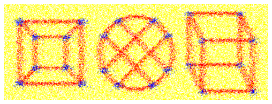
Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 12, no. 4, pp. 383–399 (2008)

An Efficient Algorithm for the Nearly Equitable Edge Coloring Problem

Xuzhen Xie[1] *Mutsunori Yagiura*[1] *Takao Ono*[1] *Tomio Hirata*[1]
Uri Zwick[2]

[1]Graduate School of Information Science
Nagoya University, Nagoya 464-8603, Japan
[2]Department of Computer Science
Tel Aviv University, Tel Aviv 69978, Israel

Abstract

An edge coloring of a multigraph is nearly equitable if, among the edges incident to each vertex, the numbers of edges colored with any two colors differ by at most two. It has been proved that the problem of finding a nearly equitable edge coloring can be solved in $O(m^2/k)$ time, where m and k are the numbers of edges and given colors, respectively. In this paper, we present a recursive algorithm that runs in $O\left(mn \log\left(m/(kn)+1\right)\right)$ time, where n is the number of vertices. This algorithm improves the best-known worst-case time complexity. When $k = O(1)$, the time complexity of all known algorithms is $O(m^2)$, which implies that this time complexity remains to be the best for more than twenty years since 1982 when Hilton and de Werra gave a constructive proof for the existence of a nearly equitable edge coloring for any multigraph. Our result is the first that improves this time complexity when m/n grows to infinity; e.g., $m = n^{\vartheta}$ for an arbitrary constant $\vartheta > 1$.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
July 2008	September 2008	October 2008	October 2008	November 2008

Published: December 2008

Article type: Regular paper

Communicated by: D. Wagner

This research was supported in part by Grant-in-Aid for Scientific Research from the Ministry of Education, Culture, Sports, Science and Technology of Japan. A preliminary version of this paper appeared in Proceedings of ISAAC [10].

E-mail addresses: sharryx@al.cm.is.nagoya-u.ac.jp (Xuzhen Xie) yagiura@nagoya-u.jp (Mutsunori Yagiura) takao@al.cm.is.nagoya-u.ac.jp (Takao Ono) hirata@is.nagoya-u.ac.jp (Tomio Hirata) zwick@tau.ac.il (Uri Zwick)

1 Introduction

Throughout this paper, the graphs considered will be multigraphs that allow multiple edges between vertices. Given a graph $G = (V, E)$ with n vertices and m edges and a color set $\mathcal{C} = \{1, 2, \ldots, k\}$, the *nearly equitable edge coloring* is an assignment of given colors to edges in G such that, among the edges incident to each vertex, the numbers of edges colored with any two colors differ by at most two. The notion of the nearly equitable edge coloring was introduced in 1982 by Hilton and de Werra [1] (a simplified version [2] was published in 1994), who also proved that any graph has a nearly equitable edge coloring. Their proof is constructive and easily leads to an algorithm for finding such a coloring in $O(km^2)$ time. Later, Nakano et al. [4] showed an algorithm that runs in $O(m^2/k + mn)$ time. In 2004, Xie et al. [8] presented a more efficient algorithm, which improves the running time to $O(m^2/k)$ and moreover satisfies the following *balanced constraint*: The numbers of the edges colored with any two colors differ by at most one. We call their algorithm BalCol in the following, which stands for *Balanced Coloring*. Some randomized algorithms were also proposed in [9, 10].

It might seem more natural to consider colorings such that, among the edges incident to each vertex, the numbers of edges colored with any two colors differ by at most one; such a coloring is called an *equitable edge coloring*. However, not all graphs admit an equitable edge coloring [1, 2], and the problem of judging the existence of an equitable edge coloring is NP-complete. Consider the case with $k = \Delta$, where Δ is the maximum vertex degree of the given graph, then this problem is equivalent with the problem of determining the existence of a (usual) edge coloring with Δ colors, a well-known NP-complete problem [3]. Some sufficient conditions for an equitable edge coloring to exist have been investigated [1, 2, 5, 6, 7].

