A Low Complexity based Edge Color Matching Algorithm for Regular Bipartite Multigraph

Rezaul Karim
Dept. of Computer Science & Engineering
University of Chittagong (CU)
Chittagong, Bangladesh

Md. Rashedul Islam
Dept. of Computer Science & Engineering
International Islamic University Chittagong (IIUC)
Chittagong, Bangladesh

Muhammad Mahbub Hasan Rony
Dept. of Computer Science & Engineering
International Islamic University Chittagong (IIUC)
Chittagong, Bangladesh

Md. Khaliluzzaman*
Dept. of Computer Science & Engineering
International Islamic University Chittagong (IIUC)
Chittagong, Bangladesh

***Abstract*—An edge coloring of a graph G is a process of assigning colors to the adjacent edges so that the adjacent edges represents the different colors. In this paper, an algorithm is proposed to find the perfect color matching of the regular bipartite multigraph with low time complexity. For that, the proposed algorithm is divided into two procedures. In the first procedure, the possible circuits and bad edges are extracted from the regular bipartite graph. In the second procedure, the bad edges are rearranged to obtain the perfect color matching. The depth first search (DFS) algorithm is used in this paper for traversing the bipartite vertices to find the closed path, open path, incomplete components, and bad edges. By the proposed algorithm, the proper edge coloring of D – regular bipartite multi-graph can be obtained in *O (D.V)* time.**

***Keywords*—matching; edge-coloring; complexity; bipartite multigraph; DFS**

I. INTRODUCTION

An edge coloring of a Graph is one of the well-known, exoteric researched topics in the arena of graph theory. Edge coloring of a graph G is used various colors, so that, the adjacent edges are obtained different colors. By using this concept of edge coloring many real-world problems can be solved. An edge coloring has applications in scheduling problems and in frequency assignment for fiber optic networks. It also used to solve the timetabling problem, register allocation, pattern matching, designing seating plans, solving Sudoku puzzles and so on.

This section provides a descriptive summary of some methods that have been implemented and tested at graph theory for solving edge coloring problems. This topic has gained importance for the purpose of efficient edge color matching in the different graphs. For example, in [1], proposed a method for edge coloring in which every $(3, \Delta)$-bipartite graph G, chromatic index $\leq 4\Delta$. This paper only considered the $(3, \Delta)$-bipartite simple graph. In [2], proposed an edge coloring method for course timetabling. One-sided interval colorings of a bipartite graph method are introduced in [3]. For any graph G with bipartite set (X, Y) where authors present upper $x'_{int}(G, X)$ for classes of bipartite graphs G with maximum degree $\Delta\ (G)$ at most 9. In particular, if $\Delta\ (G) = 4$, then $x'_{int}\ (G, X) \leq 6$ and so on. In [4], authors derived a theorem to find the closed paths from $C = M \cup N$ for matching M and N. A closed path C can be found in $O(|C|)$ time on average. This theorem helped to develop the proposed algorithm for minimal edge-coloring. This concept of open and closed path can be easily obtained from this theorem.

In [5], showed a theorem in which any edge color matching of a complete bipartite graph $K_{n,n}$ contains 18 vertexes with three colors. This method creates disjoint monochromatic cycles which together cover all vertices. The minimum number of cycles is required for this type of covering is 5. In [6], proposed an algorithm to find out two disjoint matching M_1 and M_2 for a given (X, Y) bipartite graph with set $S \subseteq X$, where, M_1 saturates X and M_2 saturates S. The problem was solved by finding and appropriate factor of the graph when $|S| \geq |X|-1$. In [7], proved a method for two bipartite graphs G and H, where, H is a fixed graph whose vertices will be shown as colors. And H-coloring of a graph G is a process of assigning colors for preserving adjacency in graph G.

In this paper, an algorithm is proposed that is developed to find a perfect color matching of a regular bipartite multigraph. This is done by dealing edge coloring with lower time complexity. For that, the proposed method is divided into some parts that are run with an independent time complexity and helps to reduce the overall time complexity.

