the previous section we described the algorithms and the mathematical properties of the resulting Fan–Raspaud colorings. Now we describe the properties of the algorithms.

6.1. Time complexity.

• *Coloring procedure.* In a Fan–Raspaud coloring of a cubic graph of order n there are always 3 colors, and $n/2$ edges with each color. Thus, the coloring procedure must be applied $3n/2$ times (or less when the coloring fails).

• *Place/nonplace scanning procedure.* After every coloring of any edge the place-nonplace scanning procedure is applied several times. Now let us compute the maximum possible number of scannings for any vertex. Let $x \in V(G)$ be an arbitrarily chosen vertex of the cubic graph G. The vertex x is scanned for places-nonplaces if and only if (i) any of x's three neighbors receives a new color and (ii) x receives a double-colored edge.

Every time any neighbor receives a color which is already assigned to any other neighbor of the vertex x, x has at most one place for that color. Hence x will either be assigned that color or had no place to legally fit this color and the algorithm will end in a failure. Thus, for m colors Case (i) will occur at most $2m$ times. For a Fan–Raspaud coloring $m = 3$, so $2m = 6$. Case (ii) can happen only once because for three colors a vertex can have only one adjacent double-colored edge. Since the vertex x is arbitrarily chosen, we know that any vertex will be scanned at most 7 times during the progress of the presented coloring algorithms.

• *Random draw phase.* The random draw phase (with determining the vertex and the color to draw and its possible places) has always $O(1)$ steps.

• *Total.* Totalling the complexities of all the procedures, we can now determine that the time complexity of the whole algorithm (in an unmodified version, when we make random draws, without splitting the graph to find the prefix tree) is $O(n)$.

6.2. Memory complexity.

• *Graph data.* Since we know that the graph is cubic, there is no need to store the full adjacency matrix of the graph. We only need to store $3n$ numbers being every vertex's neighbor indices. The data about the position of each of the three colors has only 4 possible values, so for all the vertices and colors the memory needed here is $3n \cdot 2 = 6n$ bits.

• *PDV + OP lists.* The Partially Done Vertices (PDV) list plus the temporary OP list of operations to be made on the list (new vertices to add) have a combined length less than n at any time during the run of the algorithm.

• *Binary code.* The binary code contains the results of all the random draws. Theoretically, there are at most

$3n/2$ times that the coloring procedure must be applied, and every draw result needs just one bit of memory, hence the length of the binary code will never exceed $3n/2$ bits $3n/16$ bytes.

• *Depth of recursion in the coloring procedure.* Since there are no more than $3n/2$ times that the coloring procedure is applied and information on every use of the coloring procedure (its vertices and color) has $O(1)$ bytes, at any time while running the algorithm the memory complexity of the recursive coloring does not exceed $O(n)$.

• *Total.* Counting the total needed memory for the algorithm, in both versions, we find that the memory complexity of the shown algorithm is $O(n)$.

6.3. Efficiency. The efficiency of the algorithm is measured as a probability of successfully finding the complete Fan–Raspaud coloring for a given cubic graph and starting vertex. It can be measured by the classical probability of hitting any of the 'correct' leaves of the prefix tree.

For the minimum disorder algorithm another possible measure of efficiency can be applied by counting the average number of non-single-colored edges (that is, the average size of the Fan–Raspaud disorder) of the colorings represented by the correct leaves of the prefix tree of a given cubic graph.

The efficiency of the algorithms can also be computed using other measures. Computing using classic probability is not always the most accurate option, as the algorithm uses different amounts of time for different sequences of random draws, that is, for reaching different leaves of the prefix tree (whether they are 'correct' or 'incorrect' leaves). Hence, if we include the time optimization, another measure would be needed to the effectively compute efficiency of our algorithm. Efficiency tables and results are shown in Appendix B.