In this paper, we consider the nearly equitable edge coloring problem and present a recursive algorithm that runs in $O\left(mn \log\left(m/(kn) + 1\right)\right)$ time. The edge coloring obtained by this algorithm also satisfies the balanced constraint. We call the recursive algorithm RecCol, which stands for *Recursive Coloring*.

The time complexity of RecCol is better than the previous best known time bound $O(m^2/k)$ of algorithm BalCol. When $k = O(1)$, the time complexity of all known algorithms becomes $O(m^2)$, which implies that the time complexity derived from the proof in [1] remains to be the best for more than twenty years. RecCol is the first algorithm that improves this time complexity for the case when $m/n \to \infty$ $(n \to \infty)$ holds; e.g., when $m = n^{\vartheta}$ ($\vartheta > 1$ is an arbitrary constant).

Algorithm RecCol uses a procedure that constitutes the core part of algorithm BalCol. We call this procedure ChkRec (*Check and Recolor*). Starting from a given edge coloring π, ChkRec repeats the following until π becomes nearly equitable: It checks whether the current edge coloring π is nearly equitable or not, and if π is not nearly equitable, it invokes an algorithm called Recolor to modify π. Let $E_\pi(i)$ denote the set of edges colored with i. Xie et al. introduced a potential Φ_π and showed that Recolor runs in

$O(|E_\pi(i) \cup E_\pi(j)|)$ time for relevant colors i and j and decreases Φ_π by at least one for each call to it. It is therefore preferable to construct a coloring π such that potential Φ_π is small and $|E_\pi(i)|$ is small for all $i \in \mathcal{C}$ before calling algorithm ChkRec to keep its computation time smaller.

When $|E| \leq kn$, algorithm RecCol uses algorithm BalCol directly to obtain a nearly equitable edge coloring π of G. Otherwise, it partitions E arbitrarily into two subsets E^{L} and E^{R} of about the same size, and recursively applies RecCol to both of them to obtain edge colorings π^{L} and π^{R} of E^{L} and E^{R}, respectively. It then constructs a coloring π of G by merging the two colorings π^{L} and π^{R} after permuting them. We show that the resulting edge coloring π satisfies $|E_\pi(i)| = O(m/k)$ for all colors $i \in \mathcal{C}$ and $\Phi_\pi = O(kn)$; these results lead to the new time complexity $O\left(mn \log\left(m/(kn) + 1\right)\right)$.

The paper is organized as follows. In Section 2, we give some definitions. Section 3 introduces the basic procedures in [8], which are necessary to understand the new algorithm. Section 4 explains the recursive algorithm. Finally, concluding remarks are in Section 5.

2 Preliminaries

Let V and E denote the sets of vertices and edges of graph G, respectively. The following definitions will be used throughout this paper.

- Let $n = |V|$ and $m = |E|$. Note that $m \geq n - 1$ holds for any connected graph.
- We denote the given color set by $\mathcal{C} = \{1, 2, \ldots, k\}$, where k is the number of given colors.
- We denote an edge coloring by a mapping π: $E \to \mathcal{C}$; i.e., if an edge $e \in E$ is colored with a color i, then $\pi(e) = i$.
- For each vertex $v \in V$, let $N(v)$ denote the set of edges incident to v in G and $d(v) = |N(v)|$ be its degree. Then, $d_\pi(v, i) = |\{e \in N(v) \mid \pi(e) = i\}|$ stands for the number of edges colored with i and incident to v, while $E_\pi(i) = \{e \in E \mid \pi(e) = i\}$ stands for the set of edges in E colored with i.
- For any subset $E' \subseteq E$, let $V_{E'}$ be the set of end vertices of edges in E'. Then let $G_{E'} = (V_{E'}, E')$.
- Let $V_\pi(i, j)$ be the set of end vertices of edges in $E_\pi(i) \cup E_\pi(j)$; i.e., $V_\pi(i, j) = \{v \in V \mid \exists e \in (N(v) \cap (E_\pi(i) \cup E_\pi(j)))\}$. Then let $G_\pi(i, j) = (V_\pi(i, j),\ E_\pi(i) \cup E_\pi(j))$ be the subgraph whose edges are $E_\pi(i) \cup E_\pi(j)$ and vertices are their end vertices.

