Discrete Methods in Informatics October 3, 2005

Lecture 1 (Edge-Coloring)

Lecturer: *Satoru Iwata* *Scribe*: *Yusuke Kobayashi*, *Kenjiro Takazawa*

In this course, we will cover some topics in combinatorial optimization. The textbook is [3].

1 Edge-Coloring

Let $G = (V, E)$ be an undirected graph. An *edge-coloring* of G is coloring every edge so that every pair of adjacent edges have different colors. In other words, a set of edges in the same color corresponds to a matching. We consider to find an edge-coloring with the minimum number of colors, which we call a *minimum edge-coloring*.

Let γ be the minimum number of colors, and $d(v)$ be the degree of $v \in V$. Then, we have $\gamma \geq d(v),\ \forall v \in V$. In other words, $\gamma \geq d^*$ holds, where $d^* = \max\{d(v) \mid v \in V\}$. This inequality gives rise to the question of whether the equality $\gamma = d^*$ holds or not. In fact, this equality does not always hold. For example, $\gamma = 3$, and $d^* = 2$ in Fig. 1.

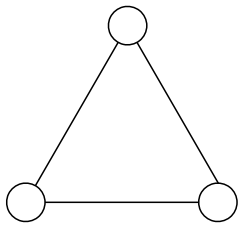

Figure 1: A counterexample to $\gamma = d^*$.

In bipartite graphs, however, it is known that the equality holds.

Theorem 1 (Kőnig, 1916)**.** *If G is bipartite, then $\gamma = d^*$.*

Proof. Let $M_1, M_2, \ldots, M_{d^*}$ be disjoint matchings whose total number of edges are maximum. If an edge e belongs to M_i, we say that e is colored in i. It is enough to show that $E = M_1 \cup M_2 \cup \cdots \cup M_{d^*}$.

Assume that there exists an edge $e = (u, v)$ in $E \setminus (M_1 \cup M_2 \cup \cdots \cup M_{d^*})$. Since there exist at most $d^* - 1$ edges incident to u except e, there is a color i in $1, 2, \ldots, d^*$ such that no edge incident to u is colored in i. Similarly, there is a color j such that no edge incident to v is colored in j.

- Suppose $i = j$. Then, we can color e in i, which contradicts the maximality of $|M_1 \cup M_2 \cup \cdots \cup M_{d^*}|$.
- Suppose $i \neq j$. Let G' be a subgraph of G whose edge set is $M_i \cup M_j \cup \{e\}$. In G', since the degree of each vertex is at most 2, each component is a path or a cycle.
  - Assume that e is contained in a cycle C. In C, the edges in M_i and M_j alternately lie, and one of the adjacent edges to e is in M_i, the other in M_j. Hence, the cycle C is odd, which contradicts that G is bipartite.

– Assume e is contained in a path P. Then, we can change M_i and M_j so that they cover all edges in P, which contradicts the maximality of $|M_1 \cup M_2 \cup \cdots \cup M_{d^*}|$.

□

This proof implies that we can find an edge-coloring with d^* colors in $O(nm)$ time, where n and m denote the number of vertices and edges, respectively. There are some more efficient algorithms for the problem such as Schrijver's algorithm in $O(md^*)$ time [2] and the algorithm in $O(m \log d^*)$ time by Cole, Ost, and Schirra [1].

2 Algorithms

In this section, we describe Schrijver's algorithm to find a minimum edge-coloring in a bipartite graph.

We say that a graph is *k-regular* if the degree of every vertex is k. The algorithm is based on the following proposition.

Proposition 2 (Hall's theorem)**.** *Any k-regular bipartite graph has a perfect matching.*

Proof. In a k-regular bipartite graph G, Theorem 1 tells us that there exists a set of disjoint matchings $M_1, \ldots, M_k$ which cover all edges. Then $\forall v \in V$, $d(v) = k$ means that v is incident to an edge in M_i for all i $(1 \leq i \leq k)$, which implies that M_i is a perfect matching. □

Given any bipartite graph $G = (U, V; E)$, we can construct a d^*-regular bipartite graph G^* as follows:

1. While there are more than one vertices in U (resp. V) whose degree is at most $d^*/2$, contract two of them so that no more than one such vertex exists in U (resp. V).

2. Add some dummy vertices and edges so that the obtained graph is d^*-regular (multiple edges are allowed).

If we can find a minimum edge-coloring in G^*, then we can also find that in G.

Here we describe an algorithm to find a minimum edge-coloring of a d^*-regular bipartite graph. A simple method consists of the iterations of finding a perfect matching and removing it. This method is, however, not efficient. To find an minimum edge-coloring more efficiently, we consider the following divide and conquer method:

1. When d^* is odd, reduce the problem to that in a $(d^* - 1)$-regular graph by removing a perfect matching.

2. When d^* is even, find an Euler cycle. Then, take every other edge along the Euler cycle to obtain two $(d^*/2)$-regular subgraphs, and reduce the problem to those in the two $(d^*/2)$-regular graphs.

In this algorithm, it is important to find a perfect matching in a k-regular bipartite graph efficiently.

The following algorithm by Schrijver [2] finds a perfect matching in a k-regular bipartite graph in $O(mk)$ time. We can find a minimum edge-coloring of a d^*-regular bipartite graph in $O(md^*)$ time using Algorithm 3.

Algorithm 3. Initialization: $w(e) := 1, \ \forall e \in E$.

Iteration: Let $E^* = \{e \in E \mid w(e) > 0\}$. If a cycle C exists in E^*, then divide the edges in C into two matchings M, N such that $w(M) \geq w(N)$, and update w as

$$w(e) := \begin{cases} w(e) + 1 & (\text{for } e \in M), \\ w(e) - 1 & (\text{for } e \in N). \end{cases}$$

Theorem 4. *Algorithm 3 finds a perfect matching in a k-regular bipartite graph in $\mathrm{O}(mk)$ time.*

Proof. Firstly, we show that E^* is a perfect matching when the algorithm terminates. The sum of the weight w of the edges incident to each vertex is k in the initialization, and is not changed in the algorithm. Note that each weight is always nonnegative.

When the algorithm terminates, the subgraph whose edge set is E^* is a forest. Assume v is a leaf of the forest (i.e. a single edge $e = (u, v)$ is incident to v). Then $w(e) = k$ and so u has no incident edges in E^* except e, which implies E^* is a matching. Since each edge has at least one incident edge in E^*, E^* is a perfect matching.

Next we consider the complexity of the algorithm. In each iteration, the value $\sum_{e \in E} w(e)^2$ increases by

$$\sum_{e \in M} ((w(e) + 1)^2 - w(e)^2) + \sum_{e \in N} ((w(e) - 1)^2 - w(e)^2) = 2w(M) + |M| - 2w(N) + |N| \geq |C|.$$

The value $\sum_{e \in E} w(e)^2$ is equal to m in the initialization, and mk in the end of the algorithm. We require $\mathrm{O}(|C|)$ time to find a cycle C, and update w. Hence the algorithm runs in $\mathrm{O}(mk)$ time. □

References

[1] R. Cole, K. Ost, S. Schirra: Edge coloring bipartite multigraphs in $\mathrm{O}(E \log D)$ time, *Combinatorica*, **21** (2001), pp. 5–12.

[2] A. Schrijver: Bipartite edge coloring in $\mathrm{O}(\Delta m)$ time, *SIAM Journal on Computing*, **28** (1998), pp. 841–846.

[3] A. Schrijver: *Combinatorial Optimization*. Springer-Verlag, 2003.