A Self-Stabilizing (Δ+4)-Edge-Coloring Algorithm for Planar Graphs in Anonymous Uniform Systems

Chi-Hung Tzeng
Department of Computer Science
National Tsing-Hua University
Taiwan, ROC

Jehn-Ruey Jiang and Shing-Tsaan Huang

Department of Computer Science and Information Engineering
National Central University
Taiwan, ROC

This research was supported in part by the National Science Council of the Republic of China under the Contract NSC 94-2213-E-008-001.

Correspondence author:
Assoc. Prof. Jehn-Ruey Jiang
Department of Computer Science and Information Engineering
National Central University
Taiwan, ROC
E-mail: jrjiang@csie.ncu.edu.tw

Key Words: distributed systems, edge coloring, planar graphs, self-stabilization.

Abstract

This paper proposes a self-stabilizing edge coloring algorithm using ($\Delta + 4$) colors for distributed systems of a planar graph topology, where $\Delta \geq 5$ is the maximum degree of the graph. The algorithm can be applied to anonymous uniform systems and its time complexity is $O(n^2)$ moves under the central daemon model.

1. Introduction

A *distributed system* can be modeled by a simple connected undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges representing intercommunication links. Its states are divided into two categories: *legitimate states* and *illegitimate states*. Ideally, it should remain in legitimate states to work properly, but unexpected transient faults may bring it into an illegitimate state. Therefore, it needs to deal with such situations by some sort of fault-tolerant mechanisms, one of which is called *self-stabilization* [5]. A system is said to be self-stabilizing if (1) it can converge to a legitimate state regardless of any initial (possibly illegitimate) state, and (2) when it is in a legitimate state, it remains so henceforth.

In this paper, we focus on the *edge-coloring* problem in a self-stabilizing system. We want to assign each edge a color such that adjacent edges (i.e., edges incident to a node) get distinct colors. The system is then said to be in a *legitimate state* if and only if it has a proper edge coloring. Edge coloring has many applications. For example, it can be used to schedule nodes so that each node can communicate with at most one node at a time. This can be achieved by demanding two neighboring nodes to communicate with each other only at a time slot corresponding to the color of the edge connecting them.

Usually, one would want to edge color G with as few colors as possible. The minimum number of colors sufficient to edge color G is called the *edge chromatic index*, denoted by $\chi'(G)$. Let Δ be the maximum degree of G. It is easy to see that $\chi'(G) \geq \Delta$. Another trivial bound is $\chi'(G) \leq 2\Delta - 1$ because each edge has at most $2\Delta - 2$ adjacent edges. In 1964, Vizing proved a remarkable result that $\chi'(G) \leq \Delta + 1$ for simple graphs [14]. By Vizing's theorem, $\chi'(G)$ is either Δ or $\Delta + 1$ for simple graphs. For general graphs, deciding whether $\chi'(G)$ is Δ has been proven to be an NP-complete problem [9]. For bipartite graphs, they have $\chi'(G) = \Delta$, while cycles of odd number of nodes have $\chi'(G) = \Delta + 1$. For planar graphs, there are instances such that $\chi'(G) = \Delta$ or $\Delta + 1$ for $\Delta = 2, 3, 4, 5$. But for the case $\Delta \geq 6$, Vizing proved that $\chi'(G) = \Delta$ if $\Delta \geq 8$; he also made the following conjecture: For a planar graph G, $\chi'(G) = \Delta$ if $\Delta = 6$ or $\Delta = 7$ [15]. This conjecture has been confirmed for the case $\Delta = 7$ in [16] and [13], but it still remains open for the case $\Delta = 6$.