The definitions of nearly equitable edge coloring introduced by Hilton and de Werra [1], and the *balanced constraint* introduced by Nakano et al. [4] are as follows.

Definition 1 ([1]) *Given a graph $G = (V, E)$ and a color set $\mathcal{C} = \{1, 2, \ldots, k\}$, the nearly equitable edge coloring π is an assignment of the given k colors to all the edges in G, such that for any vertex $v \in V$ and different colors $i, j \in \mathcal{C}$, $|d_\pi(v, i) - d_\pi(v, j)| \leq 2$.*

Definition 2 ([4]) *An edge coloring π satisfies the balanced constraint if $||E_\pi(i)| - |E_\pi(j)|| \leq 1$ holds for any two colors i and j.*

Without loss of generality, we assume that G is connected, $m > 0$ and $2 \leq k \leq m$.

3 Basic Procedures for Nearly Equitable Edge Coloring

In this section, we explain algorithm BalCol and basic procedures ChkRec and Recolor, and then summarize the results in [8]. Some basic properties and implementation issues, which were not mentioned in the previous papers, are also briefly introduced in this section. These are important for understanding the time complexity of algorithm RecCol of the next section. Note that the algorithms in this section are slightly modified from the original version to make the discussion clearer. Some implementation details and the proofs of the lemmas in this section are given in Appendix.

Algorithm BalCol initially assigns colors to all edges so that the resulting coloring satisfies the balanced constraint of Definition 2, e.g., by choosing an arbitrary order of edges and assigning colors $1, 2, \ldots, k, 1, 2, \ldots$ to them in this order (i.e., the first edge is assigned color 1, the second is assigned color 2, ..., the kth is assigned color k, the $(k+1)$st is assigned color 1, the $(k+2)$nd is assigned color 2, and so on). Then it calls algorithm ChkRec. Whenever the current coloring π has a vertex u that breaks the condition of Definition 1, algorithm ChkRec chooses two colors α and β with maximum and minimum $d_\pi(u, i)$, respectively, and calls algorithm Recolor to recolor those edges in $E_\pi(\alpha) \cup E_\pi(\beta)$. Algorithm Recolor first constructs an augmented graph $\hat{G} = (\hat{V}, \hat{E})$ by adding a vertex $\hat{v}$ and some edges to make $G_\pi(\alpha, \beta)$ connected and the degrees of all vertices even. It then finds an Euler circuit in $\hat{G}$ starting at the additional vertex $\hat{v}$ and colors the edges alternately with α and β along the Euler circuit, so that the resulting coloring π' is balanced with respect to the two colors, i.e., $||E_{\pi'}(\alpha)| - |E_{\pi'}(\beta)|| \leq 1$ holds after recoloring. The algorithms are formally described as follows.

Algorithm BalCol$(G, \mathcal{C})$
Input: a graph $G = (V, E)$ and a k-color set $\mathcal{C} = \{1, 2, \ldots, k\}$
Output: a nearly equitable edge coloring π for G

1. Let $\pi : E \to \mathcal{C}$ be an edge coloring that satisfies the balanced constraint.
2. Let $\pi \leftarrow$ ChkRec$(G, \mathcal{C}, \pi)$.
3. Output π and stop.