The edge coloring of a bipartite multigraph is highly related to finding a perfect matching efficiently. To obtain the perfect edge color matching the proposed method is divided into two parts. In the first part, the Depth First Search (DFS) algorithm is used to extract the closed and opened path circuits as well as bad edges. In the second part, the bad edges are rearranged to find the perfect edge color matching.

The rest of the paper is organized as follows. The preliminaries of the graph theory are described in Section II. The proposed minimal edge color matching algorithm is introduced in the next section. Case studies are described in Section IV. Experimental results and discussions are explained in Section V. The papers are concluded in Section VI.

II. PRELIMINARIES

A Graph G is an ordered pair $G = (V, E)$ along with a set V of vertices, nodes or points simultaneously with a set E of edges, which are two elements subsets of V. On the other hand, Graph coloring problem is a process of marking out colors from definitive components of a graph subject to certain obligations. There are two terms of Graph coloring which are edge coloring and vertex coloring. In this paper, an algorithm is developed for edge coloring. A Graph is said to be regular when every vertex has the same degree. Where, a graph is said to be bipartite which has vertices and also can be separated into two disuniting sets (U, V). Multigraph is a graph which has multiple edges and all the edges have similar finishing nodes. The edges associate with a vertex inside U to another one V is termed as a closed path. On the other hand, a path in which the first and last vertices are distinct is thread as an open path. Furthermore, a matching in a graph G is a set of pairwise disjoint edges. The edge coloring of bipartite multigraph is highly related to find a perfect matching efficiently.

III. ALGORITHM FOR MINIMAL EDGE COLORING

Assume that, G (V, E) is a regular graph. To achieve this regular graph every vertex added the edges such that every vertex has degree exactly D. In the whole description the graph circuits are considered as a closed path, whereas, the set of paths may consist of opened paths and/or closed paths. So, when a path sets are considered then there may be consisting of opened paths or closed paths.

The method of Alexander Schriver's [4] is applied in this paper. In that paper, the authors derived a theorem to find the closed paths of $C=M \cup N$ for matching M and N and a closed path C can be found in $O|(C)|$ time on average. Accordingly, the regular matching from the graph is computed to extract the close and open paths. The matching is colored and removed from the graph. This process is applied recursively to extract all matching.

Let, C_1, C_2, ... are closed paths that can be constructed from the graph G. Let, VC_1, VC_2, ... are the set of vertices of the closed path C_1, C_2, ... respectively. In this paper some closed paths are found i.e., C_1, C_2, ... such that $Vc_1 \cup Vc_2 \cup .... \subseteq V$ *and* $Vc_1 \cap Vc_2 .... = O$. This means more than one closed path from a graph G can be received, where the vertices are disjoint. These received closed paths are produce two full matching. However, it is not possible to get closed paths that cover entire vertex in G. If it is possible to add two full matching of G then it may possible to obtain one or more closed path. So, after receiving $D/2$ closed path sets another set of edges are found that holds at most V edges. Note that, at most V number of edges in closed path set can be found. And to achieve these it may needs to find the closed path for several times.

Theorem 1: A perfect matching in a regular bipartite graph can be found in $O(V)$ time.

Proof: To prove this theorem, in this paper the Alexander Schriver's [4] is considered to get the perfect matching. It is proved from [4] is that it takes $O|(C)|$ time to get a closed path and edges in the closed path are equally decomposed into two matchings. If the average open path size is L then it will take O(L) times to get the opened paths. According to Alexander Schriver's, the union of the two matching construct at least one closed path. It is also possible to get more than one disjoint closed paths and/or open paths by the union of two full matching.

In this proposed method, one or more disjoint paths are taken such that each vertex reduces its degree by maximum two. This process will be continued by using DFS. After that, checking each closed path where the number of vertices must be equal to the number of edges. If this condition is not satisfied then start the DFS again to find another closed path. After getting the entire closed path, the process is terminated and starts to get the open paths. In the normal situations all the vertices in G within the closed path set couldn't be found. However, using this theorem the maximum number of closed path can be found. There are many vertices that are not visited in the graph. For that, find one or more opened path associated with these non-visited vertices and decomposes them into two matchings.

