k-Safe Labelings of Connected Graphs

Protik Bose Pranto, Bishal Basak Papan, Md. Saidur Rahman
Graph Drawing and Information Visualization Laboratory
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka-1205, Bangladesh
Email: 1505044.pbp@ugrad.cse.buet.ac.bd, 1505043.bbp@ugrad.cse.buet.ac.bd, saidurrahman@cse.buet.ac.bd

***Abstract*—In a k-safe labeling of a graph G, each vertex is labeled by a distinct positive integer such that the difference of the labels of two adjacent vertices is at least k. The span of a k-safe labeling of G is the range between the minimum and the maximum labels used in G. The k-safe labeling problem asks to label all the vertices of G using the minimum span. This problem has practical applications in assigning frequencies of transmitters in a network. k-safe labeling problem has been proven to be NP-hard and there is not an exact upper bound on the span of k-safe labeling of a graph. In this paper, we give an upper bound on k-safe labelings of all connected graphs based on the size of the maximum clique in the graph. Our proof leads to a polynomial-time algorithm for finding a k-safe labeling of any connected graph attaining the bound.**

***Index Terms*—graph labeling, k-safe labeling, bandwidth, frequency assignment**

I. Introduction

Graph labeling refers to assigning distinct values to all the vertices and/or edges of a graph G. This assigned value is called *label* of that vertex or edge. *Safe labeling* is a special type of graph labeling where each vertex is labeled by a distinct integer such that there is at least a certain difference between the labels of two adjacent vertices [1]. If the difference is an integer k, then this labeling is called a *k-safe labeling* of G. The range between the smallest (minimum) label and the largest (maximum) label assigned to the vertices of G is called the *span* of the k-safe labeling of G. Let the minimum label and the maximum label used in the k-safe labeling of G be respectively I_l and I_s, then the span of the k-safe labeling of G is $I_l - I_s + 1$. In k-safe labeling problem, we try to find a k-safe labeling of a given graph with the minimum span possible [2]. Throughout this paper, we will consider only simple connected graphs and 1 as the minimum label. According to our consideration, the span of a k-safe labeling of a graph will be I_l, if I_l is the maximum label. In Figure 1, we show two different k-safe labelings of the same graph of seven vertices.

Figure 1: For $k = 4$, (a) labeling of a graph with span 22, (b) labeling of the same graph with span 15.

A k-safe labeling of a graph has practical applications in communication engineering for radio frequency assignment of transmitters. In a city, if there are more than one mobile operators operating, the signal from one's transmitter tower can interfere with others' signal within the interference distance if their frequency difference is less than a certain threshold. This can lead to noisy signals and loss of information. To avoid these circumstances, each transmitter is assigned a distinct frequency such that two transmitters within the interference distance receive frequencies having a big (safe) difference keeping the bandwidth (span) minimum [3], [4]. If the whole network in the city is considered as a graph, the towers as vertices and the interference between two towers' signals as edges, then the frequency assignment of each tower avoiding interference with minimum bandwidth can be done by solving the k-safe labeling problem for the corresponding graph.

The k-safe labeling problem is not same as the anti-bandwidth problem [5], [6] or the graph coloring problem [7], [8] though one might think of these problems studying the problem of k-safe labeling. In the anti-bandwidth problem, we need to label the vertices of a graph of n vertices with the integers $1, 2, \ldots, n$ in such a way that each vertex receives a distinct label and the minimum difference of the labels of adjacent vertices is maximized. But, in the k-safe labeling problem, we can label the vertices of a graph of n vertices using labels greater than n so that each vertex receives a distinct label, the difference between the labels of two adjacent vertices is at least a fixed value, k and the range of the labels is as small as possible. In a vertex coloring problem, we need to assign an integer value to each vertex of a graph in such a way that two distinct vertices receive different values. These integer values represent the colors and the objective is to minimize the number of colors used. Here same integer can be used for more than two or more vertices if they are not adjacent, but in k-safe labeling problem, one integer can be used to label at most one vertex.

Habiba et al. proved that k-safe labeling problem is NP-hard [2]. They gave the lower bounds on the spans of k-safe labelings of complete graphs and complete bipartite graphs. They also gave linear time algorithms along with the upper bounds on the spans of k-safe labelings of trees, bipartite graphs, cycles and cactus graphs. But these upper bounds for trees and cactus graphs are not tight. For a complete graph of n vertices, the span for a k-safe labeling is at least $(n-1)k+1$.