There are many algorithms [2, 3, 4, 6, 8, 11, 12, 14] proposed to solve the edge

coloring problem for different kinds of graphs (such as planar, bipartite, and general graphs, etc.) under different computing models (such as sequential, parallel, and distributed models). In this paper, we propose an edge-coloring algorithm for a planar graph under self-stabilizing computing model. The algorithm can be applied to anonymous uniform systems since it does not rely on node IDs and each node executes the same code. By labeling nodes, it assigns each edge a priority such that each edge has at most $\Delta + 3$ adjacent edges with equal or higher coloring priorities, where $\Delta \geq 5$ is the maximal degree of the graph. When two adjacent edges are of the same color, the lower-priority one is forced to change its color. The coloring hence settles from the highest-priority edges to the lowest-priority ones. Eventually the graph has a proper edge coloring and no edge can change its color henceforth. And the number of colors used is thus at most $\Delta + 4$. As we will show, the time complexity of the algorithm is $O(n^2)$ moves under the central daemon model.

The rest of the paper is organized as follows. Section 2 presents the algorithm. Its correctness proof and time complexity analyses are given in Section 3. And Section 4 concludes this paper.

2. The proposed algorithm

In this section, we propose a self-stabilizing $(\Delta + 4)$-edge-coloring algorithm for a distributed system of a simple planar graph topology $G = (V, E)$, where Δ is the maximum degree of G. We assume that $\Delta \geq 5$; otherwise the condition $\Delta + 4 > 2\Delta - 1$ holds and we can develop a trivial solution using $2\Delta - 1$ colors. Moreover, we assume that the system is uniform; i.e., all nodes are logically equivalent and run the same program. In other words, we assume anonymous networks; i.e., the algorithm does not rely on node IDs.

We express our algorithm by a set of rules in the form: *guard* → *action*, where *guard* is a Boolean formula and *action* is a set of program statements. Each node can read its own and its neighbors' states to evaluate the guards. When the guard of a certain rule of a node is evaluated to be true, the node is said to be *privileged* and can take a move to execute the action part of the rule. The *central daemon model* is assumed. The central daemon always selects one privileged node to take a move at a time. After the selected node finishes the move, the daemon again chooses a privileged node to take another move, and so on. The selection by the central daemon is unpredictable, so the execution sequence can be in any order.

To edge color G, our first step is to assign each edge a coloring priority. We give each node a label such that any node has at most five neighbors of equal or larger labels. An edge's coloring priority is then defined to be the sum of its endpoints' labels. As will be shown later, any edge has at most $\Delta + 3$ adjacent edges of equal or higher priorities. The colors are assigned to edges according to their priorities, so $\Delta + 4$

colors are sufficient to edge color G properly.

Below, we explain how to assign labels for nodes. The idea is the same as those in [7] and [10]. Let $n = |V|$ be the number of nodes in the system. It is well known that $|E| \le 3n - 6$ for simple planar graphs. The inequality implies that G contains at least one node of degree at most 5. This is because if all nodes are of degree at least 6, then we have $|E| \ge 3n$, which is a contradiction. We search for such nodes and let them form the set V_0. We then remove from G the nodes in V_0 as well as the edges incident to them to get a planar subgraph $G - V_0$. In a similar manner, we search for the nodes of degree at most 5 in $G - V_0$ and let them form the set V_1. By repeating this procedure, the node set V is partitioned into non-empty, mutually disjoint sets V_0, V_1,.... We define a node's label to be k if and only if it belongs to V_k. Each node u maintains a variable $L.u$ to denote its label, where $L.u \in \{0, 1\ldots, n\}$.

The above labeling procedure can be realized in a distributed way. Let $N.u$ denote the set of node u's neighbors and let $N^k.u = \{v \mid v \in N.u, k \le L.v\}$ denote the subset of u's neighbors whose labels are equal to or larger than k. Node u always sets $L.u$ to be the minimum k such that $|N^k.u| \le 5$. In this way, any node u in V_0 has $L.u = 0$ because the condition $|N^k.u| \le 5$ holds for any $k \ge 0$. Based on the similar reason, any node u in V_1 has $L.u = 1$,…, and so forth. Hence the label assignment rule R0 for any node u is:

(R0) $L.u \ne \min\{k \mid 0 \le k \le n, |N^k.u| \le 5\} \rightarrow L.u = \min\{k \mid 0 \le k \le n, |N^k.u| \le 5\}$;

In our design, a node needs to know the labels of all neighbors' neighbors, so each node u maintains a variable $L(v).u$ for each of its neighbors v and sets $L(v).u = L.v$ whenever $L(v).u \ne L.v$. We have the following rule R1:

(R1) $\exists v$: $L(v).u \ne L.v \rightarrow L(v).u = L.v$;

With the help of the labels, we define the coloring priority for an edge (u, v) to be $L.u + L.v$. When the labeling settles, this definition makes (u, v) have at most $\Delta + 3$ adjacent edges with equal or higher priorities for the following reasoning. We first focus on the case $L.u \ne L.v$. Without loss of generality, we further assume that $L.u < L.v$. For node u, it has at most 4 neighbors in $N.u - \{v\}$ having equal or larger labels than $L.u$, so at most 4 edges incident to u have equal or higher coloring priorities than that of (u, v). For node v, it has at most $\Delta - 1$ neighbors in $N.v - \{u\}$ with equal or larger labels than $L.u$, so at most $\Delta - 1$ edges incident to v have equal or higher coloring priority than that of (u, v). On the whole, (u, v) has at most $4 + (\Delta-1) = \Delta + 3$ adjacent edges with equal or higher coloring priorities than itself. Now, focus on the

case $L.u = L.v$. Based on the similar reasoning shown above, at most 4 edges incident to u and at most 4 edges incident to v have equal or higher coloring priorities than that of (u, v). Hence (u, v) has at most 8 (or $\Delta + 3$; this is because $\Delta + 3 \geq 8$ under the assumption of $\Delta \geq 5$) adjacent edges of the same or higher priorities than itself.

Now we begin discussing the coloring. To express an edge (u, v)'s color in a distributed way, node u (resp., node v) maintains a variable $C(v).u$ (resp., $C(u).v$) to be the color associated with this edge. Since each edge has two color variables associated with it, we let the endpoint with the smaller label decide the edge's color and let the other endpoint copy that color. That is, for an edge (u, v) such that $L.u < L.v$, node u selects a proper color for $C(v).u$, and node v just sets $C(u).v = C(v).u$. For the case of $L.u = L.v$, if node u is the first one chosen by the central daemon to take a move, it selects a proper color for $C(v).u$ and then v just copies this color for $C(u).v$, and vice versa. It is noted that the coloring priority of an edge has nothing to do with the order of the endpoints deciding the edge color.

For an edge (u, v) such that $L.u \leq L.v$, let $Z(u^*, v) = \{z \mid z \in N.u,\ L.u \leq L(z).u\} - \{v\}$ be the set of u's neighbors, except v, whose labels are larger than or equal to u's label. On the other hand, let $Z(u, v^*) = \{z \mid z \in N.v,\ L.u \leq L(z).v\} - \{u\}$ be the set of v's neighbors, except u, whose labels are larger than or equal to u's label. We also let $Used(u, v) = \{C(z).u \mid z \in Z(u^*, v)\} \cup \{C(z).v \mid z \in Z(u, v^*)\}$ be a set of colors. It contains the colors used by (u, v)'s adjacent edges that have equal or higher priorities than (u, v). By the definition, we have $|Used(u, v)| \leq |Z(u^*, v)| + |Z(u, v^*)| \leq 4 + (\Delta - 1) = \Delta + 3$. If we set $C(v).u$ to be an element not in $Used(u, v)$, we then get a proper edge coloring. Below, we further define two functions $Incorrect(C(v).u)$ and $Decide(C(v).u)$ to help node u decide $C(v).u$. For the sake of presentation, we use $Correct(C(v).u)$ to denote the complement of $Incorrect(C(v).u)$.

$$Incorrect(k) \equiv \begin{cases} true, & if\ k \in Used(u, v) \\ false, & otherwise \end{cases}$$

$Decide(C(v).u) \equiv$ to set $C(v).u = \min(\{0,1,\ldots, \Delta+3\} - Used(u, v))$;

With these functions, we have the rules R2 to R4 for a node u to edge color the graph. By the rules, an edge cannot choose the color used by its adjacent edges with equal or higher priorities. On the other hand, an edge may or may not choose the color used by a lower-priority adjacent edge. This is because the set $Used(u, v)$ may contain the colors used by lower-priority adjacent edges when node u decides the color for (u, v). However, since $|Used(u, v)| \leq \Delta + 3$, there is always a color for node u to choose to

edge color the graph properly.