In this method, the maximum $V/2$ number of closed path or $V/2$ number of opened path can be found in each time and maximum V number odd edges can be found in the paths. Every closed or opened path will possibly be minimized by its length. Then the closed and opened path's edges are alternatively distributed into two matching. While finding the paths in this method is required $V+V/2 = O(V)$ time. Hence, $O(V)$ time is needed to find a perfect matching.

Theorem 2: A proper edge coloring of a D-regular bipartite multigraph can be found in $O(D.V)$ time.

Proof: According to Theorem 1 all closed path set can be computed by $D/2$ times and found D number of matching. However, all time these types of matching are not possible by this theorem. For this reason, two different processes are required for finding the D number of matching. In process 1, the possible closed and opened paths are extracted from the graph. And the incomplete component can be identified as bad edges. In process 2, the bad edges are rearranged at incomplete matching to find the D number of matching.

Procedure 1: Extraction of possible paths

Step1: Taking a node from graph for finding closed path

Fig. 1(a) is the processing example to describe the steps of the theorem. For finding the closed path from the graph, the searching is started from vertex 1 in Fig. 1(a).

Step 2: From the starting node a closed path using DFS is found and is removed it from the graph

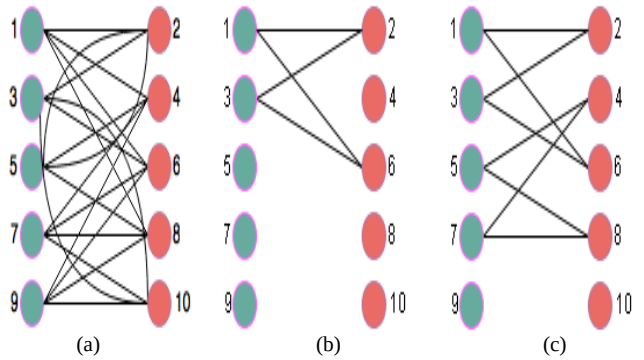

Fig. 1. Processing example: a) 4-regular bipartite multi-graph, b) first closed path with vertex 1, 2, 3, 6 c) all closed path from Fig. 1(a)

As the graph which contains the 10 vertices with degree 4. The closed and opened paths are obtained by reducing each vertex degree by two. According to step 2, from the starting node 1 the first closed path is found with the vertex 1, 2, 3, 6, 1 is shown in Fig. 1(b). By traversing the graph using the DFS the closed paths are obtained that are shown in Fig. 1(c).

Step 3: The edges in the closed path are distributed into two matching

The closed path in the Fig. 1(b) is distributed in two path set shown in Fig. 2(a) and Fig. 2(b). The closed paths that are in the Fig. 1(c) are distributed into two paths shown in Fig. 2(c) and Fig. 2(d).

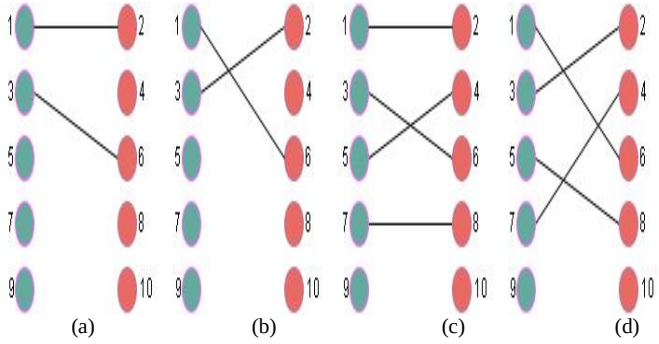

Fig. 2. The processing example of edge distribution for matching: a) and b) matching from Fig. 1(b), c) and b) matching form Fig. 1(c)

Step 4: If all the vertices are not traversed then repeat the process from Step 1 to Step 3 for non-visited vertices

