An Enumeration Algorithm for the Edge Coloring Problem on Bipartite Graphs

Yasuko MATSUI[1] and Tomomi MATSUI[2]

[1] Tokyo Metropolitan University , Tokyo 192-03, Japan
[2] University of Tokyo, Tokyo 113, Japan

METR 95–09 (OCTOBER 1995)

Abstract. In this paper, we propose an algorithm for finding all the edge colorings in bipartite graphs. Our algorithm requires $O(m \log \Delta + K \min\{n^2 + m, m \log \Delta\})$ time and $O(m\Delta)$ space, where n denotes the number of vertices, m denotes the number of edges, Δ denotes the number of maximum degree, and K denotes the number of edge colorings.

1 Introduction

The finding of all objects that satisfy a specified property is a fundamental problem in combinatorics, computational geometry, and operations research. This paper deals with the edge coloring problem in bipartite graphs. There are many applications for the edge coloring problem. For example, scheduling problems and timetable problems are discussed in [1, 3, 8, 14]. In this paper, we propose an algorithm for finding all the minimum edge colorings in bipartite graphs.

Let us consider a bipartite graph $B = (U, V, E)$ with vertex sets U, V and edge set E. We denote $|U \cup V|$ by n, $|E|$ by m and Δ denotes the maximum degree of any vertex in $U \cup V$. In this paper, we allow a graph with parallel edges and assume that $n \leq m$.

An *edge coloring* of a graph associates a color with each edge in the graph in such a way that no two edges of the same color have a common endpoint. A *minimum edge coloring* is an edge coloring which uses the fewest number of colors as possible. The edge coloring problem finds a minimum edge coloring. In 1916, König proved the following theorem [10].

Theorem 1. *Let B be a bipartite graph and let Δ be the maximum degree of any vertex. Then a minimum edge coloring of B uses exactly Δ colors.*

Cole and Hopcroft proposed an algorithm for finding a minimum edge coloring in a bipartite graph [2]. Their algorithm finds a minimum edge coloring in $O(m \log \Delta)$ time.

Recently, we proposed an algorithm for finding all the minimum edge colorings in bipartite graphs [13], which requires $O(Knm)$ time and $O(nm^2)$ space, where K denotes the number of minimum edge colorings. However, our new algorithm is more efficient in time bound and space complexity. Our algorithm

requires $O(m \log \Delta + K \min\{n^2 + m, m \log \Delta\})$ time and $O(m\Delta)$ space. So, our algorithm generates each additional edge coloring in $O(\min\{n^2 + m, m \log \Delta\})$ time.

2 Algorithm

In this section, we describe the main framework of our algorithm.

First, we give some definitions and notations. An edge subset $M \subseteq E$ is called a *matching* if each pair of edges in M are not incident with a common vertex. A matching is said to *cover* a vertex subset W, if every vertex in W is an endpoint of an edge in the matching. A degree of a vertex x, denoted by $degree(x)$, is the number of edges which are incident with x.

From Theorem 1, it is clear that a minimum edge coloring of a bipartite graph B corresponds to a set of Δ matchings which is a partition of the edge set E. In this paper, we identify a minimum edge coloring of B with a set of Δ matchings. For any edge e and any minimum edge coloring C, we denote $M(C,e)$ as a matching in C which contains the edge e. If B has exactly one minimum edge coloring, we say B is *uniquely edge colorable.* When the graph B has two distinct minimum edge colorings C and C', there exists an edge $e \in E$ satisfying that $M(C,e) \neq M(C',e)$. A matching M of B is called a *feasible matching,* when there exists a minimum edge coloring including M. Theorem 1 directly implies the following lemma.

Lemma 2. *A matching M of B is feasible if and only if M covers every vertex of B whose degree is equal to Δ.*

For any edge e, $\mathcal{F}(B,e)$ denotes a set of all feasible matchings of B including the edge e. For any edge e and for any minimum edge coloring C, it is clear that $M(C,e) \in \mathcal{F}(B,e)$. Given a feasible matching M of B, $\mathcal{C}(B,M)$ denotes the set of all minimum edge colorings of B containing the matching M. Clearly, the family $\{\mathcal{C}(B,M) | M \in \mathcal{F}(B,e)\}$ is a partition of all minimum edge colorings of B. When we have a feasible matching $M \in \mathcal{F}(B,e)$, the problem to enumerate all minimum edge colorings in $\mathcal{C}(B,M)$ is reduced to the problem to enumerate all the minimum edge colorings in the bipartite graph $B \setminus M \equiv (U,V,E \setminus M)$. So, we can divide the problem to enumerate all the minimum edge colorings of B into subproblems. This idea implies the following algorithm.