978-1-6654-1700-6/21/$31.00 ©2021 IEEE

Although this span is optimal for complete graphs only, it is also the trivial upper bound on the span of k-safe labeling of any graph having n vertices. This upper bound is loose and there is no algorithm known for k-safe labeling of any graph within a span less than this trivial upper bound. In this paper, we give a polynomial time algorithm for k-safe labeling of a connected graph of which the upper bound of span is less than the trivial upper bound obtained from the work of Habiba et al [2]. Since, for $k = 1$, the optimal k-safe labeling of any graph can be done trivially, we will consider $k \geq 2$ throughout this paper.

The remaining of the paper is organized as follows. In Section II, we give some definitions. In Section III, a polynomial time algorithm of a k-safe labeling of connected graphs is given. In Section IV we establish an upper bound on k-safe labelings of connected graphs. In Section V, we show the experimental analysis of our algorithm. Finally, Section VI concludes the paper.

II. Preliminaries

In this section we give some terminologies that will be used throughout the paper. We refer [1] for graph theoretic terminologies which are not defined here.

Let $G = (V, E)$ be a simple, connected and undirected graph consisting of a set of vertices V and set of edges E. Usually we denote $n = |V|$ and $m = |E|$. Two vertices $u, v \in V$ are adjacent if and only if there is an edge $(u, v) \in E$. Vertices u and v are called *neighbors* to each other if they are adjacent in G. G is a *complete graph* if every pair of vertices $\{u, v\} \subseteq V$ are adjacent. A graph $G' = (V', E')$ is called a *induced subgraph* of $G = (V, E)$ if and only if $V' \subseteq V$ and $E' \subseteq E$ consists of all the edges of which both endpoints are in V'. A *clique* is an induced subgraph of V' that is complete. The *size of a clique* is the number of vertices in the clique. A *maximum clique* of a graph is a clique containing the maximum number of vertices among all cliques in G. The number of vertices in the maximum clique is called the *maximum clique size* of G.

The *complement* of a graph $G = (V, E)$ is a graph G' on the same vertex set V such that two distinct vertices of G' are adjacent if and only if they are not adjacent in G. G is a *dense* graph if the number of edges, m, is close to the maximal number of edges, $\binom{n}{2}$. If m is not close to $\binom{n}{2}$, then G is called a *sparse* graph.

III. An Algorithm for k-Safe Labeling

In this section we give an algorithm for finding a k-Safe Labeling of a connected graph. We first give some lemmas which lead to the basic idea behind the algorithm. Then we describe our algorithm with an example and analyze the time complexity and correctness of our algorithm.

Lemma 1. *Let $G = (V, E)$ be a complete graph of n vertices. After deleting r edges from G, the new maximum clique size of G will be at least $n - r$.*

Proof: This lemma can be proved trivially. ∎

Lemma 2. *Let $G = (V, E)$ be a simple connected graph and $G' = (V, E')$ is the graph obtained from G after deleting an edge $e \in E$. If G has a k-safe labeling of span S, then G' also admits a k-safe labeling of span at most S.*

Proof: This lemma can be proved trivially. ∎

We now describe our algorithm.

Let $G = (V, E)$ be a simple connected graph. We first take a complete graph $G_c = (V, E_c)$ and perform a k-safe labeling on G_c with a span of $(n-1)k + 1$ using the algorithm in [2]. Let $G' = (V, E')$ be the complement graph of G and $e_h \in E'$ where $e_h = (v_{h_1}, v_{h_2})$. We set $E_c = E_c - \{e_h\}$, $E' = E' - \{e_h\}$ and $V_e = \{u_{h_1}, u_{h_2}\}$. We store the edge removed in this step and perform relabeling of $G_c = (V, E_c)$. Next, in each step, we take an edge $e_{h'} = (u_{h'_1}, u_{h'_2})$, such that $e_{h'} \in E'$ and $u_{h'_1}, u_{h'_2} \notin V_e$. If no such $e_{h'} \in E'$ is found, we set $V_e = \{\}$ and take an edge $e_{h'} \in E'$. We remove $e_{h'}$ from G_c and set E_c, E', V_e accordingly. After that, we perform relabeling of $G_c = (V, E_c)$. We repeat the process until E' is empty.