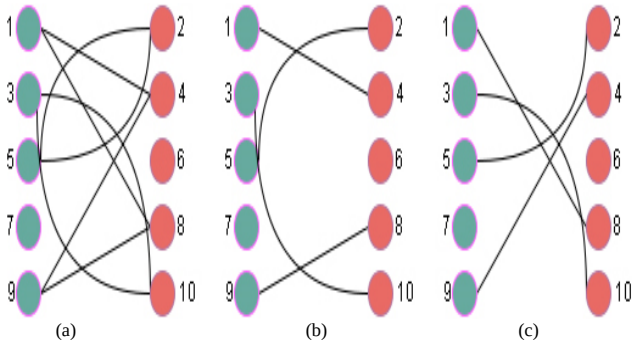

Fig. 3. Processing example of closed path extraction by repeated process: a) the closed path by repeated process, b) and c) matching from Fig. 3(a)

Since the vertices 9 and 10 in Fig. 1(a) are not visited by the DFS, Step 1 to step 3 have to be repeated. This repeating procedure traces the other closed paths that are not extracted by the previous steps shown in Fig. 3(a). The closed path in the Fig. 3(a) is distributed in two path sets that are shown in Fig. 3(b) and Fig. 3(c).

Step 5: If the remaining vertices do not make closed paths then start to find the opened paths

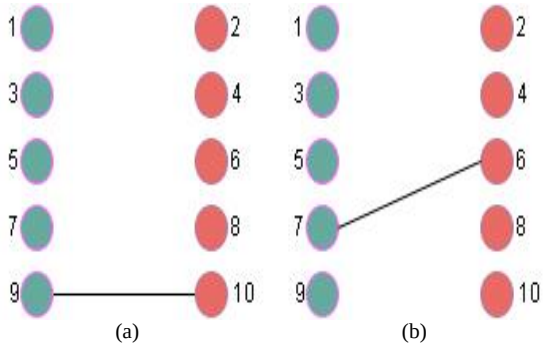

Fig. 4. The processing example of finding the opened path: a) and b) opened a paths from Fig. 1(a)

The opened path can be obtained from the vertices that are not used in the closed path. According to the Fig. 1(c), vertex 9 and 10 are not used to create any closed path. From Fig. 3(a), vertex 7 and 6 are not used to create any closed path. Those vertices are not connected by any edge. However, according to Fig. 1(a), vertex 9 and 10 as well as vertex 7 and 6 are connected by two separated edges. These separate edges are considered as opened path that are shown in Fig. 4(a) and Fig. 4(b).

Step 6: The edges of an opened path will be distributed in the matching (matching found in step 3).

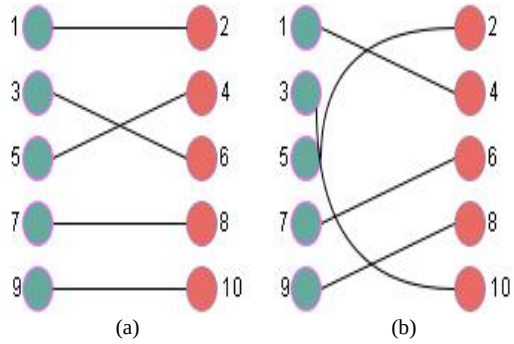

Fig. 5. The processing example of distributing opened paths into the matching: a) and b) open path in Fig. 4(a) and Fig. 4(b) are distributed into matching in Fig. 2(c) and Fig. 2(d)

The opened path shows in Fig. 4(a) and Fig. 4(b) are distributed in any two incomplete matching from Fig. 2(c), Fig. 2(d) as well as Fig. 3(b), and Fig. 3(c). In this case, the opened path in Fig. 4(a) and Fig. 4(b) are distributed in incomplete matching in Fig. 2(c) and Fig. 3(c) respectively. The remaining incomplete matching will be reconstructed in the procedure two.

Step 7: While all the vertices are traversed then switch for the next closed path set.