Algorithm *main*(B)

A1: **begin**
A2: set $C := \emptyset$
A3: find a minimum edge coloring C_t of B
A4: call *find_all_colorings*(B, C_t, C)
A5: **end**

Subprocedure *find_all_colorings*(B', C'_t, C')

B1: **begin**
B2: **if** C'_t is the unique minimum edge coloring of B' **then** output $C' \cup C'_t$
B3: **else**
B4: **begin**
B5: choose an edge e of B'
B6: generate all the matchings in $\mathcal{F}(B', e)$
B7: **for** each $M \in \mathcal{F}(B', e)$ **do**
B8: **begin**
B9: find a minimum edge coloring C'' of $B' \setminus M$
B10: *find_all_colorings*$(B' \setminus M, C'', C' \cup \{M\})$
B11: **end**
B12: **end**
B13: **return**
B14: **end**

To complete this algorithm, we must solve the following three problems.

(1) The problem to determine whether a given graph is uniquely edge colorable or not at Line **B2**.
(2) The problem to choose an appropriate edge at Line **B5**.
(3) The problem for finding all the matchings in $\mathcal{F}(B', e)$ at Line **B6**.

In the rest of this paper, we discuss the above problems.

3 Uniquely edge colorable bipartite graph

In this section, we discuss the problems (1) and (2) described in the previous section.

When a given graph B has two distinct minimum edge colorings C and C', there exists an edge e satisfying that $M(C, e) \neq M(C', e)$. In our algorithm, we choose such an edge e at Line B5. Since $M(C, e) \neq M(C', e)$, the set $\mathcal{F}(B, e)$ includes at least two feasible matchings. Therefore, if we choose the edge e at Line B5, the original enumeration problem is divided into at least two subproblems. From the above discussion, we need to solve the following problem in our algorithm.

Subproblem `uniquely-colorable`(B, C)
Input: a bipartite graph B and a minimum edge coloring C in B
Output: an edge e satisfying that $M(C, e) \neq M(C', e)$, if B has a minimum edge coloring C' different from C; else, say "uniquely edge colorable"

There are some previous studies on uniquely edge colorable graphs [9, 12]. In our algorithm, the following theorem plays an important role.

Theorem 3. *A connected bipartite graph $B = (U, V, E)$ is not uniquely edge colorable if and only if there exist two vertices x and y $(x \neq y)$ such that $degree(x) \geq 3, degree(y) \geq 2$.*

Proof. If the degree of each vertex is less than 3, then the graph B is a path or a cycle, and so B is uniquely edge colorable. Now suppose that there exists a vertex x such that $degree(x) \geq 3$ and the degrees of any other vertices are less than 2. Then the graph B becomes a star and so B is uniquely edge colorable.

Next we show the inverse implication. Assume that there exist two vertices $x, y (x \neq y)$ such that $degree(x) \geq 3$ and $degree(y) \geq 2$. Let v be a vertex with $degree(v) = \Delta \geq 3$. If every vertex adjacent to v is a leaf vertex, then B becomes a star and it is a contradiction. So there exists a vertex u adjacent to v such that $degree(u) \geq 2$. We denote the edge connecting u and v by e. Since $degree(u) \geq 2$, there exists an edge $e'_1 \neq e$ incident with u. Let C be a minimum edge coloring of B. Here we denote the feasible matching $M(C, e'_1)$ by M'. Since the degree of v is equal to Δ, the matching M' contains an edge e'_2 which is incident with v. The property $degree(v) \geq 3$ implies that there exists an edge e'' incident to v satisfying that $e'_2 \neq e'' \neq e$. The matching $M(C, e'')$ is denoted by M''. Let $\widetilde{B}$ be the graph induced by the edge set $M' \cup M''$. Clearly, each component of $\widetilde{B}$ is a path or a cycle and the component including the vertex v is not an edge. If $\widetilde{B}$ is disconnected, then $\widetilde{B}$ is not uniquely edge colorable and it directly implies that the original graph B is not uniquely edge colorable. Now consider the case that $\widetilde{B}$ is connected. Then $\widetilde{B}$ contains a path P connecting v and u and including the edges e'_1 and e'_2. The path P and the edge e forms a cycle, which is denoted by C. Obviously, the cycle C is an alternating cycle with respect to the feasible matching M'. Let M^* be the symmetric difference of C and M'. Then M^* is also a feasible matching of B and M^* is not contained in C. Lemma 2 directly implies that there exists a minimum edge coloring C^* which contains the feasible matching M^*. Since $M^* \notin C$, the minimum edge coloring C^* is different from C and so B is not uniquely edge colorable.