(R2) $\exists v$: $L.u < L.v \wedge Incorrect(C(v).u) \rightarrow Decide(C(v).u)$;

(R3) $\exists v$: $L.u > L.v \wedge C(v).u \neq C(u).v \wedge (Correct(C(u).v)$ by $v) \rightarrow C(v).u = C(u).v$;

(R4) $\exists v$: $L.u = L.v \wedge Incorrect(C(v).u) \wedge Incorrect(C(u).v) \rightarrow Decide(C(v).u)$;

(R5) $\exists v$: $L.u = L.v \wedge C(v).u \neq C(u).v \wedge Correct(C(u).v) \rightarrow C(v).u = C(u).v$;

It is noted that in rule R3, "$Correct(C(u).v)$ by v" stands for the result of $Correct(C(u).v)$ evaluated by node v. Since $Used(u, v)$ may be different from $Used(v, u)$ in the case of $L.u \neq L.v$, nodes u and v may get different results of $Correct(C(u).v)$. By reading the variables maintained by both u and v, node u can determine the result of $Correct(C(u).v)$ evaluated by v, so it can evaluate the predicate of R3 correctly.

3. Correctness and Analysis

According to the definition of edge coloring, the conditions $C(v).u = C(u).v$ and $C(v).u \neq C(v).w$ should hold in a legitimate state for any adjacent edges (u, v) and (v, w). Below, we verify that in $O(n^2)$ moves, the system reaches a legitimate state from any initial state. We first show that in $O(n^2)$ moves the labeling stabilizes; i.e., no node can execute R0 and R1. Afterward, we show that in $O(n^2)$ moves the system is in a legitimate state.

To show that in $O(n^2)$ moves no node can execute R0, we introduce the terms: *decreasing move* and *increasing move* [1]. An execution of R0 is said to be a decreasing (resp., increasing) move if and only if it makes L decrease (resp., increase). We say that a move m_1 triggers another move m_2 if and only if the node to take m_2 is privileged only after m_1 moves. Below, we show that a node's decreasing move cannot trigger increasing moves for its neighbors.

Lemma 1. For an edge (u, v), let m_1 be u's move of R0 and m_2 be v's move of R0 such that m_1 triggers m_2. If m_1 is a decreasing move, then m_2 is not an increasing move.

Proof: To ensure that v gets a privilege of R0 by u's move, we assume that v cannot take the move m_2 before u takes the decreasing move m_1. According to R0, this assumption implies that $|N^{L.v}.v| \leq 5$. And after u takes the decreasing move, the condition $|N^{L.v}.v| \leq 5$ still holds. Because the necessary condition for v to increase $L.v$ is $|N^{L.v}.v| > 5$, m_2 cannot be an increasing move. □

Lemma 2. The labeling stabilizes in $O(n^2)$ moves of R0 and R1.

Proof: To prove this lemma, we show that no node can execute R0 in $O(n^2)$ moves from the start and no node can execute R1 in $O(n^2)$ moves. We first show that each node takes at most n increasing moves and then at most n decreasing moves. By lemma 1, a node's increasing move (of R0) can only be triggered by its neighbors' increasing move (of R0). Since $0 \leq L \leq n$, a node can take at most n increasing moves.

After no node can execute an increasing move, a node executes at most n decreasing moves for the same reason that $0 \le L \le n$. To sum up, each node takes $O(n)$ moves of R0. Hence no node can execute R0 after $n*O(n) = O(n^2)$ moves.