We now give our relabeling method for each step. Let $e_1, e_2, \ldots, e_i$ be the edges which are removed until step i in this order. We first label the end vertices of e_i by $\{1, 2\}$ and label the end vertices of other removed edges in the reverse order of their removal with the smallest integer label satisfying the constraint of k-safe labeling. After labeling the end vertices of the removed edges, if there are unlabeled vertices in G_c, we assign the minimum possible label to each vertex that doesn't violate the constraints.

We store the label of each $v \in V$ in each step. If in any step, the span of the k-safe labeling becomes larger than the span in the previous step, we restore the label of each $v \in V$ from the label stored in the previous step. Otherwise, we keep the new label of each $v \in V$.

Let's look at the simulation of the algorithm on a graph $G = (V, E)$ of six vertices for $k = 5$. The graph we want to label is shown in Figure 2a. So, according to the algorithm, at first, we take a complete graph of six vertices and label each of the vertices of the complete graph using the algorithm in [2], which is shown in Figure 2b. The edge set E' of the complement graph of G is $\{(u_1, u_4), (u_3, u_5), (u_3, u_6)\}$. In this step the maximum label is 26 and the vertex u_6 is labeled as 26.

In the next iteration, we remove the edge (u_1, u_4) from the graph as well as from E' and obtain the vertices u_1 and u_4 to label. We set $V_e = \{u_1, u_4\}$. We label u_1 as 1 and u_4 as 2. Then, starting from the maximum labeled vertex on the previous step, which is u_6, we label the remaining four vertices with minimum integer values. This step is shown in Figure 2(c). We store the labeling in this step. In this step, the maximum label vertex is u_2.

From E', we get the edge (u_3, u_5), as $u_3, u_5 \notin V_e$. We remove (u_3, u_5) from the graph as well as from E' and label u_3, u_5 as 1, 2 respectively. We set $V_e = \{u_1, u_4, u_3, u_5\}$. As in the previous step, we removed the edge (u_1, u_4), after labeling u_3 and u_5, we label u_1 as 7 and u_4 as 8, as they are

the minimum possible labels that can be assigned. After that we label the vertex u_2 as 13 and then u_6 as 18. This step is shown in Figure 2(d). We replace the previous labeling stored by the labeling in this step.

In, E', there is only one edge remaining: (u_3, u_6), but $u_3 \in V_e$. Hence, after setting $V_e = \{\}$, we remove (u_3, u_6) from the graph as well as from E' and label u_3 as 1 and u_6 as 2. We set $V_e = \{u_3, u_6\}$. Then, from the edge removed in the previous step, we obtain the vertices u_3 and u_5 to label. As we have already labeled u_3, we label u_5 as 7. Then from the edge removed in the previous of the previous step, we obtain the vertices u_1 and u_4 to label. We label u_1 as 12 and u_4 as 13. Then we label the remaining vertex u_2 as 18. This step is shown in Figure 2(e). As E' is now empty, the algorithm terminates and the labeled graph shown in Figure 2(e) is the k-safe labeling of G according to the algorithm.

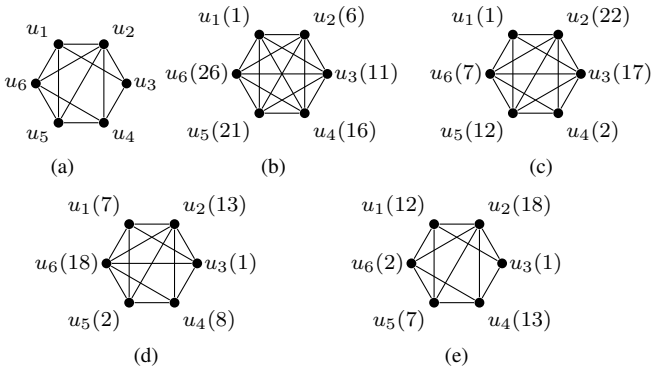

Figure 2: For $k = 5$, the steps of the algorithm for k-safe labeling of a graph of six vertices. (a) The given graph, (b) k-safe labeling of the corresponding complete graph, (c) k-safe labeling after removing edge (u_1, u_4) from step (b), (d) k-safe labeling after removing edge (u_3, u_5) from step (c), (e) k-safe labeling after removing edge (u_3, u_6) from step (d).