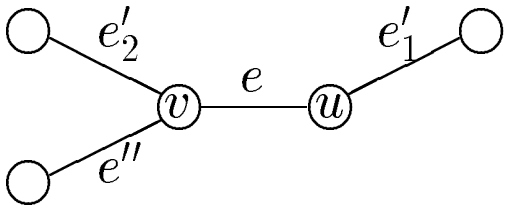

Fig. 1. Two vertices u and v such that $degree(u) \geq 2, degree(v) = \Delta \geq 3$.

The above theorem directly implies the following.

Corollary 4. *A bipartite graph B is uniquely edge colorable if and only if B is a matching, a cycle, a path, or a star.*

From Theorem 3, it is clear that we can determine whether a given bipartite graph is uniquely edge colorable or not by checking the degree of every vertex. The proof of Theorem 3 gives an idea for solving the subproblem `uniquely-colorable`(B,C). When the subgraph $\widetilde{B}$ defined in the proof is disconnected, we can output arbitrary edge in $\widetilde{B}$ as a solution of the subproblem `uniquely-colorable`(B,C). If $\widetilde{B}$ is connected, we output the edge e connecting the vertices u and v. So, the above procedure solves the subproblem in $O(n+\Delta)$time, when we know the degree of each vertex.

4 Finding all feasible matchings

In this section, we consider the problem to enumerate all the feasible matchings which contain a specified edge e.

Our algorithm partitions the set of feasible matchings including e into two subsets iteratively. More precisely, given two distinct feasible matchings M and M' in $\mathcal{F}(B,e)$, we choose an edge $f \in (M \cup M') \setminus (M \cap M')$ and partition the set $\mathcal{F}(B,e)$ into two subsets $\mathcal{M} = \{M'' \in \mathcal{F}(B,e) | f \in M''\}, \mathcal{M}' = \{M'' \in \mathcal{F}(B,e) | f \notin M''\}$. This partition implies that $M \in \mathcal{M}$ and $M' \in \mathcal{M}'$. Clearly, the recursive application of the above procedure constructs a binary tree of subproblems. By using depth first rule, we can find all the feasible matchings without repetition. Due to solve the above subproblem, it is very natural to consider the following subproblem.

Subproblem `matching-unique` (B, M, I, D)
Input: a bipartite graph B, a feasible matching M of B and a set of edges I, D which satisfy the conditions that $M \supseteq I, M \cap D = \emptyset$.
Output: a feasible matching M' satisfying that $M' \supseteq I, M' \cap D = \emptyset$ and $M \neq M'$, if one exists; else, say "none exists".

We can drop the conditions $e \in M$ and $e \in M'$ by setting I with $e \in I$.

In the following, we describe an algorithm for solving the above subproblems. From Lemma 2, we can replace the condition that M and M' is a feasible matching of B with the condition that M and M' cover a given vertex subset W. If we delete all edges in D and all vertices covered by I (and edges incident with the vertices), the subproblem `matching-unique` is transformed to the following problem.

Subproblem `find-matching` (B, W, M)
Input: a bipartite graph B, a vertex subset W of B and a matching M which covers W.
Output: a matching M' which differs from M and covers W, if one exists; else, say "none exists"

The following theorem provides an idea for solving the subproblem `find-matching`,

Lemma 5. *Let $B = (U, V, E)$ be a bipartite graph, W a vertex subset and M a matching which covers W. There exists a matching M' which differs from M and covers W if and only if either (1) or (2) holds.*
(1) There exists an alternating cycle with respect to M.
(2) There exists an alternating path with respect to M which connects two vertices not in W.