6.4. Validity of the algorithm. We now prove that the simple version of the algorithm can find a Fan–Raspaud coloring if one exists. For the starting vertex coloring problem, for any given current state of the coloring, before each random draw we have exactly two possibilities (possible places) for a particular color to add. Thus, the continuation of coloring for a given starting vertex and current code plus a 0 covers all possible colorings (successful or failed) with the drawn color added to one of the places, and for a 1 added to the code—for the other case. Given the generality of the position, we can recursively prove that the simple version of the algorithm covers all the possible states of the coloring. We assign three single-colored edges $(1, 1, 1)$ to the starting vertex because it is known that, if there exists a Fan–Raspaud coloring with all the $(2, 1, 0)$-edge vertices for a given cubic graph, the graph is not a snark (we can just convert edges with the same number of colors to three new perfect matchings, which form a classic 3-edge coloring). Of course, if the algorithm is unable to find a Fan–Raspaud coloring for a given starting vertex with 3 single-colored edges (statistically a very rare situation), we can start from any other vertex (the starting vertex number is given as a parameter of the algorithm).

7. Application of Fan–Raspaud colorings in the pair assignment problem in cubic networks

In this section we show a possible application of Fan–Raspaud colorings to assigning pairs of computers in a cubic network for a possible research project consisting of 3 parallel computations, represented by colors of edges.

7.1. Cubic network as a computer network topology. A cubic network is one of the possible topologies for computer networks. The mathematical model of such a network is a cubic graph. Its advantages include a constant vertex degree and (generally) a low graph diameter. The Fan–Raspaud conjecture also requires the cubic graph to be bridgeless, as a cubic graph with a bridge (cut-edge) may contain no perfect matching at all. However, a graph representing a cubic computer network should be bridgeless, as it is generally not a wise idea to design networks in such a way that a failure of just one connection (cable) would lead to a disconnection of the whole network and make communication between some computers and others unavailable.

7.2. Pair assignment problem in a cubic network. Let us assume that we are operating on a bridgeless cubic computer network, which is used for doing distributed programming computations, that must be performed by pairs of computers linked by a network connection. Suppose that the network is computing a distributed research problem consisting of n parallel subproblems. Assume that the data used in the computations have to be transferred, so we want the computations to be performed by pairs of computers. Of course, any two computers can be assigned as a pair for computing any of the subproblems if, and only if, they are directly linked by a network connection. Let us assume that the cables, or some other objects representing the network connections, have a transfer limit, which allows at most m computations to have data transferred at the same time by one network connection (or exceeding the limit would reduce the speed of the computatiom below a satisfactory level).

In terms of graph theory, we are trying to find n perfect matchings in a bridgeless cubic graph, such that no edge of the graph belongs to more than m of them.

Fact 5. *For $m = 1$, the problem turns into finding a proper $n-$edge coloring. For $m = 2, n = 3$, we seek a Fan–Raspaud coloring.*

If $m = 1$, we would be seeking n disjoint perfect matchings, and the problem would be associated with classic edge coloring. Most bridgeless cubic graphs can be 3-edge-colored in a classic way (see Fig. 6). However,

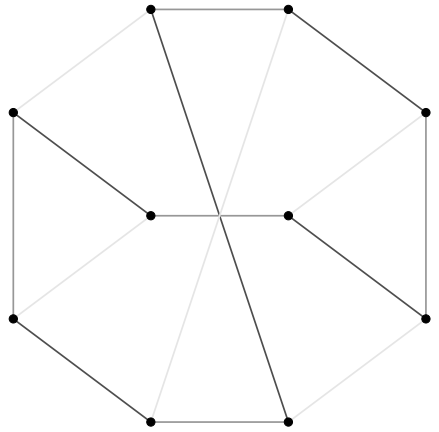

Fig. 6. Proper 3-edge-colored cubic graph.

because of the existence of *snarks*, not every bridgeless cubic graph has a proper 3-edge coloring (see Fig. 7).