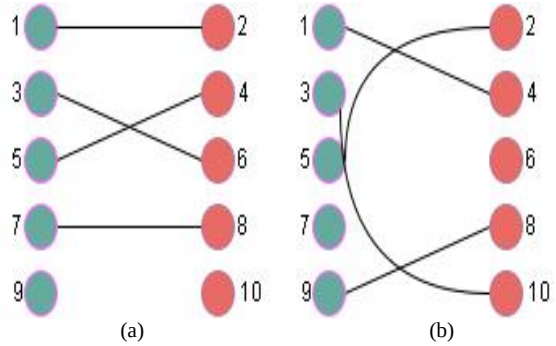

Fig. 6. The processing example of extracting components: a) components from the Fig. 2(d), b) components from the Fig. 3(b)

Step 8: Identify the component's (the vertices that have no edges) of an incomplete matching in a set

Since, the Fig. 2(d) and Fig. 3(b) are contained components, i.e., vertex 9, 10 and vertex 7, 6. That's why; Fig. 2(d) and Fig. 3(b) are considered as incomplete matching.

After *D/2* iteration, the process terminates, and gets *D/2* numbers of incomplete matching and *D/2* number of complete matchings. According to the processing example, each vertex has a degree of 4. The iteration process is performed in this processing example is two which is 4/2=2.

The algorithm in procedure 1 runs in *O (D.V)* time. This is obtained by multiplication of the number of paths being found and the total number of edges in the paths by D/2 iteration, i.e., *D/2*(V+ (V/2)) =3DV/4= O (D.V).*

After completing the procedure 1, there may exist many edges in the graph G that are not used while creating the closed and opened path are considered as bad edges.

Procedure 2: Finding the bad edges and rearranging them for perfect matching

Step 1: The remaining bad edges need to be re-distributed into an incomplete matching i.e., Fig. 2(d) and Fig. 3(b). According to the Fig. 1(a) only two edges are not used while creating the closed path that is edge(7, 10) and edge (9, 6) shown in Fig. 7(a). The bad edges are re-distributed into incomplete matching of Fig. 2(d).

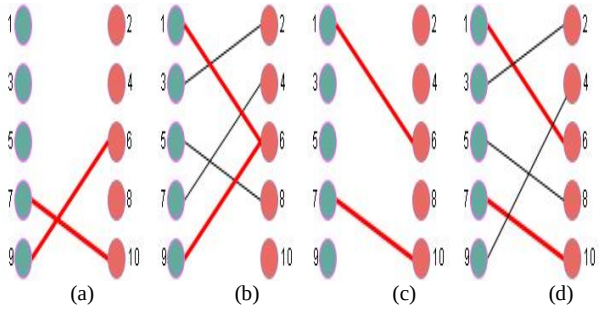

Fig. 7. The processing example of the bad edges: a) bad edges from Fig. 1(a), b) adjacent
edges after inserting the bad edge in Fig. 2(d), c) final bad edge set, d) final distribution of the bad edges

To re-distribute the bad edges of Fig. 7(a) into an incomplete matching i.e., Fig. 2(d), firstly traverse the incomplete matching by DFS to find the component. Here, the component is vertex 9 in Fig. 2(d). After that, remove the bad edge that is connected to vertex 9 from the bad edge set. This bad edge insertion may cause two adjacent edges in the matching as shown in Fig. 7(b). In this case, the bad edge that is connected to the vertex 9 is removed from the bad edge set and inserts into the other adjacent edge in the bad edge set i.e., edge (1, 6) as shown in Fig. 7(c). Finally, insert all the edges from Fig. 2(d) into Fig. 7(c) except the board edges shown in Fig. 7(d). Similarly, the bad edges are also re-distributed into incomplete matching of Fig. 3(b) is shown in Fig. 8. According to procedure one, two complete matching are found that are shown in Fig. 9(a) and Fig. 9(b) with different colors. By procedure two, another two complete matching are found that are shown in Fig. 10(a) and Fig. 10(b) with other different colors.

