Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/journals/array/2590-0056/open-access-journal

A distributed algorithm for vertex coloring problems in wireless networks

Mohammadhasan Miri [a,*], Yousef Darmani [b], Kamal Mohamedpour [b], R. Lal Tummala [c], Mahasweta Sarkar [c]

[a] Department of ECE, Golestan University, Gorgan, Iran
[b] Department of ECE, K. N. Toosi University of Technology, Tehran, Iran
[c] Department of ECE, San Diego State University, SanDiego, USA

ARTICLE INFO

Index Terms:
Bit complexity
Distributed algorithms
Media access control
Time complexity
Vertex coloring problems
Wireless networks

ABSTRACT

A lot of problems in computer networks are modeled by the vertex coloring problem (VCP), the multicoloring problem (MCP), the bandwidth coloring problem (BCP), and the bandwidth multicoloring problem (BMCP). To solve the VCP with $\Delta + 1$ colors, a myriad distributed algorithms have been proposed to reduce time complexity (the number of rounds), where Δ is the maximum vertex degree in the graph. Time and communication complexities of these algorithms are functions of n and Δ, where n is the number of vertices in the graph. The MCP can be converted into VCP after transformation of the graph; the transformation increases the time and communication complexities. However, no distributed algorithms for BCP and BMCP have been proposed in the literature. In this paper, we propose a General Distributed Vertex Coloring Algorithm (GDVCA) to solve four problems in wireless networks. In GDVCA, some (and not necessarily all) vertices assign colors to themselves and their neighbors, which we refer to them as assigning vertices. The communication cost of GDVCA is very low; the number of messages transmitted by each assigning vertex is one and by the other vertices is zero. Assigning vertices assign the first available colors to uncolored vertices; moreover assigning vertices can use heuristic methods to choose the next proper vertex for coloring. Therefore, the number of colors used by the distributed algorithm GDVCA is significantly low. The number of time slots required to color a graph is $O(\Delta^2)$ and at most n, where each round comprises several time slots.

1. Introduction

The topology of a distributed system can be modeled by a graph and paradigms of distributed systems are represented by classical problems in graph theory. Vertex coloring, computing a maximal independent set, finding a vertex cover, and finding a maximal matching are some classical problems that can be used to model numerous problems. Each solution to one of these problems is a building block for a lot of distributed algorithms: resource allocation, network synchronization, routing, topology control, or symmetry breaking [1,2]. Vertex coloring and its generalizations have been used a lot in computer and telecommunication engineering, for example to solve computer register allocation, channel assignment in cellular networks, and scheduling problems [3–7].

A legal $(K+1)$-coloring of a graph assigns colors $\{0, 1, 2, \ldots, K\}$ to the vertices such that adjacent vertices do not receive the same color [7]. In $(K+1)$-coloring, main metrics are the number of colors $(K+1)$, time, bit, and communication complexities. In the Vertex Coloring Problem (VCP), the objective is finding the minimum number of colors, which is called chromatic number χ [8]. It is an NP-hard problem [3,5,9]. According to Brook's theorem, graph G can be colored with $\Delta + 1$ colors, where Δ is the maximum vertex degree of G [10]. $\Delta + 1$ is a loose upper bound for χ and χ can not be approximated to any closer bound [11]. Tables 1 and 2 show the main notations and abbreviations used in the paper, respectively.

The greedy coloring algorithm chooses the next vertex and assigns it the first (minimum) available color [9]; the first available color for each vertex $v \in V$ is the minimum non-negative color that has not been assigned to v's neighbors, where V includes all vertices of the graph and $n = |V|$. The greedy coloring algorithm can use a heuristic method to choose the next vertex. Using heuristic methods increases the efficiency (using fewer colors). First fit, largest degree ordering, saturation degree ordering, incidence degree ordering, and their combinations are heuristic methods that have been proposed [4,12].

The MultiColoring Problem (MCP), the Bandwidth Coloring Problem

* Corresponding author.
E-mail addresses: mh.miri@gu.ac.ir (M. Miri), darmani@eetd.kntu.ac.ir (Y. Darmani), kmpour@eetd.kntu.ac.ir (K. Mohamedpour), rtummala@sdsu.edu (R.L. Tummala), msarkar2@sdsu.edu (M. Sarkar).

https://doi.org/10.1016/j.array.2020.100023
Received 3 October 2019; Received in revised form 23 December 2019; Accepted 5 March 2020
Available online 12 March 2020

Table 1
Main notations.

u	vertex
v	vertex
G	graph
V	set of G vertices
E	set of G edges
$T(G)$	transformed graph of G
V'	set of $T(G)$ vertices
E'	set of $T(G)$ edges
$V(v)$	set includes v and its adjacent vertices
$G(v)$	graph induced by $V(v)$ from G
$E(v)$	set of $G(v)$ edges
$w(v)$	weight of v
$w(u,v)$	weight of edge (u,v)
$d(v)$	degree of v
Δ	maximum vertex degree in G
Δ'	maximum vertex degree in $T(G)$
n	number of G vertices
n'	number of $T(G)$ vertices
χ	chromatic number of G
UCV	set of uncolored vertices
$c(v)$	set of v's colors
$C(V(v))$	colors assigned to $V(v)$ vertices
H	length of message overhead in bit

Table 2
Main abbreviations.

BS	Base Station
VCP	Vertex Coloring Problem
MCP	MultiColoring Problem
BCP	Bandwidth Coloring Problem
BMCP	Bandwidth MultiColoring Problem
DVCA	Distributed Vertex Coloring Algorithm
GDVCA	General Distributed Vertex Coloring Algorithm
CMSG	Coloring MeSsaGe
NH	Neighbors Head
CSMA	Carrier Sense Multiple Access

(BCP), and the Bandwidth MultiColoring Problem (BMCP) are generalizations of the VCP. They model more complex situations than VCP [13]. MCP can be used to schedule jobs with different time requirements [9, 14]. BCP and BMCP are notable for their applications in the area of frequency assignment in mobile networks [15–17]. In MCP and BMCP problems, the number of colors required by a vertex may be more than one; in BCP and BMCP problems, the absolute difference between colors assigned to two adjacent vertices may be required to be greater than one.

In MCP, BCP, and BMCP, the objective is to find the minimum number of colors. MCP, BCP, and BMCP are NP-hard because they generalize the VCP [13,14,17,18]. Various metaheuristic algorithms such as tabu search algorithms [9,17–19], evolutionary approach [13,14,20], constraint programming approaches [16], local search, simulated annealing, and hybrid algorithms have been proposed to solve these problems.

The greedy, heuristic, or metaheuristic methods presented for the four coloring problems are centralized algorithms. Centralized vertex coloring algorithms may not be applicable in a distributed system, because one node (vertex) must have knowledge of the system (graph). It requires determining that vertex and all other vertices give it their information. The coloring is done just by the vertex; coloring is done sequentially and not in parallel. Finally, the colors assigned by the vertex must be informed to all other vertices. Therefore, using centralized coloring requires lots of message exchanges (send and receive). Thus, it may not be feasible in distributed wireless systems.

A kind of Distributed Vertex Coloring Algorithm (DVCA) for VCP, in the distributed LOCAL model, has been proposed and studied for four decades. MCP can be converted into VCP after transformation of the graph [17,19–21]bib21. However, as it will be described later, the transformation introduces extra vertices and edges. Therefore, using DVCA for the MCP may not be feasible. To the best of our knowledge, no distributed algorithms have been proposed for BCP and BMCP.

Table 3
An overview on time complexities of DVCA.

Time complexity	Number of colors
$2^{O(\sqrt{\log n})}\log n$[39] DET	$\Delta+1$
$O(\sqrt{\Delta\log\Delta}\log^*\Delta+\log^* n)$[6] DET	$\Delta+1$
$O(\sqrt{\Delta}\log^{2.5}\Delta+\log^* n)$[28] DET	$\Delta+1$
$O(\Delta^{3/4}\log\Delta+\log^* n)$[27] DET	$\Delta+1$
$O(\Delta^2+\log^* n)$[26] DET	$\Delta+1$
$O(\Delta\log n)$[26] DET	$\Delta+1$
$O(\sqrt{\log\Delta})+2^{O(\sqrt{\log\log n})}$[7]	$\Delta+1$
$O(\log\Delta)+2^{O(\sqrt{\log\log n})}$[23]	$\Delta+1$
$O(\log\Delta+\sqrt{\log n})$[30]	$\Delta+1$
$O(\log n)$[41]	$\Delta+1$
$2^{O(\sqrt{\log\log n})}$[23]	$O(\Delta)$
$O(\log\log n)$[30]	$O(\Delta+\log n)$
$O(\log^* n)$[38]	$O(\Delta^2)$
$\min\left\{O\left((\log\Delta)\frac{13}{12}\log n\right), 2^{O(\log\Delta+\sqrt{\log\log n})}\right\}$[10]	$\Delta-\sqrt{\Delta}+3$

In the standard LOCAL model, time is divided into synchronous rounds. In each round, every vertex of the graph sends a single message to each of its neighbors, receives the messages sent to it by its neighbors, and performs an (unbounded) amount of local computations [7,8]. The time complexity of an algorithm is measured by the total number of rounds [6,7,22]. As it can be seen in Table 3, time complexities of proposed algorithms are functions of Δ and n; the number of colors is $\Delta+1$, $O(\Delta)$, or $O(\Delta^2)$; and there is a tradeoff between the number of colors and time complexity [23]. In many problems modeled by the VCP, the efficiency is inversely proportional to the number of colors. Therefore, our focus is on $(\Delta+1)$-coloring; it is also the most common in related work [6–8,10,24].