Now, let's focus on R1. Since any node u executes R1 as long as $L(v).u \ne L.v$ and each a node v changes its label $L.v$ at most $O(n)$ times, the total number of moves for R1 is $O(|E|)*O(n) = O(n^2)$, for G is a planar graph. That is, no node can execute R0 and R1 after $O(n^2) + O(n^2) = O(n^2)$ moves. □

Below, we define a *bounded function F* which maps a system state to an integer value:

$$F = \sum_{u \in V} \sum_{v \in N.u} f(u,v),$$

where $f(u,v) = \begin{cases} 3^{L.u+L.v}, & \text{if } L.u \le L.v \wedge (C(v).u \ne C(u).v \vee Incorrect(C(v).u)) \\ 0, & \text{otherwise} \end{cases}$

It is noted that the predicate $Incorrect(C(v).u)$ in $f(u, v)$ is evaluated by node u. The function F maps a system state to an integer value. Below, we first justify that the edge coloring is proper if and only if $F = 0$. We then show that F eventually decreases to 0.

Lemma 3. The edge coloring is proper if and only if $F = 0$.

Proof: (Necessity) In a proper edge coloring, $C(u).v = C(v).u$ and $C(v).u \ne C(v).w$ for any pair of adjacent edges (u, v) and (v, w). It is easy to check $f(u, v) = f(v, u) = 0$ for such a condition. Hence $F = 0$.

(Sufficiency) When $F = 0$, the condition $C(u).v = C(v).u$ obviously holds. Hence we use contradiction to prove $C(v).u \ne C(v).w$ to complete this lemma. Suppose that $C(v).u = C(v).w$ holds for some pair of adjacent edges (u, v) and (v, w). Without loss of generality, we assume $L.u \le L.w$. Because $L.u \le L.w = L(w).v$, we have $w \in Z(u, v^*)$ and $C(w).v \in \{C(z).v \mid z \in Z(u, v^*)\} \subseteq Used(u, v)$. Since $C(v).u = C(v).w = C(w).v \in Used(u, v)$, node u evaluates $Incorrect(C(v).u)$ true. It implies that $f(u, v) > 0$ if $L.u \le L.v$, or that $f(v, u) > 0$ if $L.u \ge L.v$. For either case, we have $F > 0$, which is a contradiction. □

Lemma 4. Suppose that the labeling has stabilized. Let (u, v) be an edge such that $L.u \le L.v$. After node u executes R2 or R4 to change $C(v).u$, F increases by at most $3^{L.u+L.x}$, where $x \in N.u$ and $L.x < L.u$.

Proof: Since the labeling has stabilized, nodes' labels are fixed and each edge (x, y) has only two possible values for $f(x, y)$, either $3^{L.x+L.y}$ or 0. Thus, if F increases after

node u executes R2 or R4, then we have that $f(u, x)$ or $f(x, u)$ alters to be $3^{L.u+L.x}$ from 0 for some u's neighboring node x. Below, we show that there is at most one such node x to increase F and $L.x < L.u$. The condition $L.x < L.u$ makes $f(u, x) = 0$ permanently, and thus we have F increases by at most $3^{L.u+L.x}$.

First, we use contradiction to prove that there is at most one node to increase F. Assume that there is more than one node to increase F. We pick up any two such nodes and denote them by x_0 and x_1. Without lose of generality, we assume $L.x_0 \le L.x_1$. Because $f(x_i, u) = f(u, x_i) = 0$ before $C(v).u$ is altered, node x_i has $C(x_i).u = C(u).x_i$ and $Incorrect(C(u).x_i)$ = false, where $i = 0$ or 1. Altering $C(v).u$ does not change $C(x_i).u = C(u).x_i$. Thus, the only way to increase $f(x_i, u)$ (or $f(u, x_i)$) is by altering $Incorrect(C(u).x_i)$ (or $Incorrect(C(x_i).u)$) to be true from false; it means that the new $C(v).u$ is same as $C(u).x_i$ (or $C(x_i).u$). We thus have $C(u).x_0 = C(u).x_1$. Since $x_1 \in Z(x_0, u^*)$, we have $C(u).x_0 = C(u).x_1 = C(x_1).u \in \{C(z).u \mid z \in Z(x_0, u^*)\} \subseteq Used(x_0, u^*)$. That is, $Incorrect(C(u).x_0)$ = true (or $Incorrect(C(x_0).u)$ = true) before node u changes its color. It is a contradiction.