We call the algorithm described above Algorithm k-Safe Labeling. A formal pseudo code for the algorithm of k-Safe Labeling is given in Algorithm 1.

We now analyze the correctness of Algorithm k-Safe Labeling. According to the algorithm, when labeling a vertex, we check the label of each neighbor of that vertex so that the difference between the labels of two adjacent vertices is at least k. In addition to this, we also check all the labels assigned to other vertices so that a label is not used twice. As a result, while labeling a vertex according to Algorithm k-Safe Labeling, the constraints of k-safe labeling are never violated.

We now analyze the running-time of Algorithm k-Safe labeling. The labeling of the initial complete graph G_c can be done in $O(n)$ time. We choose one edge from G_c in each step and remove the edge by checking the endpoints of the previously removed edges in $O(1)$ time using a map like structure. We have to remove $\binom{n}{2} - m$ edges i.e., $O(n^2)$ edges in the worst case. Then we relabel the graph by checking and labeling the end vertices of the removed edges until this iteration. Checking the end vertices takes $O(n^2)$ time in the worst case. While labeling one end vertex we check the labels of all the neighbors of the vertex, which takes $O(n)$ times in the worst case. Then we find the unused integer from from 1 to at most $1 + (n-1)k$ in the worst case. As the maximum label is $O(nk)$, the labeling of one vertex takes $O(n^2k)$ time in the worst case. Thus, to label all the end vertices of the previously removed edges, we need $O(n^2(n^2k)) = O(n^4k)$ time in the worst case. After labeling the end vertices of the previously removed edges, we label the remaining unlabeled vertices which is of $O(n)$. This step takes $O(n^3k)$ time in the worst case. Therefore, the worst case time complexity of Algorithm k-Safe Labeling is $O(n + n^2(n^4k + n^3k)) = O(n^6k)$.

Algorithm 1 Safe-Labeling(G,k)

Input: An unlabeled graph $G(V, E)$, the value of k
Output: Labeled Graph $G_L(V, E)$

$G_c(V, E_c) \leftarrow Complete\ Graph\ with\ vertices\ V$
$G'(V, E') \leftarrow Complement\ of\ G$
$S \leftarrow \emptyset$; $M \leftarrow \emptyset$; $i \leftarrow 1$; $V_r \leftarrow \emptyset$
for each $v \in V$ **do**
    $M[v] \leftarrow i$
    $i \leftarrow i + k$
end for
$G_L(V, E_l) \leftarrow G_c$
while $E' \neq \emptyset$ **do**
    find an edge $e = (e_1, e_2) \in E'$ if $e_1, e_2 \notin V_r$
    if no such edge exists, then set $V_r = \emptyset$ and find e
    $E' \leftarrow E' - \{e\}$; $V_r \leftarrow V_r \cup \{e_1, e_2\}$
    $S \leftarrow S \cup (e_1, e_2)$
    $G_L \leftarrow G_L - \{e\}$
    $M \leftarrow$ LABEL_VERTICES(G_L, k, M, S)
end while
return labeled graph G_L

procedure LABEL_VERTICES(G_L, k, M, S)
$t \leftarrow S.size$
$max \leftarrow maximum\ value\ of\ M$
$M' \leftarrow M$
$M[v] \leftarrow 0$ for each $v \in V$
while all values in $M \neq 0$ **do**
    if $t > 0$ **then**
        $(a, b) \leftarrow S[t]$
        label the unlabeled end vertex(s) among a and b
        $M[a] \leftarrow$ label of a; $M[b] \leftarrow$ label of b
        $t \leftarrow t - 1$
    else
        $S' \leftarrow all\ unlabeled\ vertices$
        sort S' in decreasing order of values in M'
        label all $u \in S'$ with minimum label possible
    end if
    $M \leftarrow M'$ if maximum value of $M > max$
end while
return M
end procedure

IV. Bounds on k-Safe Labeling

In this section, we prove the upper bound on the span obtained by Algorithm k-Safe Labeling.