Proposed distributed algorithms (the DVCA [1–4,6,7,11,15,25–31] or other algorithms [24,32–34]) use at least $\Delta+1$ colors. Besides, the number of messages is high because of need for synchronization, knowledge of Δ and n, and contention for colors; the number of messages transmitted by each vertex only for coloring is a function of the number of vertices. Therefore, it is necessary to propose a new kind of distributed vertex coloring algorithm that requires fewer colors and message exchanges. In this paper, we present a General Distributed Vertex Coloring Algorithm (GDVCA) to solve the four problems. The main advantages of GDVCA over DVCA, the most common algorithms for VCP [24], are the following:

- The initial overhead of GDVCA is less than DVCA. It does not require synchronization nor the knowledge of Δ and n. They may require time and message exchange more than coloring itself;
- In DVCA algorithms, each vertex v randomly chooses its color from set $\{0,1,\dots,d(v)\}$ or $\{0,1,\dots,\Delta\}$, where $d(v)$ is v's degree. But in GDVCA, some (and not necessarily all) vertices assign colors to themselves and their neighbors, which we refer to them as assigning vertices. Each assigning vertex assigns the first available color to itself and its neighbors. Assigning vertices can also use heuristic methods. Therefore, GDVCA reduces the number of colors and increases the efficiency;
- While DVCA cannot be extended to solve BCP or BMCP, and there are no other distributed algorithms for these problems, GDVCA solves these problems in a distributed manner without increasing time and communication complexities. In other words, time and communication complexities of GDVCA are independent of the number and weights of vertices and edges' weights;
- Communication costs usually outweigh the costs for (local) computation [30]. Message exchange includes composing a frame, listen to the channel before transmission, transmission, listen to the channel and receive a frame, and decomposing a frame. It usually consumes time and energy more than computations [35,36]. In GDVCA, the

number of messages transmitted by each vertex is at most one (compared to $O(n,\Delta)$ for VCP, where $O(n,\Delta)$ is a general form for time complexity or communication complexity of DVCA, please refer to Table 3);
- Time complexity of GDVCA is a function of only Δ and not n, unlike DVCA (please refer to Table 3). Therefore, when the maximum number of neighbors is finite and the number of vertices approaches infinity, which is very common in computer networks, coloring using GDVCA takes a finite time.

In summery, the efficiency of GDVCA is high. Moreover, the communication cost and thereby power consumption of GDVCA is very low, which is among the most important objective in wireless networks [36]. The rest of this paper is organized as follows. The next section reviews related works on vertex coloring algorithms. In Section 3, the four coloring problems are defined and formulated. GDVCA algorithm is described in Section 4. Section 5 shows the efficacy of GDVCA over 33 benchmark instances for the four vertex coloring problems. The paper concludes with Section 6.

2. Related works

The distributed $(\Delta + 1)$-coloring problem is among the most important studied problems in distributed computing since 1980's [7,23,37]. Most of the proposed algorithms are based on LOCAL model. In LOCAL model, all vertices start executing an algorithm simultaneously; coloring is composed of some rounds. In each round, each uncolored vertex chooses a color and informs its neighbors using a message, receives messages from its neighbors, and performs (unbounded) local computations [30]. Then, at the end of each round, the vertex takes the color if its neighbors have not chosen that color; otherwise there is a competition and the vertex may lose the color. Since communication costs usually outweigh the costs for (local) computation, the time complexity of an algorithm is measured in the number of rounds until every vertex commits its color.

Table 3 summarizes the time complexities of several proposed algorithms for this problem. They are deterministic or randomized. Deterministic algorithms have no probability of failure (staying uncolored); they have been marked as DET in the table. For the deterministic approach, several algorithms with running time of $O(f(\Delta)+\log^{*}n)$ have been developed [7]. The latter term is necessary as [38] showed that 3-coloring a ring requires $\Omega(\log n)$ rounds. Goldberg et al. presented two $(\Delta + 1)$-coloring algorithms with running times $O(\Delta^{2}+\log^{*}n)$ and $O(\Delta\log n)$ [26]. Barenboim first gave an algorithm running in $O(\Delta^{3/4}log\Delta+\log^{*}n)$ rounds [27]. Then, the bound was improved to $O(\sqrt{\Delta}\log^{2.5}\Delta+\log^{*}n)$ by Fraigniaud et al. [28]. After that, Barenboim et al. obtained a $(\Delta + 1)$-coloring within $O(\sqrt{\Delta\log\Delta}\log^{*}\Delta+\log^{*}n)$ time [6]. Recently, Kuhn has presented an algorithm that runs in $2^{O(\sqrt{\log n})}\log n$ [39].

In randomized coloring a graph is colored with high probability $\left(1-\frac{1}{n^{c}}\right)$, where $c>0$ is an arbitrary constant. It is seen that randomized algorithms are exponentially faster (in terms of n or Δ) than their deterministic counterparts; moreover, they are usually simpler to analyze and implement [40]. Therefore, our focus will be on randomized coloring algorithms. The randomized approach can be traced back to the $O(\log n)$ rounds [29]. The $O(\log n)$ upper bound improved when Schneider and Wattenhofer gave an algorithm of running time $O(\log\Delta+\sqrt{\log n})$ [30]. Then, Barenboim et al. improved the dependence on n to $2^{O(\sqrt{\log\log n})}$ by a graph shattering technique [31]. [7] gives an algorithm that runs in $O(\sqrt{\log\Delta})+2^{O(\sqrt{\log\log n})}$ time.

[1] decreases the bit complexity of randomized distributed algorithms from $O(log^{2}(n))$ to $O(log(n))$. But the number of colors used by the algorithm may be more than $\Delta+1$. [2] has proposed an algorithm that can be used along with the proposed algorithm by Ref. [1] to limit the number of colors to $\Delta+1$. The bit complexity of the proposed algorithm is $O(log(\Delta)\log(n))$.

[32,33], and [34] state that those fast algorithms, mentioned above, are not applicable in an unstructured radio network model, as they presume an established and powerful communication framework. They present randomized coloring algorithms for the unstructured radio network model, a model comprising autonomous nodes, asynchronous wake-up, and an unknown but geometric network topology. The algorithm presented by Ref. [32] requires $O(\Delta\log n)$ time and uses $O(\Delta)$ colors in a unit disk graph; the algorithm requires a linear bound on n and Δ. In a unit disk graph, vertices are points in the plane; two vertices are neighbors if and only if their Euclidean distance is at most 1. [33,34] generalize the network topology from unit disk graph to bounded-independence graph; bounded-independence graphs restrict for any vertex the maximum size of a set of independent vertices in its neighborhood. [34] also improves the results of [32,33]. It reduces the time complexity; its time complexity is $O(\Delta+\log\Delta\log n)$ given an estimate of n and Δ, and $O(\Delta+\log^{2}n)$ without any knowledge of Δ. Moreover, its proposed algorithm requires only $\Delta+1$ colors.

Parter proposed a randomized $(\Delta+1)$ vertex coloring algorithm that works in $O(\log(\log\Delta)\log^{*}\Delta)$-rounds in the congested clique model. In the congested clique model, each node can send $O(\log n)$ bits of information to any node in the network [24].

In summary, the time complexity of a randomized DVCA is $O(n,\Delta)$ and:

- the number of messages transmitted by each vertex is $O(n,\Delta)$. Considering LOCAL model definition, it is straightforward;
- the number of bits transmitted by each vertex is $O(\log\Delta)O(n,\Delta)$. Each message includes taken or chosen color by its sender, $\Delta+1$ color is used, and the number of messages is $O(n,\Delta)$;
- the number of time slots is $O(\Delta)O(n,\Delta)$. DVCA is composed of some rounds, and each round is composed of some time slots; in a time slot a vertex can broadcast its chosen or taken color. In each round, all vertices may need to broadcast their chosen colors and then taken colors. The minimum number of time slots in a round, required for broadcasting chosen and taken colors, is $2\chi(G)$. Since finding $\chi(G)$ is NP-hard, it is considered $2(\Delta+1)$. In DVCA with $O(n,\Delta)$ rounds and $O(\Delta)$ time slots in each round, the number of time slots required to color all graph is $O(\Delta)O(n,\Delta)$.

We use these complexities for comparison. By using the DVCA for MCP, the complexities are also the same but the number of vertices (n') and the maximum vertex degree (Δ') in the transformed graph are obtained by using equations (2) and (5), respectively.