Next, we use contradiction to prove $L.x < L.u$ for a node x to increase F. Assume $L.x \ge L.u$. We have $L(x).u = L.x \ge L.u$, so $x \in Z(u^*, v)$ and $C(x).u \in \{C(z).u \mid z \in Z(u^*, v)\}$. Because node u executes R2 or R4 to assign $C(v).u$ a value not in the set $Used(u, v)$, the new $C(v).u$ must be different from $C(x).u$. Since $f(u, x) = f(x, u) = 0$ before node u executes R2 or R4, we have $C(x).u = C(u).x$. Hence, $C(v).u \ne C(u).x$ and $Incorrect(C(u).x)$=false after the change of $C(v).u$. To sum up, $C(x).u = C(u).x$ and $Incorrect(C(u).x)$ = false before and after the change of $C(v).u$; i.e., $f(u, x)$ and $f(x, u)$ remains the same and F does not increase. Contradiction occurs. □

Lemma 5. Suppose that the labeling has stabilized. Let (u, v) be an edge such that $L.u \le L.v$. After node v executes R3 or R5, F decreases by at least $3^{L.u+L.v}$ and increases by at most $3^{L.v+L.y}$, where $y \in N.v$ and $L.y < L.u$.

Proof: Let's consider the decreasing of F first. After node v executes R3 or R5, the conditions $C(v).u = C(u).v$ and $Correct(C(v).u)$ = true for node u hold, or equivalently, $f(u, v) = f(v, u) = 0$. Therefore, F decreases by $3^{L.u+L.v}$ if $L.u < L.v$ or decreases by $2\times3^{L.u+L.v}$ if $L.u = L.v$.

Now, let's consider the increasing of F. We first want to show that if F increases, then (1) there is exactly one node y, $y \in N.v$ such that $f(y, v)$ or $f(v, y)$ increases and (2) $L.y < L.u$. The proof of (1) is similar to that in lemma 4, so we skip it. We next use contradiction to prove $L.y < L.u$. Suppose $L.y \ge L.u$. For $L(y).v = L.y \ge L.u$, we have $y \in Z(u, v^*)$ and thus $C(y).v \in Used(u, v)$. Because $Correct(C(v).u)$ = true for node u, $C(v).u \notin Used(u, v)$ and we have $C(v).u \ne C(y).v = C(v).y$. After node v executes R3 or R5 to set $C(u).v = C(v).u$, the condition $C(u).v \ne C(v).y$ holds and hence

Incorrect($C(v).y$) = false. That is, $f(y, v)$ and $f(v, y)$ remain intact and F does not increase; it is a contradiction. □

Lemma 6. Suppose that the labeling has stabilized. Let (u, v) be an edge such that $L.u \leq L.v$. After node u executes R2 or R4 and node v executes R3 or R5, F decreases by at least $3^{L.u+L.v-1}$.

Proof: According to lemmas 4 and 5, F decreases by at least $3^{L.u+Lv}$ and increases by at most $3^{L.u+L.x} + 3^{L.v+L.y}$, where $L.x, L.y < L.u \leq L.v$. Therefore, the overall decrease is at least

$$\begin{aligned} & 3^{L.u+L.v} - 3^{L.u+L.x} - 3^{L.v+L.y} \\ &= 3^{L.u+L.v}\left(1 - \frac{1}{3^{L.v-L.x}} - \frac{1}{3^{L.u-L.y}}\right) \\ &\geq 3^{L.u+L.v}\left(1 - \frac{1}{3} - \frac{1}{3}\right) \\ &= 3^{L.u+L.v-1} \end{aligned}$$

□

Theorem 1. The system eventually enters a legitimate state.

Proof: It is the direct consequence of lemmas 2, 3, 4, 5, and 6. □

Theorem 2. From any initial state, the system enters a legitimate state in $O(n^2)$ moves.

Proof: Let $T(G)$ be the number of moves for a system of topology $G = (V, E)$ to enter a legitimate state since the labeling has stabilized. According to lemma 2, the labeling stabilizes in $O(n^2)$ moves from the start, so it suffices to prove this theorem by deducing $T(G) = O(n^2)$.