Let $G = (V, E)$ be a graph of n vertices. According to Algorithm k-Safe Labeling, we take a complete graph $G_c = (V, E_c)$ and in each step, we remove one edge $e \in E_c$ from G_c until $G_c = G$. In step i, let the maximum clique size of G_c be q_i and in step $i+1$, the maximum clique size of G_c be q_{i+1} where $q_i, q_{i+1} \leq n-1$. If $q_i = q_{i+1}$, then we define q_i as an *unaffected maximum clique size* of G_c. For the minimum value of i when q_i is an unaffected maximum clique size, we define q_i as the *first unaffected maximum clique size* and denote it as q_u.

To prove the upper bound in Theorem 1, we need the following lemmas:

Lemma 3. *Let $G = (V, E)$ be a complete graph of n vertices and k be the minimum difference between the labels of two adjacent vertices. According to Algorithm k-Safe Labeling, after removing r edges $(1 \leq r \leq \lfloor n/2 \rfloor)$ from G, a new subgraph G' is induced. If the maximum clique size of G' is $n-r$, then there is a set of r pair(s), each containing consecutive integer labels of two vertices i.e. $\{(1,2), (2+k, 2+k+1), \ldots, (r+(r-1)k, r+(r-1)k+1\}$.*

Proof: Suppose $V = \{u_1, u_2, \ldots, u_n\}$. As G is a complete graph, the maximum clique size is initially n. If, at first, we remove the edge (u_i, u_j), where $u_i, u_j \in V$, then the maximum clique size becomes $n-1$ and we can label u_i, u_j as 1 and 2. In the next step, if we remove an edge that is not incident to either of u_i and u_j, then the maximum clique size becomes $n-2$. We remove the edge $(u_{i'}, u_{j'})$, where $u_{i'}, u_{j'} \in V$. As both of $u_{i'}$ and $u_{j'}$ are neighbors of the vertex labeled as 2, then we can label them as 1 and 2 and the vertices u_i and u_j as $2+k$ and $3+k$ according to Algorithm k-Safe Labeling. In the next step, if we remove an edge which is not incident to any of $u_i, u_j, u_{i'}$ and $u_{j'}$, then the resultant graph will have a maximum clique of size $n-3$. Suppose we remove the edge $(u_{i''}, u_{j''})$ in this step and both of them are neighbors to the vertex labeled as $3+k$. Hence, we can label $u_{i''}$ and $u_{j''}$ as 1 and 2 respectively and the vertices u_i and u_j as $3+2k$ and $4+2k$ respectively according to Algorithm k-Safe Labeling. After removing r edges in this way, if the maximum clique size becomes $n-r$, then the we can label the vertices u_i and u_j as $r+(r-1)k$ and $r+(r-1)k+1$, according to Algorithm k-Safe Labeling. ∎

From this point, we consider $G = (V, E)$ as a simple, connected and undirected graph of n vertices, and k as the difference between the labels of two adjacent vertices in G.

Lemma 4. *Let q_c be the size of a clique in G and $Q \subset V$ be the set of vertices of that clique. For a vertex $v \in V$, $v \notin Q$ and v is neighbor of every vertex in Q. If v is the maximum labeled vertex among the vertices in $V \setminus Q$ and the label of v is l_v, then the maximum label of the vertices on this clique can be $q_c k + l_v$.*

Proof: Suppose $Q = \{u_{q_1}, u_{q_2}, \ldots, u_{q_c}\}$. According to Algorithm k-Safe Labeling, we want to label the vertices of Q in sequence $S_q = (u_{q_1}, u_{q_2}, \ldots, u_{q_c})$ after labeling $v \in V \setminus Q$ as l_v. As v is neighbor of every vertex in Q, we have to label u_{q_1} as at least $l_v + k$. Since Q is a clique of size q_c, in sequence S_q, the label of u_{q_c} will be $q_c k + l_v$. Similarly, for any such sequence, $q_c k + l_v$ will be the maximum label of the vertices of Q. ∎

Lemma 5. *Let the first unaffected maximum clique size of G be q_u. Then $q_u \geq \lceil n/2 \rceil$.*

Proof: The proof is trivial according to the definition of first unaffected maximum clique size. ∎

Now we describe and prove the upper bounds on Algorithm 1 described in Section III.