3. Vertex coloring problems

In this section, the four coloring problems are defined. Then it is explained how MCP and BMCP problems can be converted into VCP and BCP, respectively. At first, we define the most generalized problem (BMCP) as following:

Given an undirected graph $G=(V,E)$ with vertex set V, edge set $E\subset V\times V$, vertex weight $w(v)$ for each vertex $v\in V$, and edge weight $w(u,v)$ for each $(u,v)\in E$, a legal $(K+1)$-bandwidth multicoloring is assigning

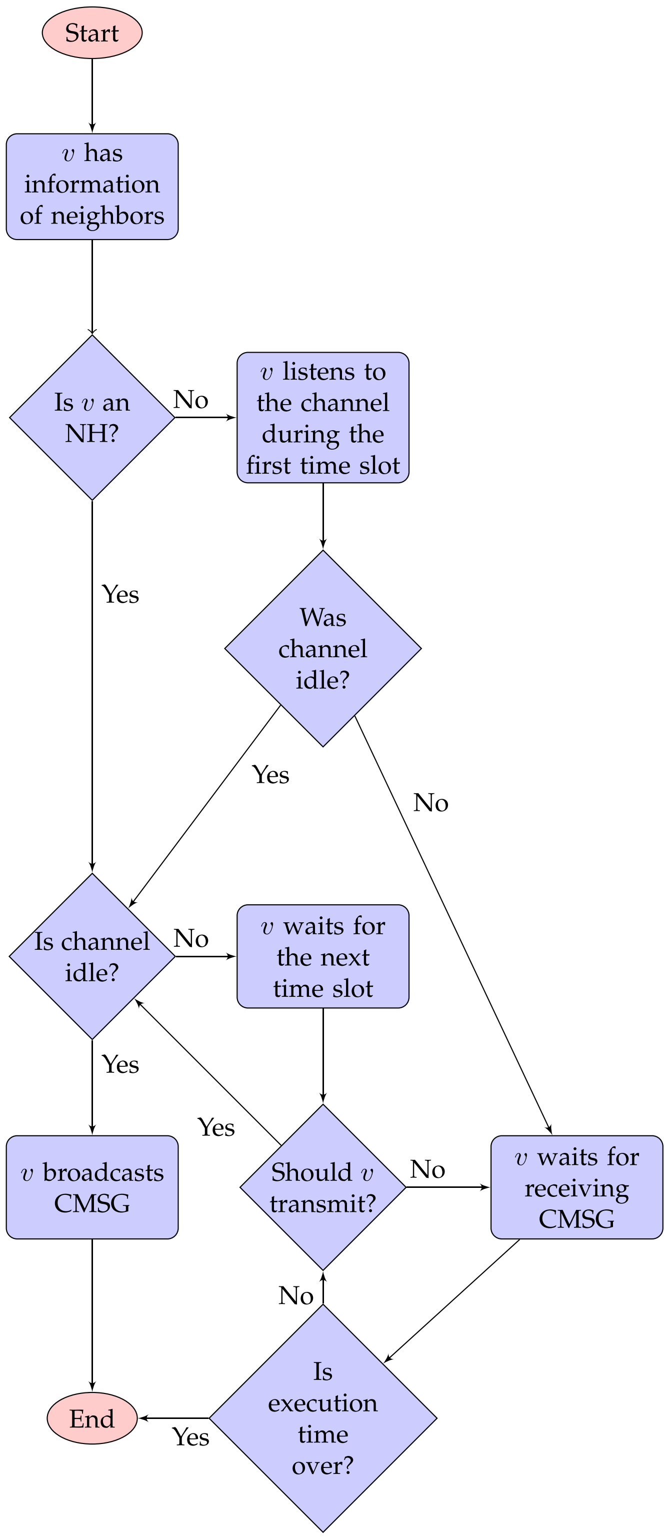
Start
v has information of neighbors
Is v an NH?
No
v listens to the channel during the first time slot
Yes
Was channel idle?
Yes
No
Is channel idle?
No
v waits for the next time slot
Yes
Yes
v broadcasts CMSG
Should v transmit?
No
v waits for receiving CMSG
No
Is execution time over?
End
Yes

Fig. 1. How vertex v determines whether it must broadcast CMSG.

$c(v)$ colors to v such that $c(v) \subset \{0, 1, \ldots, K\}, |c(v)| = w(v)$, and the absolute difference between any two colors in $c(u)$ and $c(v)$ of an edge $(u, v) \in E$ is at least $w(u, v)$, that is

$$|c(u) - c(v)| \geq w(u, v), \forall (u, v) \in E. \tag{1}$$

MCP is a special case of BMCP where $w(u,v) = 1, \forall (u,v) \in E$; BCP is a special case of BMCP where $w(v) = 1, \forall v \in V$. VCP is a special case of BMCP where $w(u,v) = 1, \forall (u,v) \in E$ and $w(v) = 1, \forall v \in V$.

As it was said before, MCP can be converted into VCP after transformation of the graph. Transformation of graph G into graph $T(G)$ is done as follows. For each vertex v in G, there is a clique with size $w(v)$ in $T(G)$; where a clique is a complete subgraph. Any vertex of a clique is connected to all vertices of another clique in $T(G)$ if and only if the two corresponding vertices in G are adjacent [20]. The number of vertices and edges of the transformed graph are obtained by equations (2) and (3), respectively [20]:

$$n' = |V'| = \sum_{v \in V} w(v) \tag{2}$$

$$|E'| = \sum_{v \in V} \frac{w(v)(w(v) - 1)}{2} + \sum_{(u,v) \in E} w(u)w(v) \tag{3}$$

In transformation, each v in G is converted to a clique in $T(G)$. The degree of v', any vertex in the clique corresponded to v, is obtained by equation (4). The maximum vertex degree in graph $T(G)$ is obtained by using equation (5). Furthermore, for each vertex v in G, we define new metric $d'(v)$ as (6).

$$d(v') = \sum_{u \in V(v)} w(u) - 1 \tag{4}$$

$$\Delta' = max_{v' \in T(G)} \{d(v')\} = max_{v \in V} \left\{ \sum_{u \in V(v)} w(u) \right\} - 1 \tag{5}$$

$$d'(v) = d(v') \tag{6}$$

BMCP problem can be converted into BCP and solved by using a centralized algorithm after the transformation of the graph [17,19–21].

4. General Distributed Vertex Coloring Algorithm

In GDVCA algorithm, some (and not necessarily all) vertices assign colors to themselves and their neighbors. Colors assignment must be declared by broadcasting Coloring MeSsaGe (CMSG). GDVCA consists of two parts. Part one determines which vertices and when must assign colors, which Fig. 1 shows it. Part two is related to how a vertex assigns color(s) to itself and its neighbors, which Algorithm 1 shows it.

Before coloring, each vertex has broadcasted its ID and neighbors' IDs and received neighbors' information. Each vertex $v \in V$ composes graph $G(v)$ induced by $V(v)$, where $V(v)$ includes v and its neighbors. Vertices may not start coloring at the same time because they are not synchronized. Coloring duration consists of some time slots. The upper bound for the number of time slots, "execution time" in Fig. 1, is estimated using Theorem 1 and simulation results, which will be described later. Each vertex has a unique ID. ID can be the unique MAC address of the node that the vertex represents or the coordinator node (BS or hub) if the vertex represents a network.

Nodes (vertices) access to the channel using slotted Carrier Sense Multiple Access (CSMA). CSMA is used to determine the availability of the channel before a node begins its transmission. The channel is checked first whether it is busy (channel is being accessed by other nodes) or it is in idle state (channel is available) before transmission [42]. Every time slot is composed of 2 parts. Part one consists of R mini-slots. A vertex before broadcasting CMSG, chooses one mini-slot from $\{0, 1, \ldots, R-1\}$ and senses the channel during the chosen mini-slot. If the channel is idle, it broadcasts the message; otherwise it waits for the next slot. The length of part two equals the maximum length of CMSG. In order to avoid choosing the same mini-slot by two or more adjacent vertices, vertices must choose a mini-slot randomly. On the other hand, the recipients of CMSG require to know the beginning of the next slot. It can be possible if they know the chosen mini-slot by the sender. To do this, the number of

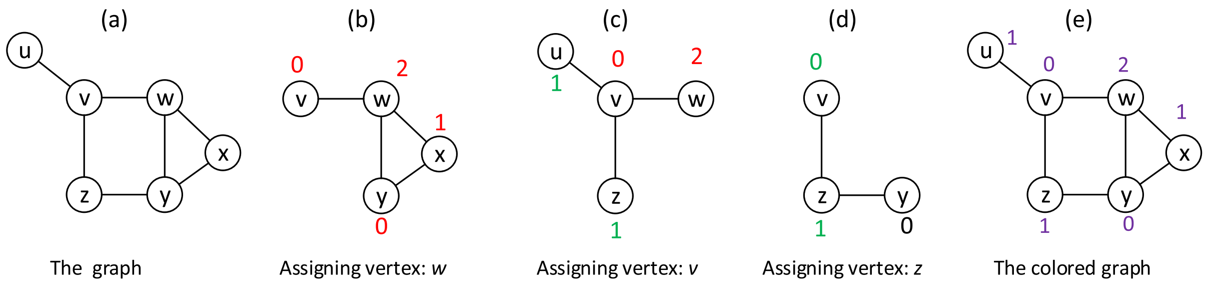

Fig. 2. Coloring a graph using GDVCA.

mini-slot chosen by a sender vertex is set to $r = (ID\text{mod}R)$, where ID is the identifier of the sender vertex and *mod* is the remainder of $ID \div R$.

CSMA protocols suffer from “hidden node problem” and “exposed node problem” [43]. Hidden node problem occurs when a node is sending frames without realizing the presence of another node that is currently transmitting frames to the same destination. Transmission from either one of these hidden nodes may result in collision at the receiver. This will lead to increases in message loss ratio and execution time of a coloring algorithm. Exposed node problem occurs when a node is mistakenly restrained from transmitting frame due to overhearing transmission from neighbors fearing that collisions might happen [42]. This unnecessary prevention from transmitting frames lead to increase in execution time of a coloring algorithm. In GDVCA design, these two problems have been considered and solved. In GDVCA, when a vertex broadcasts a CMSG, all its neighbors require to receive that and they should not transmit at that time. In other words, there is no exposed node problem in GDVCA. To solve hidden node problem, every vertex must be able to sense the channel as busy when at least one of its neighbors or neighbor of neighbors is transmitting [44]. Thus, the value of “energy detection threshold” is determined such that the range of busy detection is twice of the range of reception. The range of reception is the distance that a vertex can receive a CMSG correctly, and the range of busy detection is the distance that a vertex reports the channel as busy if the power of the received signal is above of the energy detection threshold. In other words, vertex v can receive from vertices with distance 1 and detect energy when vertices with distance 1 or 2 are transmitting CMSG.