Let (u, v) be an edge with the highest coloring priority. Each time (u, v) changes its color, in the worst case, all the other edges with lower priorities in G are forced to change their colors. On the other hand, (u, v) is never forced to change its color. Therefore, $T(G)$ can be defined by a recursive formula: $T(G) = T(G - (u, v)) + O(|E|)$, where $T(G) = 0$ if $G = (V, \phi)$. We have $T(G) = O(|E|^2) = O(n^2)$ since G is planar. □

Theorem 3. When the system is in a legitimate state, it remains so henceforth.

Proof: It is easy to check that no node can execute a rule to change the node labels or the edge colors when the system is in a legitimate state. Therefore, the theorem holds. □

4. Conclusion

In this paper, we have proposed a self-stabilizing edge coloring algorithm for distributed systems of a planar graph topology. It utilizes the concept of node labeling to assign each edge a coloring priority. It can edge color the graph properly with $\Delta + 4$

colors, where $\Delta \geq 5$ is the maximum degree of the graph. It is applicable to anonymous uniform systems since it does not rely on node IDs and each node executes the same code. The time complexity of the algorithm is $O(n^2)$ moves under the central daemon model.

References

[1] S. C. Bruell, S. Ghosh, M. H. Karaata, and S. V. Pemmaraju, Self-Stabilizing algorithms for finding centers and medians of trees, SIAM Journal of Computing, Vol. 29, No. 2, pp 600-614, 1999.

[2] J. F. Boyar and H. J. Karloff, Coloring planar graphs in parallel, Journal of Algorithms, Vol. 8, pp. 470-479, 1987.

[3] M. Chrobak and T. Nishizeki, Improved Edge-Coloring Algorithms for Planar Graphs, Journal of Algorithms, Vol. 11, pp. 102-116, 1990.

[4] R. Cole, K. Ost, and S. Schirra, Edge coloring bipartite multigraphs in $O(E \log D)$ time, Combinatorica, Vol 21, No. 1, pp. 5-12, 2001.

[5] E. W. Dijkstra, Self-stabilizing systems in spite of distributed control, Communications of the ACM, Vol. 17, pp. 643-644, 1974.

[6] H. N. Gabow and O. Kariv, Algorithms for edge coloring bipartite graphs and multigraphs, SIAM journal on Computing, Vol. 11, No. 1, pp. 117-129, 1982.

[7] S. Ghosh and M. H. Karaata, A self-stabilizing algorithm for coloring planar graph, Distributed Computing, Vol. 7, pp. 55-59, 1993.

[8] D. A. Grable and A. Panconesi, Nearly optimal distributed edge coloring in O(loglog n) rounds, RSA, Vol. 10, No. 3, pp 385-405, 1997.

[9] I. Holyer, The NP-Completeness of edge coloring, SIAM Journal of Computing, Vol. 10, No. 4, pp. 718-720, 1981.

[10] S. T. Huang, S. S. Hung, and C. H. Tzeng, Self-stabilizing coloration in anonymous planar networks, Information Processing Letters, Vol. 95, No. 1, pp. 307-312, 2005.

[11] H. J. Karloff and D. B. Shmoys, Efficient parallel algorithms for edge coloring problems, Journal of Algorithms, Vol. 8, pp. 39-52, 1987.

[12] M. V. Marathe, A. Panconesi, and L. D. Risinger, An Experimental Study of a Simple, Distributed Edge-Coloring Algorithm, ACM Journal of Experimental Algorithms, Vol. 9. pp. 1-22, 2004.

[13] D. Sanders, Y. Zhao, Planar graphs of maximum degree seven are class I, Journal of Combinatorial Theory, Series B, Vol. 8, No. 2, pp. 201-212, 2001.

[14] V. G. Vizing, On an estimate of the chromatic class of a p-graph, Diskret. Analiz, Vol. 3, pp. 25-30, 1964. (in Russian)

[15] V. G. Vizing, Critical graphs with given chromatic class, Metody Diskret. Analiz. Vol. 5, pp. 9-17, 1965. (in Russian)

[16] L. Zhang, Every planar graph with maximum degree 7 is of class 1, Graphs and

Combinatorics, Vol. 16, No. 4, pp. 467-495, 2000.