Algorithm ChkRec$(G, \mathcal{C}, \pi)$
Input: a graph $G = (V, E)$, a k-color set $\mathcal{C} = \{1, 2, \ldots, k\}$, and an edge coloring π
Output: a nearly equitable edge coloring π for G

1. **for** each $u \in V$ **do**
2. **while** there exist colors $i, j \in \mathcal{C}$ such that $|d_\pi(u, i) - d_\pi(u, j)| \geq 3$ **do**
3. For the vertex u, find two colors $\alpha, \beta \in \mathcal{C}$ satisfying
$d_\pi(u, \alpha) = \max_{i \in \mathcal{C}} (d_\pi(u, i))$
$d_\pi(u, \beta) = \min_{i \in \mathcal{C}} (d_\pi(u, i))$.
4. Call Recolor(G, α, β, π).
5. Output π and stop.

Algorithm Recolor(G, α, β, π)
Input: a graph $G = (V, E)$, colors α and β, and an edge coloring π
Task: modify the edge coloring π for $G_\pi(\alpha, \beta)$

1. Let $\hat{V} \leftarrow V_\pi(\alpha, \beta) \cup \{\hat{v}\}$ $(\hat{v} \notin V)$ and $\hat{E} \leftarrow E_\pi(\alpha) \cup E_\pi(\beta)$.
2. **for** each connected component H in $G_\pi(\alpha, \beta)$ **do**
3. **if** H has at least one odd-degree vertex **then**
4. For each odd-degree vertex v in H,
add into $\hat{E}$ an edge connecting v and $\hat{v}$.
5. **else**
6. **if** H has a vertex v such that $|d_\pi(v, \alpha) - d_\pi(v, \beta)| \geq 2$ **then**
Draw two parallel edges between v and $\hat{v}$,
and add them into $\hat{E}$.
7. **else**
Let v be an arbitrary vertex in H. Draw two parallel edges between v and $\hat{v}$, and add them into $\hat{E}$.
8. Let $\hat{G} \leftarrow (\hat{V}, \hat{E})$.
9. Let a sequence of edges $e_1, e_2, \ldots, e_l$ be an Euler circuit of $\hat{G}$ such that the tail of e_1 is $\hat{v}$. Then let $\hat{\pi}(e_t) \leftarrow \alpha$ if t is odd and $\hat{\pi}(e_t) \leftarrow \beta$ otherwise for all $t = 1, 2, \ldots, l$.
10. Let $\pi(e) \leftarrow \hat{\pi}(e)$ for all edges e in $G_\pi(\alpha, \beta)$, and stop.

To show that algorithm Recolor works properly (i.e., the algorithm is well-defined), it is sufficient to observe that the augmented graph $\hat{G}$ has an Euler circuit. Any graph (including $G_\pi(\alpha, \beta)$) has even number of odd-degree vertices; hence the degree of the newly added vertex $\hat{v}$ is even. Thus all vertices in $\hat{G}$ have even degrees, which implies that $\hat{G}$ has an Euler circuit.

To observe that algorithm ChkRec outputs a nearly equitable edge coloring for the given graph when it stops, the following lemma is essential.

Lemma 1 *For any coloring π and two colors $\alpha, \beta \in \mathcal{C}$, let π' be the coloring after invoking* Recolor(G, α, β, π). *Then we have $d_{\pi'}(v, \alpha) + d_{\pi'}(v, \beta) = d_\pi(v, \alpha) + d_\pi(v, \beta)$ and $|d_{\pi'}(v, \alpha) - d_{\pi'}(v, \beta)| \leq |d_\pi(v, \alpha) - d_\pi(v, \beta)|$ for all vertices $v \in V_\pi(\alpha, \beta)$.*

(The proof is given in Appendix.) From this lemma, we have

$$\max_{i\in\mathcal{C}} d_{\pi'}(v,i) \leq \max_{i\in\mathcal{C}} d_{\pi}(v,i) \tag{1}$$

$$\min_{i\in\mathcal{C}} d_{\pi'}(v,i) \geq \min_{i\in\mathcal{C}} d_{\pi}(v,i) \tag{2}$$

for any $v \in V$. This indicates that, once

$$\left|\max_{i\in\mathcal{C}} d_{\pi}(v,i) - \min_{i\in\mathcal{C}} d_{\pi}(v,i)\right| \leq 2 \tag{3}$$