At the beginning of coloring, each vertex determines wether it is a Neighbors Head (NH). NH vertices try to start coloring. Between any two adjacent vertices, at most one of them is an NH. NH determination depends on the problem. For the VCP and BCP problems, an NH vertex has the maximum degree among its neighbors; in other words, v is an NH if $d(v) = \max_{u \in V(v)}\{d(u)\}$. If some neighbors of the vertex also have the maximum degree, then the vertex is an NH if its ID is greater than the IDs of those neighbors. For the MCP and BMCP, v is an NH if $d'(v) = \max_{u \in V(v)}\{d'(u)\}$, where $d'(v)$ is obtained by equation (6). If some v's neighbors have the maximum value too; then the degrees, weights, and IDs of those neighbors are considered to determine wether v is an NH, respectively. The weights are considered only if the degrees are also the same, and so on.

In the first time slot, an NH vertex tries to access the channel and broadcast CMSG; a non-NH vertex listens to the channel for the whole time slot. If a non-NH vertex senses the channel as busy, it waits to receive CMSG. Otherwise, it tries to access the channel and broadcast CMSG at the second time slot. In any situation, if a vertex wants to broadcast CMSG and senses the channel as busy, it waits for the next time slot. If a vertex still (at the beginning of the next time slot) has to broadcast CMSG, it broadcasts CMSG if the channel is idle. The vertex continues this procedure until the channel becomes idle and it broadcasts CMSG.

After vertex v received a CMSG, if it has not broadcasted CMSG, it must broadcast a CMSG in any of the following two cases:

- some of v's neighbors have not been assigned colors yet;
- if all v's neighbors have been assigned colors, v has been colored by more than one vertex, and colors assigned by different neighbors are not the same, then v must assign itself the minimum available color(s). If there is at lease one neighbor that has not broadcasted CMSG yet, v must broadcast CMSG and inform its neighbors of colors assigned to itself.

CMSG includes a field that shows the time slot number of its sender. If a vertex has not received any CMSG yet, it sets this field to zero. Otherwise, it uses the sender's timing; if a vertex received more than one CMSG with different timings, it would use the timing of the last CMSG's sender. A recipient vertex knows the beginning of the received message and the chosen mini slot by its sender; so it knows the beginning of the current time slot and its number. The length of time slot is constant, so the recipient knows the end of the current time slot and the end of the algorithm. Vertex v can finish the coloring algorithm if:

Algorithm 1
colors assignment to $V(v)$ by vertex v using GDVCA

```
Result: $C(V(v))$
1  $v$ has information of its neighbors (their ID, their
   neighbors, CMSG broadcasted by them);
2  if $v$ has not received any CMSG yet then
3      $UCV = V(v)$
4  end
5  else
6      $UCV = v$'s neighbors that have not been colored
       yet;
7      if $v$ has been assigned colors by two or more neighbors,
       but the sets of colors are not the same then
8          $UCV = UCV \cup \{v\}$;
9      end
10 end
11 foreach $u \in UCV$ do
12     $c(u) = \emptyset$;
13 end
14 foreach $u \in (V(v) - UCV)$ do
15     $c(u) = c_0(u)$;
16 end
17 $v$ colors $UCV$ using Algorithm 2, 3, 4, or 5 for the
   VCP, MCP, BCP, or BMCP problem, respectively.
```

- it broadcasted CMSG; or
- it has received CMSG from all its neighbors and it has obtained $w(v)$ allowed colors.

Algorithm 1 shows how assigning vertex v assigns colors to $V(v)$. At first, it determines the UnColored Vertices (the set $UCV \subset V(v)$, lines 2–10). If v has not received CMSG yet, then $UCV = V(v)$. If v has been assigned different color(s) sets by neighbors, v must be included in UCV. At the next step (lines 11–17), v assigns color(s) to $V(v)$ vertices. v assigns to each $u \in (V(v) - UCV)$ what u has been assigned before. To color other vertices (UCV), v uses Algorithm 2, 3, 4, or 5 for the VCP, MCP, BCP, or BMCP problem, respectively. In other words, Algorithm 1 is the common and initial part of Algorithm 2, 3, 4, or 5 to solve the VCP, MCP, BCP, or BMCP problem, respectively. Any algorithm or method that has been suggested to color G centrally in related work, can be used by v to color UCV based on v's information. In this paper, the greedy algorithm along with a heuristic method is used to solve the four vertex coloring problems because heuristic methods are easy to implement and fast [13,14,18,45]. The heuristic method for each problem is described later.

Fig. 2 shows how the graph in Fig. 2(a) is colored using GDVCA; the problem is the VCP. According to NH definition, vertices $v, w,$ or y may be an NH and the other vertices can not. If the w's ID is greater than both v and y's IDs, then w is the only NH in the graph. Otherwise, v,y, or both are NHs. We consider the case w is the only NH; consequently it starts coloring. The graph induced by w and its neighbors ($V(w)$) has been shown in Fig. 2(b) w colors it with 3 colors and informs its neighbors by a CMSG. After that, v and y try to access the channel and broadcast a CMSG; because they have received a CMSG and they have uncolored neighbors. If v accesses to the channel sooner, it colors the induced graph $G(v)$ as Fig. 2(c). Then, z and y try to access the channel. If z accesses to the channel sooner, colors assignment is as Fig. 2(d). Vertex y has been colored by two vertices and the colors are the same, so y does nothing. If two different colors were assigned to it, y would assign itself the minimum color that has not been assigned to its neighbors.

By definition, communication/bit complexity of a distributed algorithm is the total number of messages/bits transmitted during its execution. [46] states that the cost of a distributed algorithm is both time and bits. Bit complexity is considered as a finer measure of communication complexity [1,2]. It has been used as a measuring cost of coloring algorithms by related work. One important matter that is not considered by them is that any communication between two or more nodes in a distributed wireless network must be done by physical frames. A physical frame includes header, payload (bits related to colors information), and trailer. The overhead (header and trailer) can be even more than the payload in bit. For example in IEEE 802.11n (WiFi), the overhead of physical and MAC layers for “management frame” is 376 bits [47]. Related work, for example [2], considers only payload that its length is $\log(\Delta+1)$ bits. In case wireless network is based on IEEE 802.11n and $\Delta < 2^{376}$, the length of payload is less than the overhead. Thus, the first and the most important step to reduce the number of bits is to reduce the number of messages, and the second step is to reduce the number and length of fields that must be present in the MAC frame body. GDVCA does the first step excellently and each vertex broadcasts at most one CMSG. The number of bits transmitted by a vertex is zero if it is not an assigning vertex. Otherwise, it is the length of the message transmitted by the assigning vertex.

A CMSG transmitted by vertex v should include colors assigned to all ($V(v)$) or some vertices (UCV). If CMSG includes colors assigned to $V(v)$ vertices, there are no need for extra fields to tell which color(s) has been assigned to which neighbor because v's neighbors know v's neighbors and their sequence. If a CMSG includes colors assigned to UCV vertices, fields are needed to show which color(s) assigned to which neighbor. v has $d(v)$ neighbors and its neighbors know the sequence. Therefore the length of each field, which shows the position in the sequence, is $\log(d(v)+1)$. A 1-bit field is required to show which case is used. The smaller case in bit is used. The number of bits transmitted by v is at most $H + \min\left\{n_1 * \log(n_2), \sum_{u \in UCV} (\log(d(v)+1) + w(u) * \log(n_3))\right\}$, where H is the length of the overhead in bit. The overhead includes physical and MAC headers and trailer, the field that tells the number of bits used for each color, the 1-bit field that shows which case is used, and the number of current time slot. The two arguments of the *min* function are related to the first and second cases; n_1 is the sum of colors that vertices $V(v)$ require; n_2 is the maximum color that is used for coloring $V(v)$; n_3 is the maximum color that is used for coloring UCV; and $w(u)$ is the number of colors required by vertex u. The values of $n_1, n_2,$ and n_3 for each coloring problem will be calculated in the corresponding section. Time complexity of GDVCA is independent of the problem and determined by the following theorem.

Theorem 1. *The number of time slots required to color n-vertex graph G is* $O(\Delta^2)$ *and at most n.*

Proof. *Each vertex is assigned color(s) by itself or its neighbors using CMSG. NH and non-NH vertices broadcast CMSG at the first and second time slots, respectively; unless they sense the channel as busy. In case of busy, an NH tries to broadcast CMSG at the next slot and non-NH waits to receive a CMSG and broadcasts after that; unless they do not need to broadcast CMSG; they do not need to broadcast if they and all their neighbors have been properly colored. The number of neighbors and neighbors of neighbors for each vertex is at most* Δ *and* Δ^2, *respectively. Therefore, each vertex receives or broadcasts a CMSG in* $O(\Delta^2)$ *time slots. In other words, the graph is colored in* $O(\Delta^2)$ *time slots. On the other hand, the number of neighbors plus the number of neighbors of neighbors is less than n. Consequently, the theorem is proved.*

In many applications, the maximum number of neighbors (Δ) is limited. In other words, the number of interfering/conflicting nodes for each node is limited. According to Theorem 1, coloring the graph of this network using GDVCA takes a finite time even if the number of vertices approached infinity. Moreover, coloring Special graphs using GDVCA has interesting features.