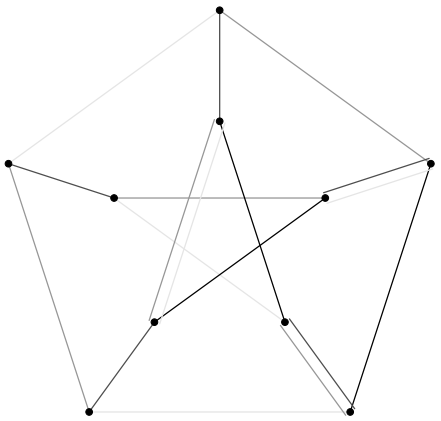

Fig. 7. Petersen graph has no proper 3-edge coloring. This is an example of a Fan–Raspaud coloring.

Because of the previously mentioned facts, we turn our attention to the case when $m = 2$.

Fact 6. *For $m = 2, n = 3$, the problem turns into finding a* Fan–Raspaud coloring *for a given graph. Unless the Fan–Raspaud conjecture turns out to be false, it is possible to assign pairs of computers for three parallel computations with every computer being assigned to a pair for every of the three computations, and there are no edges (network cables) overloaded by transferring data for all three of them.*

7.3. When to use which algorithm. Depending on the additional condition associated with the pair assignment problem in a cubic computer network, it varies which of the presented algorithms would be the best to use. We consider the following two cases

(i) Network connections may be assigned to 2 computations without loss of efficiency. In this case the number of overloaded network cables (represented by double-colored edges) is unimportant. In this case we use the basic algorithm, which has greater probability of finding a complete Fan–Raspaud coloring without specific properties.

(ii) Assigning 2 computations to a single connection would reduce efficiency of the computed problem. In this case we seek an assignment with minimum possible overloaded connections (double-colored edges). Hence we use the **md** algorithm.

8. Snarks as the chosen class of cubic graphs for the algorithm

The graphs which we have used as the input while testing the algorithm are snarks (although, technically, both the algorithms can handle any cubic graph). The reason is the following fact and its implications.

Fact 7. *Every proper 3-edge coloring of any cubic graph is a Fan–Raspaud coloring.*

The above fact implies that the Fan–Raspaud conjecture is true for all the cubic graphs with a proper 3-edge coloring. Hence, the only interesting graphs when exploring the Fan–Raspaud conjecture, and the only possible counterexamples, are snarks. Although it has been proven (Holyer, 1981) that it is an NP-complete problem whether a given cubic graph is a snark, we can test the algorithm for many graphs which are known to be snarks.

8.1. Classes of snarks used for testing the algorithms. As the Fan–Raspaud conjecture is open only for snarks, the snarks are the graphs used as the input for the presented algorithm. We now describe various subclasses of snarks, for which the algorithm has found a Fan–Raspaud coloring (shown in the figures in Appendix A).

- *Flower snarks.* Isaacs (1975) found an infinite family of snarks of order $8n + 20$,$n \in \mathbb{N}$, named respectively "Flower snark $J_5, J_7, J_9, \ldots$". A sample Fan–Raspaud coloring of the Flower snark 5 is shown in Fig. 8.

- *Goldberg snarks.* Goldberg (1981) found an infinite family of snarks of order $16n + 24$, $n \in \mathbb{N}$. These snarks are now named respectively "Goldberg snark $3, 5, 7, \ldots$. Because the Goldberg snark 3 is trivial (contains triangles), as the input graphs for the studied algorithms we use only Goldberg snarks 5 and higher. A sample Fan–Raspaud coloring of the Goldberg snark 5 is shown in the Fig. 9.

- *Szekeres snarks.* Another famous and one of the oldest known snarks is the 50-vertex snark found by Szekeres (1973), now known as the Szekeres snark. Furthermore, the Szekeres snark can be generalized into an infinite family of $50 + 40n$-vertex snarks, $n \in \mathbb{N}$, called "generalized Szekeres snarks". The construction of generalized Szekeres snarks is shown by Watkins (1989). An example of a Fan–Raspaud coloring of the classic (50-vertex) Szekeres snark is shown in Fig. 13.

- *Watkins snark.* Another well-known 50-vertex snark is the Watkins snark, found by Watkins and Wilson (1988). An example of a Fan–Raspaud coloring of the Watkins snark is shown in the Fig. 12.