is satisfied for a vertex $v \in V$, condition (3) will never be violated again for the vertex v throughout the remaining execution of ChkRec. Note that (3) is equivalent with

$$\forall i,j \in \mathcal{C},\ |d_{\pi}(v,i) - d_{\pi}(v,j)| \leq 2, \tag{4}$$

the condition in Definition 1. For each vertex $u \in V$ in Line 1 of ChkRec, whenever the "while" loop in Line 2 of ChkRec terminates, condition (4) holds with $v = u$, and this condition will never be violated again as mentioned above. Hence the coloring π output by ChkRec satisfies (4) for all vertices $v \in V$.

We now consider the running time of algorithm BalCol. As Line 1 takes $O(m)$ time, its running time is dominated by the running time of ChkRec. Note that, in ChkRec, algorithm Recolor only modifies the coloring for $G_\pi(\alpha,\beta)$, which can be extracted from G in $O(|E_\pi(\alpha) \cup E_\pi(\beta)|)$ time if appropriately implemented; e.g., by keeping, for each color $i \in \mathcal{C}$, a linked list of those edges colored with i.

Two colors α and β in Line 3 of ChkRec can be found in $O(1)$ time for each iteration if appropriately implemented, and additional time needed for maintaining data structures for this computation during the whole execution of ChkRec can be kept in $O(m+\xi)$, where ξ is the total number of calls to Recolor from ChkRec. Here is a brief sketch of an example of such data structures. For the current vertex u and for each $d = 0, 1, 2, \ldots, d(u)$, prepare a linked list of those colors $i \in \mathcal{C}$ satisfying $d_\pi(u,i) = d$, and keep pointers to the maximum and minimum d, respectively, among those d with a nonempty list. Then α and β can be found in the first cells of the linked lists identified by these two pointers. The linked lists can be constructed so that the cell for i in the linked list can be accessed in $O(1)$ time whenever a color i is specified. Then it takes $O(1)$ time to remove the cells with colors α and β from the current position and insert them to appropriate positions after executing Recolor. After modifying the positions of α and β, we may need to update the pointers to the maximum and minimum d with a nonempty list. Because of (1) and (2), these pointers are changed $O(d(u))$ times during the whole execution of the "while" loop for each vertex u; hence this part takes $O\left(\sum_{u\in V} d(u) + \xi\right) = O(m+\xi)$ time during the whole execution of ChkRec. Care should be taken in initializing the above data structure when ChkRec begins the computation for a new vertex u, because if this part is naively implemented, $\Omega(k)$ time may be spent for each u, which results in spending $\Omega(kn)$ time during the whole

computation of ChkRec. A key idea to avoid this is to reuse the list for $d = 0$ without constructing it from scratch. Then the initialization can be done in $O(d(u))$ time. This part is rather complicated and hence is explained in Appendix.

Based on the above discussion, we can conclude that the running time of ChkRec is decided by the running time of Recolor and the number of calls to Recolor. To analyze the number of calls to Recolor, Xie et al. introduced a potential Φ_π, which is defined as follows. For all vertices $v \in V$, let $\bar{d}(v) = \lfloor d(v)/k \rfloor$. Define

$$\begin{aligned} \Phi_\pi(v,i) &= \varphi^{(2)}_{\bar{d}(v)-1}\left(d_\pi(v,i)\right) \\ \Phi_\pi(v) &= \sum_{i\in\mathcal{C}} \Phi_\pi(v,i) \\ \Phi_\pi &= \sum_{v\in V} \Phi_\pi(v), \end{aligned}$$

where $\varphi^{(2)}_{\bar{d}(v)-1}\left(d_\pi(v,i)\right)$ is defined by

$$\varphi^{(b)}_a(x) = \max\{x-a-b, a-x, 0\} \tag{5}$$

with $x = d_\pi(v,i)$, $a = \bar{d}(v) - 1$ and $b = 2$. By definition, $\Phi_\pi \geq 0$ holds for any coloring π.