Definition 1. If there is at least one vertex in graph G that is connected to all other vertices, then G is a Special graph.

For examples, complete, wheel, and windmill graphs are Special graphs.

Theorem 2. *If G is a Special graph, the numbers of time slots and messages required to color G are one. Moreover, coloring can be done with the minimum number of colors.*

Proof. *If there is only one vertex (v) that is connected to all other vertices, then and according to NH definition, v is the only NH in the graph. If there are some vertices that are connected to all other vertices, then only one of them is an NH (v), because from any two adjacent vertices, at most one of them is an NH. The NH is connected to all other vertices. After it broadcasts CMSG, all vertices of G are assigned colors, and there is no need any vertex to broadcast CMSG. Moreover, coloring G by the NH is similar to a centralized algorithm and can be done with the minimum number of colors.*

4.1. Vertex coloring problem

In this subsection, it is described how assigning vertex v colors the uncolored set UCV and the problem is the VCP. Then, the upper bound of colors and bit complexity of the algorithm are calculated. The number of messages transmitted by each vertex is at most one and the number of time slots is $O(\Delta^2)$ and at most n, for the four problems.

Algorithm 2 shows how v assigns colors to the uncolored set UCV; or how v implements line 17 of Algorithm 1. At first, v finds the next vertex for color assignment using a heuristic method. The heuristic method is the combination of saturation degree, largest-degree first, and largest-ID first; it has been proposed by Ref. [4,45]. Saturation degree of a vertex is the number of different colors assigned to the vertex's neighbors. v does not have complete knowledge of colors assigned to the neighbors of its neighbors, so it computes the saturation degree of its neighbors based on what it has. If two vertices have the maximum saturation degree, their degrees and IDs are considered, respectively. ID is used only if their degrees are also the same.

Algorithm 2
colors assignment to *UCV* vertices by vertex *v* for the VCP problem

```
   Result: $C(UCV)$
1  while There are uncolored vertices do
2      $v$ chooses the next vertex ($u$) from uncolored
       neighbors based on saturation degree, degree $d(.)$,
       and ID;
3      $c = 0$;
4      while $c(u)$ is empty do
5          if $c$ is not assigned to $u$'s neighbors then
6              $c(u) = \{c\}$;
7          end
8          $c = c + 1$;
9      end
10 end
```

Theorem 3. *The number of colors used by GDVCA to assign exactly one color to each vertex of G, is at most $\Delta + 1$.*

Proof. *In GDVCA, some vertices assign colors using the greedy algorithm (Algorithm 2). If v has not received CMSG, according to Brook's theorem it can color $G(v)$ with at most $\Delta + 1$ colors. Because the number of vertices in $G(v)$ is $d(v) + 1$ and $d(v) \le \Delta$. Otherwise, in which k vertices of $G(v)$ have been colored with at most k colors, v can color remaining vertices (UCV) with at most $\Delta + 1 - k$ colors.*

Theorem 4. *The number of bits transmitted by assigning vertex v is at most $H + \min\{(d(v)+1)^*\log(\Delta+1), |UCV|\log((d(v)+1)(\Delta(v)+1))\}$, where $\Delta(v) = \max_{u\in V(v)}\{d(u)\} \le \Delta$.*

Proof. *Considering the explanations for obtaining the bit complexity of GDVCA in* Section 4, *we require to prove that $n_1 = d(v) + 1, n_2 = \Delta + 1$,* and *$n_3 = \Delta(v) + 1$. In the VCP problem, $w(u) = 1\ \forall u \in V$, so the sum of colors needed by $V(v)$ is $n_1 = d(v) + 1$. Based on* Theorem 3, *the maximum number of colors is at most $n_2 = \Delta + 1$. Vertex v can assign to $\forall u \in V(v)$ a color less than $d(u) + 1$ independent of the colors assigned to u's neighbors, so the maximum color used to color UCV is less than $n_3 = \max_{u\in V(v)}\{d(u)\} + 1 = \Delta(v) + 1$.*

4.2. Multicoloring problem

To solve MCP using DVCA, time, communication, and bit complexities are functions of Δ' and n' which are obtained by equations (2) and (5), respectively. The values of n' and Δ' can be very large. Therefore, using DVCA for the MCP is not a good solution because the mentioned disadvantages of DVCA are directly proportional to the number of vertices and the maximum vertex degree of the graph. But the complexities of GDVCA are low and coloring is done as following.

Algorithm 3 shows how *v* assigns colors to *UCV*. At the beginning of the algorithm, *v* selects the next vertex in *UCV* using the heuristic method (line 2 of the algorithm). Then, *v* assigns the minimum allowed colors to the chosen vertex.

Algorithm 3
colors assignment to *UCV* vertices by vertex *v* for the MCP problem

```
   Result: $C(UCV)$
1  while There are uncolored vertices do
2      $v$ chooses the next vertex ($u$) from uncolored
       neighbors based on saturation degree, $d'(.)$, degree
       $d(.)$, and ID;
3      $c = 0$;
4      while $|c(u)| < w(u)$ do
5          if $c$ is not assigned to $u$'s neighbors then
6              $c(u) = c(u) \cup \{c\}$;
7          end
8          $c = c + 1$;
9      end
10 end
```

Theorem 5. *To solve MCP using GDVCA, the number of colors is at most $\Delta' + 1$.*

Proof. *v assigns colors by using* Algorithm 3. *If v has not received CMSG yet, it can color $G(v)$ with colors less than $d'(v) + 1$ and $d'(v) \le \Delta'$. The maximum color in $\{0, 1, \ldots, d'(v)\}$ is used when each color is assigned just to one vertex. If v has received CMSG before, $V(v) - UCV$ vertices have been colored with at most $\sum_{u\in(V(v)-UCV)} w(u)$ colors. v can color remaining vertices with $\Delta' + 1 - \sum_{u\in(V(v)-UCV)} w(u)$ colors.*■

Theorem 6. *The number of bits transmitted by assigning vertex v is at most $H + \min\left\{(d'(v)+1)^*\log(\Delta'+1), |UCV|^*\log(d(v)+1) + \log(\Delta'(v) - +1)^* \sum_{u\in UCV} w(u)\right\}$, where $\Delta'(v) = \max_{u\in V(v)}\{d'(u)\} \le \Delta'$.*

Proof. *Considering the explanations for obtaining the bit complexity of GDVCA in* Section 4, *we require to prove that $n_1 = d'(v) + 1, n_2 = \Delta' + 1$,* and *$n_3 = \Delta'(v) + 1$. In the MCP problem, the sum of colors needed by $V(v)$ is $n_1 = \sum_{u\in V(v)} w(u) = d'(v) + 1$. Based on* Theorem 5, *the maximum number of colors is at most $n_2 = \Delta' + 1$. Vertex v can assign to $\forall u \in V(v)$ colors less than $d'(u) + 1$, independent of the colors assigned to u's neighbors, so the maximum color used to color UCV is less than $n_3 = \max_{u\in V(v)}\{d'(u)\} + 1 = \Delta'(v) + 1$.*

The costs of two algorithms DVCA and GDVCA are respectively:

- the numbers of messages transmitted by each vertex are $O(n', \Delta')$ and at most 1. For a graph, they are $n'O(n', \Delta')$ and at most n;
- the numbers of bits transmitted by each vertex are $O(\log\Delta')O(n', \Delta')$ and $O(\Delta'\log\Delta')$;
- the numbers of time slots are $O(\Delta')O(n', \Delta')$ and $O(\Delta^2)$.

4.3. Bandwidth coloring problem

To solve BCP centrally, a series of BCP problems with a given number of colors *k* are solved [13,17,21]bib21. Several algorithms have been proposed to solve the *k*-BCP problem, for example learning-based hybrid search [21], combination of tabu search approach with a specialized crossover operator [13], and multistart iterated tabu search [17]. The initial number of colors *k* must be large enough. The initial *k* can equal $\sum_{(u,v)\in E} w(u,v) + 1$ but this upper bound is very weak [16]. Therefore, *k* is obtained by using fast greedy heuristic algorithms [13,14,16]bib16. As soon as a legal k-BCP is found, they decrease k to $k-1$ and solve the new k-BCP problem. This process is repeated until no legal k-coloring can be found and the result is the last k for which a legal k-coloring is reached [13,17,19,21]bib21. GDVCA can be used as a distributed solution for the BCP. Any presented algorithm for BCP to color G can be used by v to color $G(v)$.