Proof. One direction is trivial. For the other direction, suppose that there exists a matching M' which covers W and differs from M. Consider the graph B' induced by the edges $(M \cup M') \setminus (M \cap M')$. In this graph, each connected component is an alternating path or an alternating cycle with respect to M. When the graph B' contains an alternating cycle, we have done. Consider the case that B' has an alternating path. Then, the terminal vertices of the path are covered by one matching and not by another matching. If a terminal vertex is contained in W, it contradicts our assumption that each vertex in W is covered by both M and M'. So, the alternating path connects two vertices not in W.

From now, we explain how to solve `find-matching`(B, W, M). Let $G'(B, M)$ be a directed graph obtained from B by directing the edges in $E - M$ from U to V, and the edges in M from V to U. Each directed elementary cycle in $G'(B, M)$ corresponds to an alternating cycle with respect to B and M. The inverse implication also holds. The depth first search method finds a directed elementary cycle in a given directed graph(see [11]).

Next, we describe how to find an alternating path with respect to M which connects two vertices not in W. Let U''(respectively V'') be a set of vertices in U(respectively V) which are covered by M. A directed graph $G''(B, M)$ is a graph obtained from B by directing edges in $E - M$ from U to V, the edges in M from V to U, add the source vertex s and the edges from s to $(V'' \setminus W) \cup (U \setminus U'')$, and add the sink vertex t and the edges from $(V \setminus V'') \cup (U'' \setminus W)$ to t. Each directed s-t path in $G''(B, M)$ corresponds to an alternating path with respect to M which connects two vertices not in W. The inverse implication also holds. The depth first search method finds a directed elementary s-t path [11].

From the above, we can solve the subproblem `matching-unique` in $O(n+m)$ time and $O(n+m)$ space by employing the depth first search method [11] (where n denotes the number of vertices and m denotes the number of edges in B).

Our algorithm generates the feasible matchings by solving a sequence of subproblems iteratively. When we apply the problem dividing procedure described before, we obtain a binary tree structure of subproblems. At any inner node (non-leaf node) of the binary tree structure, the corresponding subproblem finds a new matching and a given enumeration problem is divided into two subproblems. At every leaf node of the binary tree structure, the corresponding subproblem says "none exists" and we output the current feasible matching. So, the number of nodes of the binary tree structure is equal to $2|\mathcal{F}(B, e)| - 1$. Thus, the total time complexity for finding all the feasible matchings in $\mathcal{F}(B, e)$ is $O((n + m)|\mathcal{F}(B, e)|)$, when we have one matching in $\mathcal{F}(B, e)$. Since we solve the subproblems recursively, we traverses the binary tree structure by using the depth first rule. So, the space requirement becomes $O(n + m)$.

5 Complexity

Finally, we discuss the computational complexity and memory requirements of our algorithm for finding all the minimum edge colorings.

At Line A3 of the algorithm *main*, we employ Cole and Hopcroft's method [2] and find a minimum edge coloring in $O(m \log \Delta)$ time. In Section 3, we proposed a method to choose an edge e at Line B5 such that $\mathcal{F}(B', e)$ has at least two feasible matchings. Thus, the original enumeration problem is divided into at least two subproblems at Line B7. The algorithm *main* calls the subprocedure *find_all_colorings* recursively and generates a tree structures of the subprocedures. At every leaf node of the tree structure, we output a minimum edge coloring. At any inner node, we divide a given enumeration problem into at least two subproblems. Thus, the number of inner nodes of the tree structure is less than the number of minimum edge colorings in the original graph. So, the algorithm *main* calls the subprocedure *find_all_colorings* at most $2K - 1$ times, where K denotes the number of minimum edge colorings in the original graph. The computational time required at Lines B1-B5,B7,B8,B10-B14 is bounded by $O(m)$. At Line B6, we can generate all the feasible matchings in $\mathcal{F}(B', e)$ in $O(m|\mathcal{F}(B', e)|)$ time. Clearly, the subprocedure *find_all_colorings*(B', C'_t, C) calls the subprocedure *find_all_colorings* $|\mathcal{F}(B', e)|$ times at Line B10. Thus, the total time required at Line B6 is bounded by $O(mK)$. From the above, the total computational effort required at Lines B1-B8,B10-B14 is bounded by $O(mK)$. At Line B9, we need to find a new coloring of the graph $B' \setminus M$. If we employ Cole and Hopcroft's method, it requires $O(m \log \Delta)$ time. However, we have an edge coloring C'_t of B'. Then the set $\{M' \setminus M : M' \in C'_t\}$ is an edge coloring of $B' \setminus M$ using Δ' colors where Δ' is the maximum degree of any vertex in B'. Since M is a feasible matching of B', the maximum degree of any vertex in $B' \setminus M$ is $\Delta' - 1$. If we employ König's color flipping method [10], we can find an edge coloring of $B' \setminus M$ from C'_t in $O(n^2 + m)$ time. Thus, the time complexity of Line B9 is bounded by $O(\min\{n^2 + m, m \log \Delta\})$. From the above, the total time complexity of the algorithm *main* is bounded by $O(m \log \Delta + K \min\{n^2 + m, m \log \Delta\})$.