Lemma 2 ([8]) *For any colors $\alpha, \beta \in \mathcal{C}$ and a coloring π, let π' be the coloring after calling* Recolor(G, α, β, π). *Then* Recolor *runs in $O(|E_\pi(\alpha) \cup E_\pi(\beta)|)$ time, and the coloring π' satisfies $||E_{\pi'}(\alpha)| - |E_{\pi'}(\beta)|| \leq 1$.*

Lemma 3 ([8]) *Assume that there exist a vertex u and colors α, β such that $d_\pi(u,\alpha) \geq \bar{d}(u)+1$, $d_\pi(u,\beta) \leq \bar{d}(u)$ and $d_\pi(u,\alpha) - d_\pi(u,\beta) \geq 3$ for a coloring π, and let π' be the coloring after calling* Recolor(G, α, β, π). *Then $\Phi_{\pi'} \leq \Phi_\pi - 1$ holds.*

The initial edge coloring π of algorithm BalCol satisfies $||E_\pi(i)| - |E_\pi(j)|| \leq 1$ for any $i, j \in \mathcal{C}$, which implies that $|E_\pi(i)| = O(m/k)$ holds for all $i \in \mathcal{C}$. Lemma 2 implies that after invoking Recolor, the edge coloring π' also satisfies $||E_{\pi'}(i)| - |E_{\pi'}(j)|| \leq 1$ for any $i, j \in \mathcal{C}$; thus Recolor always runs in $O(m/k)$ time. When Line 3 of algorithm ChkRec is executed, vertex u and colors α and β satisfy the condition of Lemma 3. Since $\Phi_\pi \geq 0$ holds for any π by definition, Lemma 3 implies that the number of calls to Recolor is bounded by the value of Φ_π for the initial coloring π, which can be calculated to be $O(m)$. Thus, algorithm BalCol runs in $O(m^2/k)$ time and the following theorem holds.

Theorem 1 ([8]) *Algorithm* BalCol *solves the nearly equitable edge coloring problem in $O(m^2/k)$ time, where m and k are the numbers of edges and given colors, respectively. Moreover, the edge coloring π satisfies the balanced constraint; i.e., $||E_\pi(i)| - |E_\pi(j)|| \leq 1$ for any two colors i, j, where $E_\pi(i)$ denotes the set of edges colored with i.*

4 A Recursive Algorithm for Nearly Equitable Edge Coloring

In this section, we present the recursive algorithm RecCol, whose worst-case running time is $O(mn \log(m/(kn)+1))$. Section 4.1 explains the algorithm, and Section 4.2 shows its properties. Then Section 4.3 analyzes the running time of RecCol.

4.1 Recursive Algorithm

The basic framework of algorithm RecCol is explained as follows. When $|E| \leq kn$, RecCol calls algorithm BalCol to obtain an edge coloring of G. Otherwise, it partitions E arbitrarily into two subsets E^{L} and E^{R} so that they satisfy $E^{\mathrm{L}} \cup E^{\mathrm{R}} = E$, $E^{\mathrm{L}} \cap E^{\mathrm{R}} = \emptyset$, $|E^{\mathrm{L}}| = \lceil |E|/2 \rceil$ and $|E^{\mathrm{R}}| = \lfloor |E|/2 \rfloor$, and then recursively applies RecCol to $G_{E^{\mathrm{L}}}$ and $G_{E^{\mathrm{R}}}$ to obtain their edge colorings π^{L} and π^{R}, respectively. It then constructs a coloring π of G by merging the two colorings π^{L} and π^{R} after permuting them so that the resulting coloring π satisfies $||E_\pi(i)| - |E_\pi(j)|| \leq 1$ for any two colors $i, j \in \mathcal{C}$. Then it applies ChkRec to the resulting coloring π.