- *Celmins-Swart snarks.* The Celmins–Swart snarks are two 26-vertex snarks discovered by Celmins and Swart (1979). Examples of Fan–Raspaud colorings of the Celmins–Swart snarks 1 and 2 are shown in Figs. 10 and 11, respectively.

Fact 8. *Studying the resulting Fan–Raspaud colorings for the subclasses of snarks listed in the paper, it is possible to create a generalized mathematical construction of a Fan–Raspaud coloring. This, combined with the proof of validity of the algorithm shown in Section 6.4, proves that the algorithm finds Fan–Raspaud colorings for whole classes of snarks, not just for individual ones.*

9. Conclusion

The class of cubic graphs plays an important role in graph theory and related fields of research, with various applications. A survey of cubic graphs is well described by Greenlaw (1995). In this paper we have studied edge coloring, which is a well-known subfield of graph theory, and presented algorithms seeking a kind of generalized 3-edge colorings known as Fan–Raspaud colorings, which can represent a mathematical model of pair assignments for parallel computations in cubic computer networks. The performance results of both algorithms are shown in Appendix B. For nearly all tested classes of graphs, the minimum disorder version has a high probability of finding a coloring with disorder of length 6 (which is a minimum possible for a snark), except for Goldberg snarks, for which the most often found disorder length is 8, but it is still a good approximation.

Another possible application of the implemented algorithms is that of being a tool for mathematicians working in the field of cubic graph theory, which allows discovering Fan–Raspaud colorings with interesting mathematical properties and formulate new related conjectures, some of which we propose below.

9.1. Proposed generalization of the Fan–Raspaud conjecture and related problems. Having performed the presented algorithm for various snarks and studied the resulting Fan–Raspaud colorings and their properties, we describe an interesting property of the Fan–Raspaud colorings, the minimum number of cycles in a Fan–Raspaud coloring of a given snark and related problems.

• *Minimum number of cycles in the Fan–Raspaud disorder.* By testing the algorithm we have found that for the tested classes of snarks there exists a Fan–Raspaud coloring with a disorder consisting of only one small cycle, usually a 6-cycle.

However, Kochol (1996) found a method (using graph superposition) to construct snarks of large (> 6) girth. That implies that there does not exist a constant number k such that for every snark there exists a Fan–Raspaud coloring with the Fan–Raspaud disorder of length not greater than k.

Nevertheless, it is not known whether there exists a constant number k such that for every snark there exists a Fan–Raspaud coloring with the disorder consisting of at most k cycles (of any length). Thus, after testing the described algorithms for various classes of snarks (and individual snarks) appearing i the literature, we propose a generalization of the Fan–Raspaud conjecture, which we call the K-cycle Fan–Raspaud conjecture.

• *K-cycle Fan–Raspaud conjecture.* If there does not exist a one-cycle-disorder Fan–Raspaud coloring for any snark, does there exist a minimal number $k > 1$ for which there exists a Fan–Raspaud coloring with the disorder having no more than k cycles, for every snark?

Conjecture 2. *For every snark there exists a Fan–Raspaud coloring whose Fan–Raspaud disorder contains no more than k cycles.*

• Finding the subclass of non l-cycle Fan–Raspaud-colorable snarks. Another interesting problem is to find and define the subclass of snarks which contain no Fan–Raspaud coloring with disorder consisting of at most l cycles, $l = \{1, 2, \ldots\}$. As most snarks have a Fan–Raspaud coloring with a single-cycle disorder (which has been found by performing the descibed algorithms on various classes of snarks), we think that the most interesting case to study would be for $l = 1$.

References

Celmins, U.A. and Swart, E. (1979). The constructions of snarks, *Research Report CORR*, No. 18, Department of Combinatorics and Optimization, University of Waterloo, Waterloo.

Fan, G. and Raspaud, A. (1994). Fulkerson's conjecture and circuit covers, *Journal of Combinatorial Theory, Series B* **61**(1): 133–138.