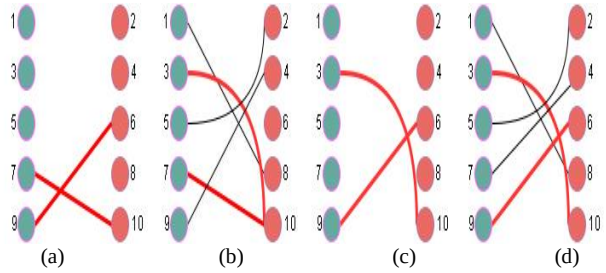

Fig. 8. The processing example of the bad edges: a) bad edges from Fig. 1(a), b) adjacent
edges after inserting the bad edge in Fig. 3(b), c) final bad edge set, d) final distribution of the bad edges

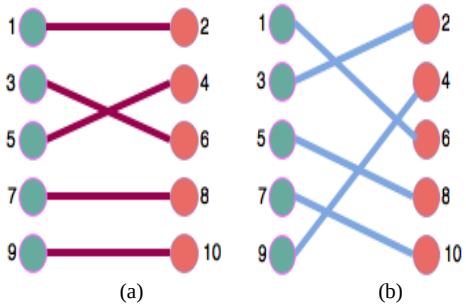

Fig. 9. The processing example of complete matching: (a) and (b) complete matching from procedure one, c) and d) complete matching from procedure one

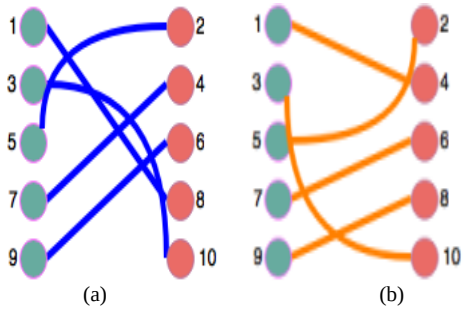

Fig. 10. The processing example of complete matching: (a) and (b) complete matching from procedure two, c) and d) complete matching from procedure two

The perfect edge coloring of Fig. 1(a) is found after combining all the complete matching in Fig. 9 and Fig. 10 that is shown in Fig.11.

Step 2: If it is unable to redistribute the bad edges in the incomplete matching then the same process can be followed for full matching and incomplete matching sequentially until re-distributed the bad edges into the D-matching's.

After *O (D.V)* times, the bad edge sets and the disjoint vertex sets are similar and re-distributed them according to the matching. After finding D-matching color the edges are colored into D-color. This part of the algorithm takes *O (D.V)* times. Because each bad edge has (2D-2) adjacent edges, so, to re-distribute each bad edge to a matching perfectly, it needs maximum (2D-2) iteration. So maximum *V* number of bad edges needs to iterate for *V*(2D-2) = 2D.V-2V = O (D.V)* times on average.

So, the overall time complexity of edge color matching algorithm is *O (D.V) +O (D.V) =O (D.V).*

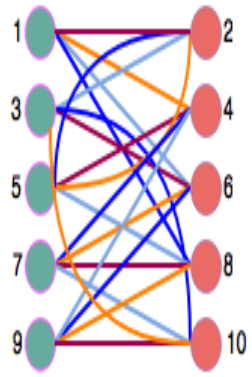

Fig. 11. The processing example of final color matching: the final edge coloring of Fig. 1 with perfect matching

IV. CASE STUDY

Case 1: In the following case study shows a bipartite graph that has 8 vertices, 16 edges and maximum degree 4. Here, the graph is 4-regular.

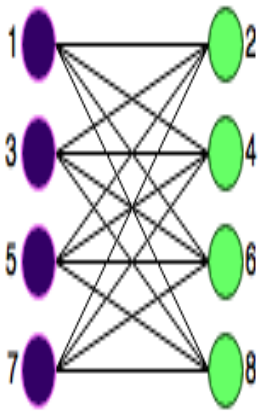

Fig. 12. The processing example of the case study: A regular bipartite multigraph with 8 vertices and degree 4

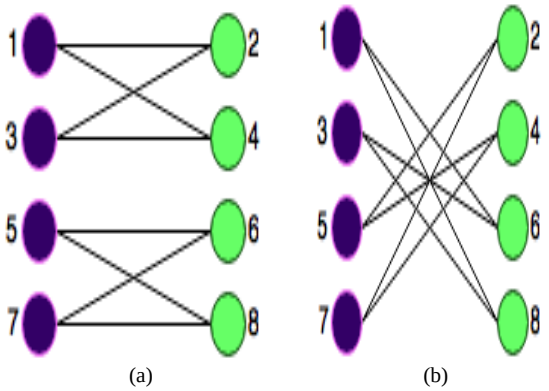

Fig. 13. The processing example of finding closed paths from Fig. 12: a) and b) the set of closed paths from Fig. 12

Fig. 13(a) and Fig. 13(b) shows two sets of the closed path from which four matching can be achieved that are shown in Fig. 14 and Fig. 15. Fig. 14 shows two matching that is achieved from the Fig. 13(a) closed path sets. And Fig. 15 shows two matching which is achieved from the Fig. 13(b) closed path sets.