Algorithm RecCol$(G, \mathcal{C})$
Input: a graph $G = (V, E)$ and a k-color set $\mathcal{C} = \{1, 2, \ldots, k\}$
Output: a nearly equitable edge coloring π for G

1. **If** $|E| \leq kn$ **then**
   Let $\pi \leftarrow$ BalCol $(G, \mathcal{C})$.
2. **else**
3. Partition E into two subsets E^{L} and E^{R} so that they satisfy $E = E^{\mathrm{L}} \cup E^{\mathrm{R}}$, $E^{\mathrm{L}} \cap E^{\mathrm{R}} = \emptyset$, $|E^{\mathrm{L}}| = \lceil |E|/2 \rceil$ and $|E^{\mathrm{R}}| = \lfloor |E|/2 \rfloor$.
4. Let $\pi^{\mathrm{L}} \leftarrow$ RecCol$(G_{E^{\mathrm{L}}}, \mathcal{C})$.
   Let $\pi^{\mathrm{R}} \leftarrow$ RecCol$(G_{E^{\mathrm{R}}}, \mathcal{C})$.
5. Sort the color indices so that $\left|E^{\mathrm{L}}_{\pi^{\mathrm{L}}}(i_1)\right| \geq \left|E^{\mathrm{L}}_{\pi^{\mathrm{L}}}(i_2)\right| \geq \ldots \geq \left|E^{\mathrm{L}}_{\pi^{\mathrm{L}}}(i_k)\right|$ holds.
   Define a permutation $\sigma^{\mathrm{L}} : \mathcal{C} \to \mathcal{C}$ by $\sigma^{\mathrm{L}}(i_l) = l$ for all $l = 1, 2, \ldots, k$.
   Let $\pi(e) \leftarrow \sigma^{\mathrm{L}}\left(\pi^{\mathrm{L}}(e)\right)$ for all $e \in E^{\mathrm{L}}$.
6. Sort the color indices so that $\left|E^{\mathrm{R}}_{\pi^{\mathrm{R}}}(j_1)\right| \leq \left|E^{\mathrm{R}}_{\pi^{\mathrm{R}}}(j_2)\right| \leq \ldots \leq \left|E^{\mathrm{R}}_{\pi^{\mathrm{R}}}(j_k)\right|$ holds.
   Define a permutation $\sigma^{\mathrm{R}} : \mathcal{C} \to \mathcal{C}$ by $\sigma^{\mathrm{R}}(j_l) = l$ for all $l = 1, 2, \ldots, k$.
   Let $\pi(e) \leftarrow \sigma^{\mathrm{R}}\left(\pi^{\mathrm{R}}(e)\right)$ for all $e \in E^{\mathrm{R}}$.
7. Let $\pi \leftarrow$ ChkRec$(G, \mathcal{C}, \pi)$.
8. Return π.

4.2 Properties of Algorithm RecCol

This section analyzes the properties of algorithm RecCol. Since the edge coloring obtained by RecCol is the output of algorithm BalCol or ChkRec,

it is obvious that algorithm RecCol outputs a nearly equitable edge coloring for the given graph when it stops. Let us consider the computation time of algorithm RecCol except Line 4. Algorithm BalCol of Line 1 uses $O(m^2/k)$ time by Theorem 1. The permutations in Lines 5 and 6 can be computed in $O(k) = O(m)$ time (recall that $|E| > kn$ holds when these lines are called), and hence the time complexities of Lines 3, 5 and 6 are $O(m)$. Thus the remaining is the time for algorithm ChkRec of Line 7, which is determined by the running time of algorithm Recolor and the number of calls to Recolor. Lemmas 2 and 3 imply that the running time of Recolor is $O\left(\max_{i\in\mathcal{C}} |E_\pi(i)|\right)$ and the number of calls to Recolor is bounded by the value of Φ_π, where $E_\pi(i)$ and Φ_π are those of the coloring π just before calling ChkRec in Line 7. We now show the following lemmas.