Fouquet, J.-L. and Vanherpe, J.-M. (2008). On Fan–Raspaud conjecture, *CoRR—Computing Research Repository*, abs/0809.4821.

Goldberg, M.K. (1981). Construction of class 2 graphs with maximum vertex degree 3, *Journal of Combinatorial Theory, Series B* **31**(3): 282–291.

Holyer, I. (1981). The NP-completeness of edge coloring, *SIAM Journal on Computing* **10**(4): 718–720.

Isaacs, R. (1975). Infinite families of nontrivial trivalent graphs which are not Tait colorable, *American Mathematical Monthly* **82**(3): 221–239.

Kochol, M. (1996). Snarks without small cycles, *Journal of Combinatorial Theory, Series B* **67**(1): 34–47.

Greenlaw, R. P. (1995). Cubic graphs, *ACM Computing Surveys* **27**(4): 471–495.

Szekeres, G. (1973). Polyhedral decompositions of cubic graphs, *Bulletin of the Australian Mathematical Society* **8**(3): 367–387.

Vizing, V. (1964). On an estimate of the chromatic class of a p-graph, *Diskretnyj Analiz* **3**: 25–30.

Watkins, J. and Wilson, R. (1988). A survey of snarks, *in* Y. Alavi, G. Chartrand, O.R. Oellermann, and A.J. Schwenk (Eds.), *Graph Theory, Combinatorics, and Applications*, Wiley Interscience, New York, NY/Kalamazoo, MI.

Watkins, J. (1989). Snarks, *Annals of the New York Academy of Sciences* **576**: 606–622.

Piotr Formanowicz received the Ph.D. degree in 2000 and the habilitation qualification in 2005 from the Poznań University of Technology. He is an associate professor at the Institute of Computing Science, Poznań University of Technology, and at the Institute of Bioorganic Chemistry, Polish Academy of Sciences. His research interest include combinatorial optimization, scheduling theory, computational complexity, graph theory and computational biology.

Krzysztof Tanaś received his M.Sc. in the field of computer science in 2007 at Adam Mickiewicz University in Poznań. Now he is a Ph.D student at the Poznań University of Technology, Faculty of Computer Science, Laboratory of Computing Systems and Algorithm Design. His research interests include discrete mathematics, theoretical computer science, graph theory and algorithms. Currently he focuses on the edge coloring of cubic graphs, especially snarks.

Appendix A

Sample results and their visualization

This appendix contains sample results of the described algorithm, which are sample Fan–Raspaud colorings and their visualization in the *Asymptote* environment.

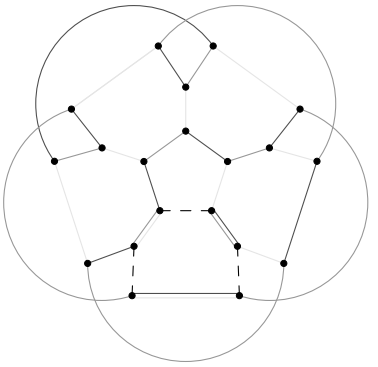

Fig. 8. Fan–Raspaud coloring of the Flower snark 5 with a single 6-cycle disorder.

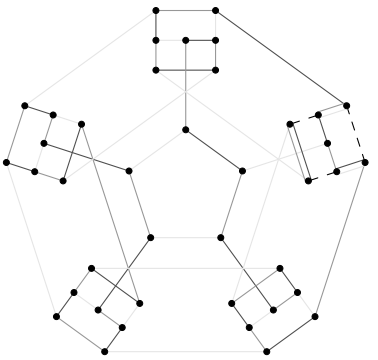

Fig. 9. Fan–Raspaud coloring of the Goldberg snark 5 with a single 6-cycle disorder.