At last, we discuss the space requirement. If we generate all the feasible matchings in $\mathcal{F}(B', e)$ at Line B6, then we need a huge amount of memory space, since $\mathcal{F}(B', e)$ contains exponential number of matchings in the worst case. However, we can reduce the space requirement easily. At Line B6, we execute the matching enumeration algorithm proposed in Section 2 and at each time instance when an additional matching is obtained, we give a pause to the matching enumeration algorithm and execute the loop B8-B11. Since the matching enumeration algorithm requires $O(m)$ space, the space complexity of the subprocedure *find_all_colorings* is bounded by $O(m)$. Clearly, the height of the tree structure of the subprocedures generated by the algorithm *main* is less than or equal to Δ. Since the algorithm *main* traverses the tree structure by using the depth first rule, the total memory requirement is bounded by $O(m\Delta)$.

From the above discussions, our algorithm requires $O(m \log \Delta + K \min\{n^2 + m, m \log \Delta\})$ time and $O(m\Delta)$ space. The bottleneck of the time complexity is Line B9. If we have an $O(T)$ time algorithm for finding an edge coloring

using Δ colors (i.e., a minimum coloring) of a bipartite graph B from an edge coloring using $\Delta + 1$ colors, the time complexity of our algorithm is reduced to $O(m \log \Delta + K(m + T))$ time.

References

1. Bondy,J.A., Murty,U.S.R.: Graph theory with applications. North-Holland (1976)
2. Cole,R., Hopcroft,J.: On Edge Coloring Bipartite Graphs. SIAM J.Comput. **11** (1982) 540-546
3. Dempster,M.A.H.: Two algorithms for the time-table problem, Combinatorial Mathematics and its Applications(ed. D.J.A.Welsh). Academic Press New York (1971) 63-85
4. Fukuda,K., Matsui,T.: Finding All the Minimum Cost perfect Matchings in bipartite Graphs. Networks **22** (1992) 461-468
5. Fukuda,K., Matsui,T.: Finding All the Perfect Matchings in Bipartite Graphs. Appl. Math. Lett. **7** 1 (1994) 15-18
6. Gabow,H.N.: Using Euler Partitions to Edge Color Bipartite Multigraphs. Information J. Comput. and Information Sciences **5** (1976) 345-355
7. Gabow,H.N., Kariv,O.: Algorithms for Edge Coloring Bipartite Graphs and Multigraphs. SIAM J. Comput. **11** (1982) 117-129
8. Gonzalez,T., Sahni,S.: Open Shop Scheduling to Minimize Finish Time. J. ACM. **23** (1976) 665-679
9. Greenwell,D.L., Kronk,H.V.: Uniquely Line Colorable Graphs. Canad. Math. Bull. **16** (1973) 525-529
10. König,D.: Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlhere. Math. Ann. **77** (1916) 453-465
11. Tarjan,R.E.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1** (1972) 146-160
12. Thomason,A.G.: Hamiltonian Cycles and Uniquely Edge Colourable Graphs. Ann. Discrete Math. **3** (1978) 259-268
13. Yoshida,Matsui.Y., Matsui,T.: Finding All the Edge Colorings in Bipartite Graphs. T.IEE. Japan 114-C **4** (1994) 444-449 (in Japanese)
14. de Werra,D.: On some combinatorial problems arising in scheduling. INFOR. **8** (1970) 165-175