Assigning vertex v colors UCV using Algorithm 4. In the beginning, it chooses the next vertex for color assignment based on their scores. [13] proposes equation (7) to obtain the v's score; it is a function of the number of different colors assigned to its adjacent vertices and the distance to be respected between the color of the vertex and the colors of its neighbors. If two vertices have the maximum score, their degrees and IDs are considered. Then, it assigns the minimum allowed color to the chosen vertex.

$$s(v) = \sum_{h=1,\dots,k} \max_{u\in(V(v)-\{v\}) \text{ and } c(u)=h} w(u,v) \tag{7}$$

Algorithm 4

colors assignment to UCV by vertex v for the BCP problem

```
Result: $C(UCV)$
1  while There are uncolored vertices do
2      $v$ chooses the next vertex $(u)$ from uncolored
       neighbors based on score $s(.)$, degree $d(.)$, and ID;
3      $c = 0$;
4      while $c(u)$ is empty do
5          if $|c - c(z)| \geq d(u,z)$ for each colored vertex $z$
           that is a neighbor of $u$ then
6              $c(u) = \{c\}$;
7          end
8          $c = c + 1$;
9      end
10 end
```

Theorem 7. *To solve BCP using GDVCA , the number of colors is at most* $\sum_{(u,v)\in E} w(u,v) + 1$.

Proof. *If v has not received CMSG yet, it assigns color 0 to the first vertex of $G(v)$; color k to the second vertex, where $k=0$ if the two vertices are not connected otherwise k's value is the weight of the edge between the two vertices and so on; finally, color $\sum_{(u,v)\in E(v)} w(u,v)$ to the last vertex, where $E(v)$ includes $G(v)$'s edges. Otherwise, in which assigning vertex v has received CMSG before, each vertex of set $V(v)-UCV$ has been colored with color $\sum_{(u,v)\in E_1} w(u,v)$, where $E_1 \subset E$ and no edges in E_1 are connected to UCV vertices. v can color UCV with $\sum_{(u,v)\in(E_1\cup E_2)} w(u,v)$ colors, where $E_2 \subset E(v)$ and its members are connected to UCV vertices. The equation $(E_1 \cup E_2) \subset E$ implies that the maximum color used by v is less than the upper bound.*

Theorem 8. *The number of bits transmitted by assigning vertex v is at most* $H + \min\Big\{(d(v)+1)*\log\big(\sum_{(u,v)\in E} w(u,v)+1\big),\ |UCV|*\log\big((d(v)+1)*\big(\sum_{(u,v)\in E} w(u,v)+1\big)\big)\Big\}$.

Proof. *In the BCP problem, the sum of colors used by $V(v)$ is $d(v)+1$. Moreover, $\sum_{(u,v)\in E} w(u,v)+1$ is an upper bound for coloring $V, V(v)$, or UCV. Therefore, the theorem is proved.*

4.4. Bandwidth multicoloring problem

GDVCA solves the BMCP in a distributed manner in a time complexity independent of the number and weights of vertices as follows. Assigning vertex v colors UCV using Algorithm 4. In the beginning, it chooses the next vertex for colors assignment using equation (7). If two vertices have the maximum score then their $d'(.)$, $d(.)$, and IDs are considered. After that, it assigns the minimum allowed colors to the chosen vertex. The following lemma provides an upper bound for the number of colors used by any centralized greedy algorithm. Theorem 9 states that it is also an upper bound for the number of colors used by our proposed algorithm.

Lemma 1. *To solve BMCP using any centralized greedy algorithm,* $\sum_{v\in V} w(v) + \sum_{(u,v)\in E} w(u,v) + 1 - n$ *is an upper bound for the number of colors.*

Proof. *Set $V=\{v_1, v_2, \dots, v_n\}$ denotes the vertices, and set $[a,b]$ includes colors $a, a+1, \dots, b$. Vertex v_1 is assigned $[0, w(v_1)-1]$. Vertex v_2 is assigned colors $[0, w(v_2)-1]$ if it is not connected to v_1; otherwise, v_2 is assigned $[w(v_1)+w(v_1,v_2)-1, w(v_1)+w(v_2)+w(v_1,v_2)-2]$. Continuing in the same way, vertex v_n can be assigned colors $\Big[\sum_{u\in V-\{v_n\}} w(u) + \sum_{(u,v)\in(E-E_1)} w(u,v) - (n-1), \sum_{v\in V} w(v) + \sum_{(u,v)\in E} w(u,v) - n\Big]$, where E_1 is the set of all edges that are connected to v_n.*

When $w(u) = 1\ \forall u \in V$, the BMCP is converted into BCP, and the bound $\sum_{v\in V} w(v) + \sum_{(u,v)\in E} w(u,v) + 1 - n$ is converted into $\sum_{(u,v)\in E} w(u,v)+1$, which is the upper bound of colors for solving BCP.

Theorem 9. *To solve BMCP using GDVCA,* $\sum_{u\in V} w(u) + \sum_{(u,v)\in E} w(u,v) + 1 - n$ *is an upper bound for the number of colors.*

Proof. *If v has not received CMSG yet, it can color $G(v)$ with $\sum_{u\in V(v)} w(u) + \sum_{(u,v)\in E(v)} w(u,v) + 1 - |V(v)|$ colors based on Lemma 1. If assigning vertex v has received CMSG before, each colored vertex in set $V(v)-UCV$ has been colored with colors less than $\sum_{u\in V_1} w(u) + \sum_{(u,v)\in E_1} w(u,v) + 1 - |V_1|$; where $V_1 \subset V$ and does not include any vertex of UVC, and $E_1 \subset E$ and none of edges in E_1 are connected to UCV vertices. v can color UCV vertices with colors less than $\sum_{u\in(V_1\cup UCV)} w(u) + \sum_{(u,v)\in(E_1\cup E_2)} w(u,v) + 1 - |V_1 \cup UCV|$; where $E_2 \subset E(v)$ and its members are connected to UCV vertices. The equations $(V_1 \cup V_2) \subset V$ and $(E_1 \cup E_2) \subset E$ imply that the maximum color used by v is less than the upper bound.*

Theorem 10. *The number of bits transmitted by assigning vertex v is at most* $H + \min\Big\{(d'(v)+1)*\log\big(\sum_{u\in V} w(u) + \sum_{(u,v)\in E} w(u,v) + 1 - n\big),\ |UCV|*\log(d(v)+1) + \log\big(\sum_{u\in V} w(u) + \sum_{(u,v)\in E} w(u,v) + 1 - n\big)*\sum_{u\in UCV} w(u)\Big\}$

Proof. *The sum of colors used by $V(v)$ is $d'(v)+1$. Considering the upper bound of colors (Theorem 9), the theorem is proved.*

Table 4
Detailed simulation results on VCP instances.

Instance name	Δ	#Colors (central coloring)	#Colors (GDVCA)	#Trans mitted messages	#Time slots
GEOM20	4	3	3	6	3
GEOM20a	8	6	6	7	4
GEOM20b	7	5	5	8	4
GEOM307	6	6	12	5	
GEOM30a	12	7	7	12	6
GEOM30b	9	7	7	14	7
GEOM40	10	5	6	19	7
GEOM40a	16	9	10	16	9
GEOM40b	17	10	10	18	8
GEOM50	11	7	7	19	7
GEOM50a	16	11	12	35	15
GEOM50b	21	13	13	15	9
GEOM60	15	7	9	39	13
GEOM60a	18	10	11	31	13
GEOM60b	22	11	13	39	20
GEOM70	18	10	11	35	14
GEOM70a	25	15	17	48	26
GEOM70b	25	13	14	31	16
GEOM80	19	10	11	46	16
GEOM80a	27	15	16	48	26
GEOM80b	28	13	16	38	24
GEOM90	21	12	12	39	13
GEOM90a	31	17	20	57	33
GEOM90b	38	19	21	48	24
GEOM100	22	13	13	46	15
GEOM100a	40	20	21	58	34
GEOM100b	35	18	23	56	32
GEOM110	25	15	15	64	23
GEOM110a	46	21	21	41	26
GEOM110b	40	20	23	62	33
GEOM120	22	13	14	66	18
GEOM120a	46	23	25	63	38
GEOM120b	46	22	24	63	37

Table 5
Detailed simulation results on MCP instances.

Instance Name	Δ	Δ′	#Colors (central coloring)	#Colors (GDV CA)	#Trans mitted messages	# Time slots
GEOM20	4	29	19	19	5	3
GEOM20a	8	33	23	23	8	4
GEOM20b	7	16	10	10	7	4
GEOM30	7	42	37	37	10	5
GEOM30a	12	84	50	50	14	7
GEOM30b	9	22	14	14	14	7
GEOM40	10	57	29	33	20	8
GEOM40a	16	91	49	49	19	10
GEOM40b	17	34	19	21	20	11
GEOM50	11	73	44	44	14	5
GEOM50a	16	102	67	68	26	13
GEOM50b	21	48	28	29	17	10
GEOM60	15	108	54	55	39	16
GEOM60a	18	106	57	66	30	14
GEOM60b	22	45	21	24	39	20
GEOM70	18	109	56	61	36	14
GEOM70a	25	161	79	96	49	21
GEOM70b	25	50	24	27	36	18
GEOM80	19	91	50	58	45	16
GEOM80a	27	176	93	104	41	20
GEOM80b	28	52	25	31	49	28
GEOM90	21	136	75	75	46	14
GEOM90a	31	173	96	112	55	32
GEOM90b	38	85	40	46	52	27
GEOM100	22	118	79	79	48	17
GEOM100a	40	265	126	135	58	33
GEOM100b	35	67	32	36	57	32
GEOM110	25	143	77	77	70	23
GEOM110a	46	269	117	126	47	30
GEOM110b	40	89	41	49	63	34
GEOM120	22	130	85	92	73	18
GEOM120a	46	263	115	139	67	39
GEOM120b	46	90	40	48	63	38

Algorithm 5
colors assignment to *UCV* by vertex *v* for the BMCP problem

```
   Result: $C(UCV)$
1  while There are uncolored vertices do
2      $v$ chooses the next vertex $(u)$ from uncolored
       neighbors based on score $s(.)$, $d'(.)$, degree $d(.)$,
       and ID;
3      $c = 0$;
4      while $|c(u)| < w(u)$ do
5          if $|c - c(z)| \geq d(u, z)$ for each colored vertex $z$
            that is a neighbor of $u$ then
6              $c(u) = c(u) \cup \{c\}$;
7          end
8          $c = c + 1$;
9      end
10 end
```

5. Simulation results and discussions

To evaluate GDVCA performance, we use 33 benchmark geometric instances given by Trick. They have been used by related work to assess their proposed algorithms for MCP, BCP, and BMCP problems [9,13,14,16,17,20,21]. These instances belong to three kinds: GEOM*n*, GEOM*n*a, and GEOM*n*b, where *n* represents the number of vertices in the graph. In these graphs, the vertices are randomly generated in a 10,000 by 10,000 grid and are connected if they are close enough together. Edge weights are inversely proportional to the distance between the corresponding vertices. The weights of vertices are uniformly randomly generated between 1 and 10 for sets GEOM*n* and GEOM*n*a, and between 1 and 3 for set GEOM*n*b. GEOM*n* instances are sparse and the two others are denser graphs [9,13].