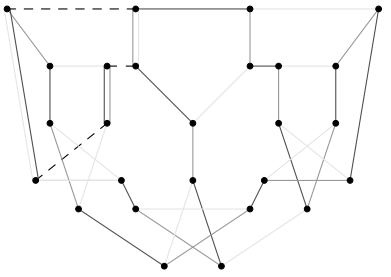

Fig. 10. Fan–Raspaud coloring of the Celmins–Swart snark 1 with a single 6-cycle disorder.

Appendix B

Efficiency tables

This appendix contains results in the field of efficiency for the tested algorithms. The first two tables measure the probability of success (finding a successful Fan–Raspaud coloring) for various starting vertices for the example of the Flower snark 5 for both presented algorithms.

The measure is performed by constructing two tables $Y[0,\ldots,m]$, $N[0,\ldots,m]$, where $Y[i]$ and $N[i]$ is the total number of correct and incorrect leaves in the prefix tree, respectively. The number m is the maximum length of a coded leaf in the tree.

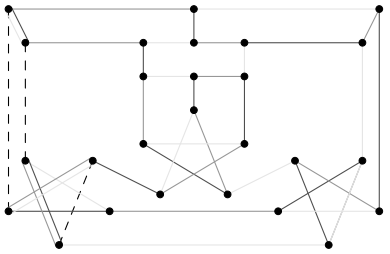

Fig. 11. Fan–Raspaud coloring of the Celmins–Swart snark 2 with a single 6-cycle disorder.

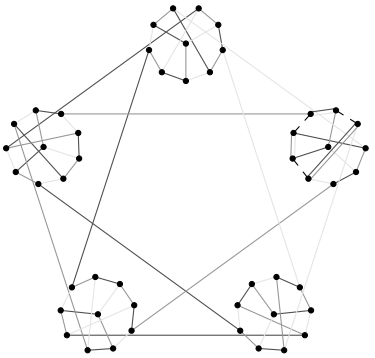

Fig. 12. Fan–Raspaud coloring of the Szekeres snark with a single 6-cycle disorder.

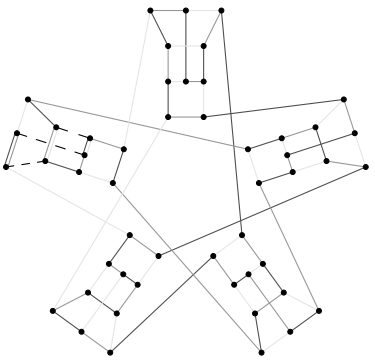

Fig. 13. Fan–Raspaud coloring of the Watkins snark with a single 6-cycle disorder.

The last table shows the effectiveness of the minimum disorder algorithm formed by computing the average size of the Fan–Raspaud disorder for all the successful colorings found by testing the algorithm for every starting vertex for various graphs.

The variables used in the efficiency tables are as follows:

Y[]: table showing numbers of correct leaves for various code lengths.

N[]: table showing numbers of incorrect leaves for various code lengths.

total: total number of leaves in the prefix tree.

Y_total: total number of correct leaves in the prefix tree.

total2: total number of possible random draw results for given code length. $total2$ is equal to $2^{max\_code\_length}$.

Y_total2: total number of random draws leading to successful coloring (= a correct leaf). Y_total2 is equal to $\sum_{i=0}^{m} Y[i] \cdot 2^{m-i}$, where $m = max\_code\_length$.