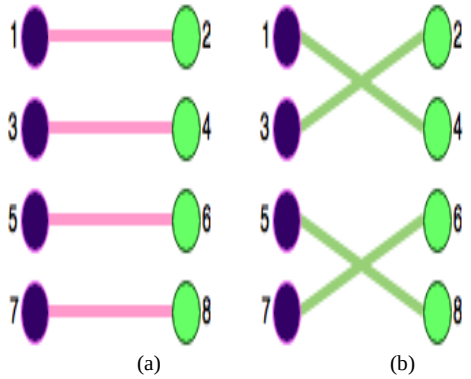

Fig. 14. Processing example of matching found: a) and b) matching from Fig. 13(a)

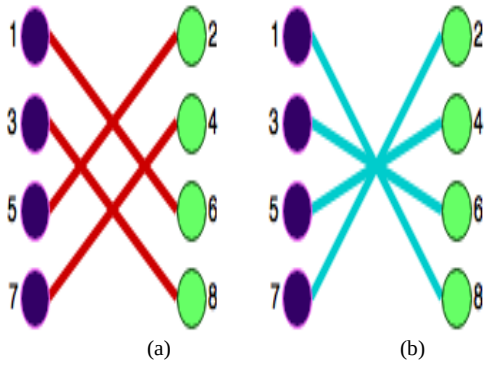

Fig. 15. The processing example of matching: a) and b) matching from Fig. 13(b)

After getting these four matching shown in Fig. 14 and Fig. 15 color the edges of each matching with separate colors to represent the edge color matching shown in Fig. 16.

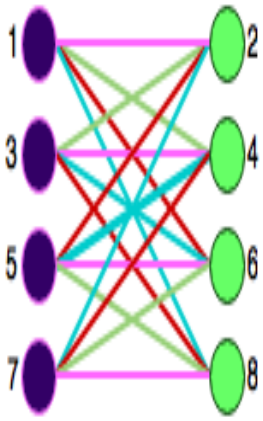

Fig. 16. Processing example final color matching: the final edge coloring of Fig. 11 with perfect matching

Case 2: In the following case study shows a bipartite graph that has 10 vertices, 20 edges and maximum degree 4 with 1 multiple edges.

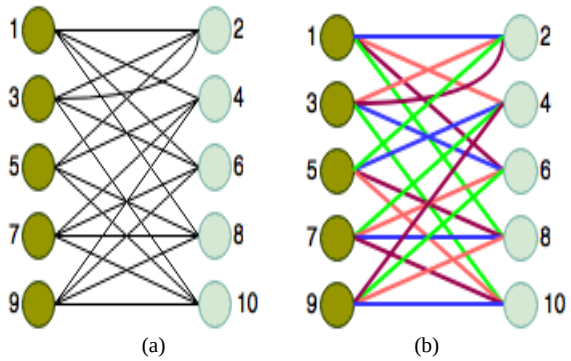

Fig. 17. The edge coloring of with perfect matching: a) input graph and b) final edge coloring with perfect matching

Case 3: This is a regular bipartite graph with no multiple edges. This graph consists of 20 edges, 10 vertices and 4 edges in every vertex, i.e., degree 4.