The aim of related work has been presenting a new metaheuristic centralized coloring algorithm to reduce the number of colors to solve the generalized vertex coloring problems in an acceptable time. In their simulations, the metrics have been "number of colors" and "time". The aim of our work is presenting a distributed algorithm to solve those problems with low time and colors. In simulations of our work, the metrics are also "number of colors" and "time".

GDVCA is a distributed algorithm and its results must be compared to a distributed coloring algorithm. But there are no distributed algorithms for BCP and BMCP. Therefore, we use a centralized vertex coloring for each problem to evaluate GDVCA efficiency (the number of colors). For each problem, the centralized algorithm and Algorithm 1 use the greedy algorithm along with a simple heuristic method to color *G* and *G*(*v*), respectively. The simple heuristic method for each problem was explained in the subsections of Section 4. The approach, used to color *G* or *G*(*v*), is simple and fast. For comparison, consider this example. [13] uses a tabu search algorithm to solve BCP and BMCP problems. At first, it performs 20 runs of the greedy algorithm; the minimum number of colors that has been used is the initial number of colors *k*. Then the tabu search algorithm runs for many iterations to decrease *k*. The number of iterations for solving BMCP on the instance GEOM60b is 878,357,241.

5.1. Simulation results of GDVCA on the VCP instances

This subsection is dedicated to an evaluation of the GDVCA performance for the VCP using 33 VCP benchmark graphs. Table 4 gives the detailed results. Columns in the table are Instance name, Δ (the number of colors used by DVCA to color *G* minus one), the number of colors used by the centralized coloring algorithm, the number of colors used by

Table 6
Detailed simulation results on BCP instances.

Instance name	Δ	#Colors (central coloring)	#Colors (GDV CA)	#Trans mitted messages	# Time slots
GEOM20	4	11	11	5	3
GEOM20a	8	30	30	8	4
GEOM20b	7	31	31	6	3
GEOM30	7	30	30	11	5
GEOM30a	12	48	48	12	6
GEOM30b	9	41	38	14	7
GEOM40	10	29	29	21	7
GEOM40a	16	38	38	17	9
GEOM40b	17	58	58	21	10
GEOM50	11	47	47	18	8
GEOM50a	16	60	72	33	16
GEOM50b	21	54	54	16	10
GEOM60	15	36	43	35	13
GEOM60a	18	57	59	32	15
GEOM60b	22	53	59	34	18
GEOM70	18	45	53	32	12
GEOM70a	25	75	75	27	13
GEOM70b	25	64	64	33	16
GEOM80	19	56	56	48	17
GEOM80a	27	79	96	41	22
GEOM80b	28	63	70	44	20
GEOM90	21	57	60	49	15
GEOM90a	31	84	107	57	34
GEOM90b	38	93	114	50	26
GEOM100	22	92	92	50	18
GEOM100a	40	91	102	48	31
GEOM100b	35	97	97	56	33
GEOM110	25	71	72	66	23
GEOM110a	46	98	98	44	27
GEOM110b	40	84	109	57	31
GEOM120	22	70	77	68	19
GEOM120a	46	113	116	61	37
GEOM120b	46	106	114	64	38

Table 7
Detailed simulation results on BMCP instances.

Instance Name	Δ	Δ′	#Colors (central coloring)	#Colors (GDV CA)	#Trans mitted messages	# Time slots
GEOM20	4	29	27	27	5	3
GEOM20a	8	33	45	53	8	4
GEOM20b	7	16	37	37	6	3
GEOM30	7	42	64	64	10	5
GEOM30a	12	84	91	91	12	6
GEOM30b	9	22	47	43	14	7
GEOM40	10	57	54	62	20	8
GEOM40a	16	91	91	106	16	9
GEOM40b	17	34	65	65	20	11
GEOM50	11	73	95	94	15	5
GEOM50a	16	102	119	118	27	14
GEOM50b	21	48	78	86	19	11
GEOM60	15	108	95	99	36	14
GEOM60a	18	106	118	115	33	15
GEOM60b	22	45	72	78	36	21
GEOM70	18	109	102	130	34	14
GEOM70a	25	161	155	183	45	19
GEOM70b	25	50	83	85	35	18
GEOM80	19	91	118	104	45	17
GEOM80a	27	176	197	218	43	21
GEOM80b	28	52	99	101	46	26
GEOM90	21	136	141	141	49	14
GEOM90a	31	173	200	217	57	33
GEOM90b	38	85	121	155	53	28
GEOM100	22	118	171	154	49	17
GEOM100a	40	265	282	289	60	34
GEOM100b	35	67	114	138	59	34
GEOM110	25	143	178	177	73	26
GEOM110a	46	269	267	281	51	33
GEOM110b	40	89	122	148	65	36
GEOM120	22	130	181	172	77	20
GEOM120a	46	263	249	278	69	40
GEOM120b	46	90	163	154	65	36

GDVCA, the number of CMSG transmitted by all vertices (the number of vertices that have broadcasted CMSG), and the number of times slots required to color G using GDVCA. Comparing columns 2 and 4, it can be observed that the number of colors used by our algorithm is always less than $\Delta + 1$; it is about $\frac{\Delta+1}{2}$. Moreover, the numbers of colors used by GDVCA, which is a distributed algorithm, and the centralized algorithm are close. Column 5 reports that about half of vertices have broadcasted CMSG. Column 6 shows that the number of time slots required to color G is always less than Δ, whereas by using DVCA, it is always more than Δ.

5.2. Simulation results of GDVCA on the MCP instances

The results of simulations for the MCP have been summarized in Table 5. Compared to Table 4, one column has been added that shows Δ' (the number of colors used by DVCA minus one). Column 5 shows that the number of colors used by GDVCA is far from $\Delta' + 1$ and close to the number of colors used by the centralized algorithm. Column 6 reports that about half of vertices have broadcasted CMSG. Finally, column 7 shows that the number of time slots required to color G is less than Δ; except in instance GEOM90a that they are 32 and 31, respectively.

5.3. Simulation results of GDVCA on the BCP instances and BMCP instances

The results of the simulations for BCP and BMCP have been summarized in Tables 6 and 7, respectively. Observing the tables, conclusions are the same for the four coloring problems. The numbers of colors used by GDVCA and the centralized coloring algorithm are near. About half of vertices have broadcasted CMSG. And required time is less than Δ time slots.

Comparing the last two columns of the four tables, one can observe that the number of messages and time required to solve the problems are very close. In other words, simulation results confirm that time and communication complexities of GDVCA are independent of the weights of vertices and edges.

6. Conclusion

The vertex coloring problem and its generalizations have a lot of applications in computer networks. But proposed algorithms have serious issues that prevent them from being usable in real world especially in wireless networks. We have proposed a distributed vertex coloring algorithm (GDVCA) for vertex coloring problems in wireless networks.

GDVCA does not require synchronization nor the knowledge of the entire graph. Each vertex only requires to know its neighbors and if they are connected together, so there is no need to relay the messages of other vertices. It means that initial overhead of GDVCA is very low. The number of messages transmitted by each vertex for coloring is at most 1. Simulation results of GDVCA on 33 benchmark instances for each problem, show that about half of the vertices broadcast coloring message and half of the vertices do not. GDVCA algorithm requires less colors than the upper bound used by centralized coloring algorithms. Simulation results report that the numbers of colors used for the problems using GDVCA and the corresponding centralized algorithms are close together. Besides, the algorithm can solve any of the four problems with the minimum number of colors on Special graphs. Time complexity of the proposed algorithm is a function of only maximum vertex degree. According to simulation results, the number of time slots required to color an instance is less than the maximum vertex degree in the instance. Therefore, coloring using GDVCA takes less time compared to DVCA.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Mohammadhasan Miri: Writing - original draft, Software, Investigation. **Yousef Darmani:** Software, Investigation. **Kamal Mohamedpour:** Conceptualization, Methodology. **R. Lal Tummala:** Conceptualization, Writing - review & editing. **Mahasweta Sarkar:** Writing - review & editing, Validation.