P: probability of the coloring ending in a success (hitting a correct leaf).

```
Flower snark 5: Efficiency. The basic algorithm

FLOWER_SNARK_J5  starting_vertex=0
max_code_length=14
Y=[0,0,0,0,0,0,0,2,10,44,56,176,194,184,40]
N=[0,0,0,0,0,0,2,2,100,98,18,0,0,0,0]
total=926, Y_total=706
total2=16384,Y_total2=5792  P = 0.353516

FLOWER_SNARK_J5  starting_vertex=1
max_code_length=15
Y=[0,0,0,0,0,0,0,0,0,2,40,70,154,212,100,24]
N=[0,0,0,0,0,0,16,22,60,34,116,34,0,0,0,0]
total=884, Y_total=602
total2=32768,Y_total2=4832  P = 0.147461

FLOWER_SNARK_J5  starting_vertex=2
max_code_length=16
Y=[0,0,0,0,0,0,0,6,6,21,78,160,201,148,58,18,4]
N=[0,0,0,0,0,0,2,0,103,102,30,2,0,0,0,0,0]
total=939, Y_total=700
total2=65536,Y_total2=22080  P = 0.336914

FLOWER_SNARK_J5  starting_vertex=3
max_code_length=16
Y=[0,0,0,0,0,0,0,6,6,21,78,160,201,148,58,18,4]
N=[0,0,0,0,0,0,2,0,103,102,30,2,0,0,0,0,0]
total=939, Y_total=700
total2=65536,Y_total2=22080  P = 0.336914
```

```
Flower snark 5: Efficiency. The md algorithm

FLOWER_SNARK_J5  starting_vertex=0
max_code_length=4
Y=[0,0,0,1,0]
N=[0,0,0,5,4]
total=10, Y_total=1
total2=16,Y_total2=2  P = 0.125000

FLOWER_SNARK_J5  starting_vertex=1
max_code_length=5
Y=[0,0,0,0,0,7]
N=[0,0,0,6,0,1]
total=14, Y_total=7
total2=32,Y_total2=7  P = 0.218750

FLOWER_SNARK_J5  starting_vertex=2
max_code_length=4
Y=[0,0,0,2,2]
N=[0,0,0,2,6]
total=12, Y_total=4
total2=16,Y_total2=6  P = 0.375000

FLOWER_SNARK_J5  starting_vertex=3
max_code_length=4
Y=[0,0,0,2,2]
N=[0,0,0,2,6]
total=12, Y_total=4
total2=16,Y_total2=6  P = 0.375000

FLOWER_SNARK_J5
 Number of found colorings for possible
 F-R disorder sizes:
 0  6  8 10 12 14 16 18
----------------------------------------
 0 65 25  0  0  0  0  0
 Number of all found colorings:90
 Average size of F-R disorder:6.555556

GOLDBERG_SNARK_5
 Number of found colorings for possible
 F-R disorder sizes:
  0  6   8 10 12 14 16 18 20 22
 24 26  28 30 32 34 36 38
----------------------------------------
  0 30 619 10 58  6  1  0  0  0
  0  0   0  0  0  0  0  0
 Number of all found colorings:724
 Average size of F-R disorder:8.325967

FIRST_CELMINS-SWART_SNARK
 Number of found colorings for possible
 F-R disorder sizes:
 0  6 8 10 12 14 16 18 20 22 24
----------------------------------------
 0 34 2  0  1  0  0  0  0  0  0
 Number of all found colorings:37
 Average size of F-R disorder:6.270270

SECOND_CELMINS-SWART_SNARK
 Number of found colorings for possible
 F-R disorder sizes:
 0  6 8 10 12 14 16 18 20 22 24
----------------------------------------
 0 36 4 16  4  0  0  0  0  0  0
 Number of all found colorings:60
 Average size of F-R disorder:7.600000

SZEKERES_SNARK_50
 Number of found colorings for possible
 F-R disorder sizes:
  0  6  8 10 12 14 16 18 20 22 24 26 28
 30 32 34 36 38 40 42 44 46 48
----------------------------------------
 0 684  0  0 66 60 29 73  0  2  0  0  0
 0   0  0  0  0  0  0  0  0  0
 Number of all found colorings:914
 Average size of F-R disorder:8.269147

WATKINS_SNARK_50
 Number of found colorings for possible
 F-R disorder sizes:
  0  6  8 10 12 14 16 18 20 22 24 26 28
 30 32 34 36 38 40 42 44 46 48
----------------------------------------
 0 798  0 26 11 46 20  0  0  0  0  5  0
 0   0  0  0  0  0  0  0  0  0
 Number of all found colorings:906
 Average size of F-R disorder:6.924945
```

Received: 27 March 2011
Revised: 21 September 2011
Re-revised: 16 January 2012