Theorem 1. *Let q be the maximum clique size and q_u be the first unaffected maximum clique size of G. Algorithm k-Safe Labeling obtains a k-safe labeling of G which admits an upper bound:*

$$S_u = \begin{cases} (q-1)k + n - q + 1 & \text{if } q \geq q_u \\ (q_u - 1)k + n - q_u + 1 & \text{if } q < q_u \end{cases}$$

Proof: Algorithm k-Safe Labeling starts from a complete graph $G_c = (V, E_c)$ of n vertices and runs in some iterations. In each iteration we delete an edge from G_c and then label all the vertices in the resultant graph. We continue this continues until the given graph, G, is labeled.

Let I_l be the maximum label at any iteration. So we will prove the theorem by removing an edge in every step and showing that $I_l \leq S_u$.

Case 1. *If $q \geq q_u$*

Suppose, we have removed r edges from G_c, where $1 \leq r \leq \lfloor n/2 \rfloor$. In this case, $q = n - r$ according to Lemma 1. According to Lemma 3, there is a set of r pairs each containing consecutive integer labels. Among all the labels in the set of pairs, the maximum label will be $r+(r-1)k+1$. Let this be the label of the $2r^{th}$ vertex, u_{2r}. The remaining $n-2r$ vertices can form a clique and u_{2r} is neighbor of every vertex on the clique. So, according to Lemma 4, $I_l = r + (r-1)k + 1 + (n-2r)k$.

As $q = n - r \implies r = n - q$, we can write that $I_l = n - q + (n-q-1)k + 1 + (n - 2n + 2q)k = (q-1)k + n - q + 1$. So, after removing r edges, $I_l = S_u$ when $q = n - r$ and $q \geq q_u$.

If $q > n - r$ and $q = q_u$, then after removing at most $r-1$ edges from G_c, we get a graph G_1 having a maximum clique of size $q_u = n - (r-1)$. According to (IV), G_1 admits a k-safe labeling of span S_u. After removing the r^{th} edge from G_c, we get the graph G which has a maximum clique of size q_u. According to Lemma 2, as G_1 has a k-safe labeling of span S_u, G also admits a k-safe labeling of span at most S_u. So, after removing r edges, $I_l \leq S_u$ when $q > n - r$ and $q = q_u$.

Case 2. *If $q < q_u$*

Suppose $G_1 = (V, E_1)$ was the graph having a maximum clique of size q_u, where $E_1 = E \cup e$. After removing e from G_c, we get G having a maximum clique of size $q < q_u$. From Case 1, we can see that G_1 admits a k-safe labeling within a span of $S_u = (q_u - 1)k + n - q_u + 1$. According to Lemma 2, as G is obtained by removing an edge from G_1, G also admits a k-safe labeling of span at most S_u. Thus for any $q < q_u$, G admits a k-safe labeling of span $I_l \leq S_u$, where $S_u = (q_u - 1)k + n - q_u + 1$. ■

V. Experimental Result

We have simulated our algorithm on different types of graphs generated by *networkx* library in *Python*. A Python implementation of this algorithm can be found from this link: https://github.com/bPapan/k_safe_labelings_of_connected_graphs. We have varied the values of the parameters: vertex count n and k using different number of edges. In all the runs, the spans always remain within our proposed upper bounds. In Table I, we show the summary after simulating the algorithm on some random graphs.

n	k	q	q_u	Span	Bound
10	15	3	5	47	66
12	25	5	7	128	156
12	20	4	6	66	107
15	29	5	9	177	239
18	20	6	10	145	189
20	12	5	11	90	117
20	13	7	11	99	140
25	15	10	17	127	217
25	20	4	13	103	206
25	23	7	13	281	289
30	50	4	16	352	531
33	50	8	18	606	866
35	30	9	19	485	557
40	30	20	24	707	707
45	4	15	24	107	114
50	45	25	26	1150	1150

TABLE I: Summary of experimental results of the developed "k-Safe Labeling" algorithm

Our algorithm gives better result for dense graph than for sparse graphs. For a dense graph, the span remains almost same as the upper bound. Figure 3 shows the simulation result of our algorithm on a graph of 50 vertices with different numbers of edges.