References

[1] Metivier Y, Robson JM, Saheb-Djahromi N, Zemmari A. An analysis of an optimal bit complexity randomised distributed vertex colouring algorithm. International Conference On Principles Of Distributed Systems Dec 2009:359–64.
[2] Robson JM, Metivier Y, Saheb-Djahromi N, Zemmari A. About randomised distributed graph colouring and graph partition algorithms. Inf Comput Nov 2010; 208(11).
[3] Choudhary S, Purohit GN. Distributed algorithm for optimized vertex coloring. In: 2010 international conference on methods and models in computer science (ICM2CS-2010); 2010. p. 65–9.
[4] Omari HA, Sabri KE. New graph coloring algorithms. J Math Stat 2006;2:439–41.
[5] Blas AD, Jagota A, Hughey R. Energy function-based approaches to graph coloring. IEEE Trans Neural Network 2002;13(1):81–91.
[6] Barenboim L, Elkin M, Goldenberg U. Locally-iterative distributed (Δ+1)-coloring below szegedy-vishwanathan barrier, and applications to self-stabilization and to restricted-bandwidth models. In: Proceedings of the 2018 ACM symposium on principles of distributed computing, ser. PODC '18. New York, NY, USA: ACM; 2018. p. 437–46. https://doi.org/10.1145/3212734.3212769 [Online]. Available:.
[7] Harris DG, Schneider J, Su H-H. Distributed (Δ+1)-coloring in sublogarithmic rounds. J ACM Apr. 2018;65(4):19:1–19:21. https://doi.org/10.1145/3178120 [Online]. Available:.
[8] Konrad C, Zamaraev V. Distributed minimum vertex coloring and maximum independent set in chordal graphs. In: 44th international symposium on mathematical foundations of computer science (MFCS 2019), ser. Leibniz international proceedings in informatics (LIPIcs). vol. 138; 2019. 21:1–21:15.
[9] Lim A, Zhu Y, Lou Q, Rodrigues B. Heuristic methods for graph coloring problems. In: Proceedings of the 2005 ACM symposium on applied computing, ser. SAC '05; 2005. p. 933–9.
[10] Bamas E, Esperet L. Distributed coloring of graphs with an optimal number of colors. In: 36Th international symposium on theoretical aspects of computer science (stacs 2019); 2019. p. 10. https://doi.org/10.4230/LIPIcs.STACS.2019.10 [Online]. Available: http://infoscience.epfl.ch/record/268183.
[11] Kothapalli K, Scheideler C, Onus M, Schindelhauer C. Distributed coloring in $\tilde{O}(\sqrt{logn})$ bit rounds. In: Proceedings 20th IEEE international parallel and distributed processing symposium, rhodes island. vol. 24; 2006. p. 10. 3.
[12] Kosowski A, Manuszewski K. Classical coloring of graphs. Contemp Math 2004;352: 1–20.
[13] Malaguti E, Toth P. An evolutionary approach for bandwidth multicoloring problems. Eur J Oper Res 2008;189(3):638–51.
[14] Fijuljanin J. Two genetic algorithms for the bandwidth multicoloring problem. Yugosl J Oper Res 2012;22(44):225–46.
[15] Aardal KI, van Hoesel SPM, Koster AMCA, Mannino C, Sassano A. Models and solution techniques for frequency assignment problems. Ann Oper Res 2007;153(1): 79–129.
[16] Dias B, de Freitasa R, Maculan N, Michelon P. Solving the bandwidth coloring problem applying constraint and integer programming techniques [Online]. Available: http://www.optimization-online.org; 2016.
[17] Xiangjing L, Zhipeng L. Multistart iterated tabu search for bandwidth coloring problem. Comput Oper Res 2013;40(5):1401–9.
[18] Marti R, Gortazar F, Duarte A. Heuristics for the bandwidth colouring problem. Int J Metaheuristics May 2010;1(1):11–29.
[19] Lai X, Lü Z, Hao J, Glover F, Xu L. Path relinking for bandwidth coloring problem. CoRR 2014;abs/1409.0973 [Online]. Available: http://arxiv.org/abs/1409.0973.
[20] Han K, Kim C. An evolutionary approach for graph multi-coloring problem. Appl Math Sci Feb 2015;9(34):1677–84.
[21] Jin Y, Hao JK. Effective learning-based hybrid search for bandwidth coloring. IEEE Transactions on Systems, Man, and Cybernetics: Systems 2015;45(4):624–35.
[22] Su H-H, Vu HT. "Towards the locality of vizing's theorem. CoRR 2019;abs/1901.00479.
[23] Barenboim L, Elkin M, Pettie S, Schneider J. The locality of distributed symmetry breaking. J ACM Jun. 2016;63(3):20:1–20:45. https://doi.org/10.1145/2903137 [Online]. Available:.
[24] Parter M. (Δ+1) coloring in the congested clique model. In: 45th international colloquium on automata, languages, and programming (ICALP 2018), ser. Leibniz international proceedings in informatics (LIPIcs). vol. 107; 2018. 160:1–160:14.
[25] Panconesi A, Rizzi R. Some simple distributed algorithms for sparse networks. Distr Comput Apr 2001;14(2):97–100.
[26] Goldberg AV, Plotkin SA, Shannon GE. Parallel symmetry-breaking in sparse graphs. SIAM J Discrete Math 1988;1(4):434–46.
[27] Barenboim L. Deterministic (Δ+1)-coloring in sublinear (in Δ) time in static, dynamic and faulty networks. In: Proceedings of the 2015 ACM symposium on principles of distributed computing, ser. PODC '15; 2015. p. 345–54.
[28] Fraigniaud P, Heinrich M, Kosowski A. Local conflict coloring. In: 2016 IEEE 57th annual symposium on foundations of computer science (FOCS); 2016. p. 625–34.
[29] Alon N, Babai L, Itai A. A fast and simple randomized parallel algorithm for the maximal independent set problem. J Algorithm Dec. 1986;7(4):567–83.
[30] Schneider J, Wattenhofer R. A new technique for distributed symmetry breaking. In: Proceedings of the 29th ACM SIGACT-SIGOPS symposium on principles of distributed computing, ser. PODC '10; 2010. p. 257–66.
[31] Barenboim L, Elkin M, Pettie S, Schneider J. The locality of distributed symmetry breaking. In: 2012 IEEE 53rd annual symposium on foundations of computer science; Oct 2012. p. 321–30.
[32] Moscibroda T, Wattenhofer R. Coloring unstructured radio networks. In: Proceedings of the seventeenth annual ACM symposium on parallelism in algorithms and architectures, ser. SPAA '05; 2005. p. 39–48.
[33] Wattenhofer R, Moscibroda T. Coloring unstructured radio networks. Distr Comput Oct 2008;21(4):271–84.
[34] Schneider J, Wattenhofer R. Coloring unstructured wireless multi-hop networks. In: Proceedings of the 28th ACM symposium on principles of distributed computing, ser. PODC '09; 2009. p. 210–9.
[35] Moosavi H, Bui FM. Optimal relay selection and power control with quality-of-service provisioning in wireless body area networks. IEEE Trans Wireless Commun Aug 2016;15(8):5497–510.
[36] Zheng J, Jamalipour A. Wireless sensor networks: a networking perspective. New Jersy, USA: Wiley-IEEE Press; 2009.
[37] Barenboim L, Elkin M, Gavoille C. A fast network-decomposition algorithm and its applications to constant-time distributed computation. Theor Comput Sci 2016 [Online]. Available: http://www.sciencedirect.com/science/article/pii/S030439751630319.
[38] Linial N. Locality in distributed graph algorithms. SIAM J Comput Feb 1992;21(1): 193–201.
[39] Kuhn F. Faster deterministic distributed coloring through recursive list coloring. CoRR 2019;abs/1907.03797 [Online]. Available: http://arxiv.org/abs/1907.03797.
[40] Hefetz D, Kuhn F, Maus Y, Steger A. Polynomial lower bound for distributed graph coloring in a weak local model. In: Gavoille C, Ilcinkas D, editors. distributed computing. Berlin, Heidelberg: Springer Berlin Heidelberg; 2016. p. 99–113.
[41] Johansson O. Simple distributed (Δ+1)-coloring of graphs. Inf Process Lett 1999; 70:229–32. 70.
[42] Roslan I, Kawasaki T, Nishiue T, Takaki Y, Ohta C, Tamaki H. Control of transmission power and carrier sense threshold to enhance throughput and fairness for dense WLANs. In: 2016 international conference on information networking (ICOIN); Jan 2016. p. 51–6.
[43] Fullmer CL, Garcia-Luna-Aceves JJ. Complete single-channel solutions to hidden terminal problems in wireless lans. In: IEEE international conference on communications, ICC '97 montreal. vol. 2; Jun 1997. p. 575–9. 2.
[44] Thorpe C, Murphy L. A survey of adaptive carrier sensing mechanisms for ieee 802.11 wireless networks. IEEE Communications Surveys Tutorials Third 2014; 16(3):1266–93.
[45] Kumlander D, Kulitškov A. An experimental comparison of heuristic coloring algorithms in terms of found color classes on random graphs. In: Optimization of complex systems: theory, models, algorithms and applications. Cham: Springer International Publishing; 2020. p. 365–75.
[46] Santoro N. Design and analysis of distributed algorithms. New Jersy: John Wiley & Sons; 2007.
[47] IEEE802.11n-2009 standard for information technology– local and metropolitan area networks– specific requirements– Part 11: wireless LAN medium access control (MAC)and physical layer (PHY) specifications amendment 5: enhancements for higher throughput. Oct 2009.