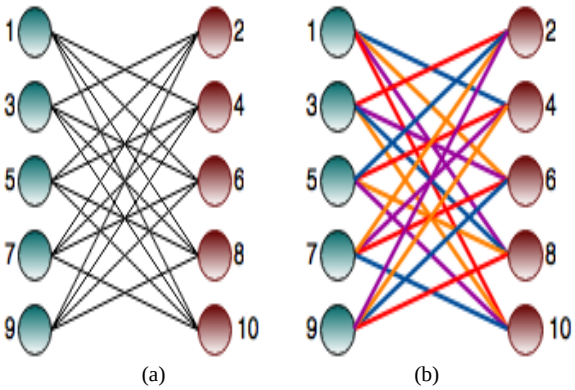

Fig. 18. The edge coloring of with perfect matching: a) input graph and b) final edge coloring with perfect matching

V. Experimental Results

In this paper, four different types of bipartite graph including multi-graphs are used to verify the proposed edge color matching algorithm. The first graph is used in the processing example and describes the proposed algorithm step by step. Other three graphs are shown in the case study, where case 1 is explained shortly with processing example. Case 2 and case 3 are shows the output only. The experimental results run time of procedure 1 that is described in Theorem 2 for the four types of the regular bipartite graph is shown in Table I. Accordingly, the experimental results run time of procedure 2 that is describes in Theorem 2 for the four types of the regular bipartite graph is shown in Table II. From the experimental result in Table I and Table II it is seen that total runtime is less than O (D.V).

TABLE. I. The Experimental Runtime for Procedure 1 Theorem 2

Input Graph	Number of vertices (V)	Degree of the graph (D)	Runtime for procedure 1
Processing example (Fig. 1(a))	8	4	16
Case 1 (Fig. 11(a))	10	4	33
Case 2 (Fig. 14(a))	10	4	34
Case 3 (Fig. 15(a))	10	4	26

TABLE. II. The Experimental Runtime for Procedure 2 Theorem 2

Input Graph	Number of vertices (V)	Degree of the graph (D)	Runtime for procedure 1
Processing example (Fig. 1(a))	8	4	0
Case 1 (Fig. 11(a))	10	4	4
Case 2 (Fig. 14(a))	10	4	4
Case 3 (Fig. 15(a))	10	4	14

VI. Conclusions

This paper has been presented an algorithm of edge coloring for finding perfect matching. This paper considered the regular bipartite multigraphs. To prove the algorithm two theorems for edge coloring is considered. The first theorem shows the perfect matching of a regular bipartite graph and the second theorem shows proper edge coloring of a D-regular bipartite multigraph. The overall time complexity of the proposed edge color matching algorithm is O *(D.V) +O (D.V = O (D.V)* times. The algorithm reduces overall time complexity. Experimental results show that the total runtime is less than *O (D.V)* time. A graph with large vertex cannot be considered with the proposed theorem. In future, this work will be extended to develop an algorithm in order to solve the large volume of the graph.

References

[1] J. Bensmail, A. Lagoutt, & P. Valicov,outt, & P. Valicov, "Strong edge-coloring edge coloring with perfect matching of (3, Δ)-bipartite graphs," *Discrete Mathematics*, Vol. 339, No. 1, pp. 391-398, 2016.

[2] H. A. Razak, Z. Ibrahim, and N.M. Hussin," Bipartite graph edge coloring approach to course timetabling," In 2010 *International Conference on Information Retrieval & Knowledge Management, (CAMP),* pp. 229-234, IEEE, March, 2010.

[3] C. J. Casselgren and B. Toft "One-sided interval edge-colorings of bipartite graphs," *Discrete Mathematics*, Vol. *339*, pp. 2628-2639, 2016.

[4] A. Schrijver, "Bipartite Edge Coloring in O(Δm) Time, **"***SIAM Journal on Computing*," Vol. 28, No. 3, pp. 841-846, 1998.

[5] R.Lang, O.Schaudt, and M. Stein, "Partitioning 3-edge-coloured complete bipartite graphs into monochromatic cycles," *Electronic Notes in Discrete Mathematics*, Vol. 49, pp. 787-794, 2015.

[6] G. J. Puleo, "Complexity of a disjoint matching problem on bipartite graphs," *Information Processing Letters*, 2016.

[7] J. Engbers and D. Galvin, "H-colouring bipartite graphs, " *Journal of Combinatorial Theory, Series B*, Vol. 102, No. 3, pp. 726-742, 2012.