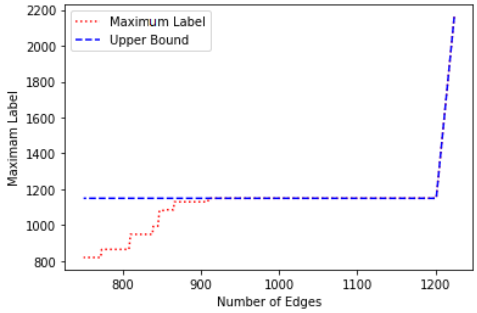

Figure 3: Simulation result on a graph of 50 vertices with edges varying from 1225 to 750 for $k = 45$.

In this figure, we can see that the span and the upper bound remains same as long as the graph remains dense. The reason behind this is, in the upper bounds that we have proven, when clique size, $q \geq q_u$, the upper bound changes along with the maximum clique size. The edge removal step in each iteration of Algorithm k-Safe Labeling allows us to label the vertices in an optimal way according to Lemma 3. This optimal k-safe labeling can be done until $q \geq q_u$. As a result, when $q \geq q_u$, the span in Algorithm k-Safe Labeling is equal to the upper bound. But when $q < q_u$, the bound does not change. According to Lemma 5, the first unaffected maximum clique size of a graph of n vertices is at most $\lceil n/2 \rceil$. But in sparse graphs, the maximum clique size might be less than $\lceil n/2 \rceil$. In such cases, though the maximum clique decreases than $\lceil n/2 \rceil$, the upper bound does not decrease. As the span of k-safe labeling depends on the maximum clique size, in Algorithm k-Safe Labeling, the difference between the span and the upper bound increases when the maximum clique size decreases from first unaffected maximum clique size.

VI. Conclusion and Future Works

In this paper, we have given an algorithm for k-safe labeling of any simple connected graph. We have also given an upper bound on the span of k-safe labeling of a connected graph obtained by the algorithm. There are several natural open problems raised by our work. The upper bound we gave is not tight for all classes of graphs. Finding tight upper bounds specifically for some classes of graphs is an open problem. As our algorithm starts k-safe labeling from a complete graph, it works reasonably faster for dense graphs than for sparse graphs. Developing an algorithm that will work equally fast for dense graphs and sparse graphs is an open problem.

Acknowledgement

The authors would like to thank anonymous referees for their suggestions for improving the presentation of the paper. The work is supported by the Basic Research Grant of BUET.

References

[1] M. S. Rahman, *Basic graph theory*. Springer, 2017.

[2] U. Habiba, M. S. Rahman, S. H. Lamia, and T. Chowdhury, "Safe labeling of graphs with minimum span," in *2015 International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*. IEEE, 2015, pp. 1–4.

[3] A. Reichman, M. Priesler, and A. Czylwik, "Time and frequency resource allocation in of DMA wireless mesh networks," in *OFDM 2012; 17th International OFDM Workshop 2012 (InOWo'12)*. VDE, 2012, pp. 1–6.

[4] J. Riihijarvi, M. Petrova, and P. Mahonen, "Frequency allocation for WLANs using graph colouring techniques," in *Second Annual Conference on Wireless On-demand Network Systems and Services*. IEEE, 2005, pp. 216–222.

[5] M. S. Rahaman, T. A. Eshan, S. Al Abdullah, and M. S. Rahman, "Antibandwidth problem for itchy caterpillars," in *2014 International Conference on Informatics, Electronics & Vision (ICIEV)*. IEEE, 2014, pp. 1–6.

[6] A. Raspaud, H. Schröder, O. Sỳkora, L. Torok, and I. Vrt'o, "Antibandwidth and cyclic antibandwidth of meshes and hypercubes," *Discrete Mathematics*, vol. 309, no. 11, pp. 3541–3552, 2009.

[7] D. Bozdağ, U. Catalyurek, A. H. Gebremedhin, F. Manne, E. G. Boman, and F. Özgüner, "A parallel distance-2 graph coloring algorithm for distributed memory computers," in *International conference on high performance computing and communications*. Springer, 2005, pp. 796–806.

[8] Y. Hu, S. Kobourov, and S. Veeramoni, "On maximum differential graph coloring," in *International Symposium on Graph Drawing*. Springer, 2010, pp. 274–286.