Lemma 4 *The edge coloring π output by algorithm* RecCol *satisfies the balanced constraints; i.e., $||E_\pi(i)| - |E_\pi(j)|| \leq 1$ for any two colors $i, j \in \mathcal{C}$.*

Proof: We prove this by the induction on the number of edges.

When $|E| \leq kn$, we call algorithm BalCol, and in this case, Theorem 1 implies that the edge coloring π given by algorithm BalCol satisfies the balanced constraint. We now consider the case with $|E| > kn$, and assume that the lemma holds for any graph whose number of edges is smaller than $|E|$. Then, as $|E^{\mathrm{L}}|$ and $|E^{\mathrm{R}}|$ are smaller than $|E|$, after calling RecCol on both $G_{E^{\mathrm{L}}}$ and $G_{E^{\mathrm{R}}}$ in Line 4, the obtained two edge colorings π^{L} and π^{R} satisfy

$$\left|\left|E^{\mathrm{L}}_{\pi^{\mathrm{L}}}(i)\right| - \left|E^{\mathrm{L}}_{\pi^{\mathrm{L}}}(j)\right|\right| \leq 1 \tag{6}$$

$$\left|\left|E^{\mathrm{R}}_{\pi^{\mathrm{R}}}(i)\right| - \left|E^{\mathrm{R}}_{\pi^{\mathrm{R}}}(j)\right|\right| \leq 1 \tag{7}$$

for any two colors $i, j \in \mathcal{C}$ by the inductive hypothesis. Hence, after permuting the colors in Lines 5 and 6, we have

$$||E_\pi(i)| - |E_\pi(j)|| \leq 1 \tag{8}$$

for any two colors $i, j \in \mathcal{C}$, where π is the edge coloring just before calling ChkRec. Then, Lemma 2 implies that this condition still holds after calling algorithm ChkRec. □

Lemma 5 *The edge coloring π just before calling* ChkRec *in algorithm* RecCol *satisfies*

(i) $||E_\pi(i)| - |E_\pi(j)|| \leq 1$ *for all* $i, j \in \mathcal{C}$, *and*

(ii) $\Phi_\pi = O(kn)$.

Proof: (i) is obvious from the proof of Lemma 4. Below we give the proof of (ii).

To make the proof simple, we define $\hat{\Phi}_\pi(v,i)$ and $\hat{\Phi}_\pi$ as follows:

$$\begin{aligned} \hat{\Phi}_\pi(v,i) &= \varphi^{(0)}_{d(v)/k}\left(d_\pi(v,i)\right) = \left| d_\pi(v,i) - \frac{d(v)}{k} \right| \\ \hat{\Phi}_\pi(v) &= \sum_{i\in\mathcal{C}} \hat{\Phi}_\pi(v,i) \\ \hat{\Phi}_\pi &= \sum_{v\in V} \hat{\Phi}_\pi(v), \end{aligned} \tag{9}$$

where $\varphi^{(0)}_{d(v)/k}\left(d_\pi(v,i)\right)$ is defined by (5) with $x = d_\pi(v,i)$, $a = d(v)/k$ and $b = 0$. It is easy to show that for any constants a, a', $b \geq 0$ and $b' \geq 0$ satisfying $a \leq a'$ and $a'+b' \leq a+b$, $\varphi^{(b)}_a(x) \leq \varphi^{(b')}_{a'}(x)$ holds for all x. Thus, for $\bar{d}(v)-1 \leq d(v)/k$ and $(d(v)/k)+0 \leq (\bar{d}(v)-1)+2$, we obtain $\Phi_\pi(v,i) \leq \hat{\Phi}_\pi(v,i)$ for all vertices $v \in V$ and all colors $i \in \mathcal{C}$, which imply $\Phi_\pi \leq \hat{\Phi}_\pi$.

Let $d^{\mathrm{L}}(v)$ denote the number of edges in E^{L} incident to the vertex v and $d^{\mathrm{L}}_{\pi^{\mathrm{L}}}(v,i)$ denote the number of edges in E^{L} that are incident to v and colored with i under the coloring π^{L}. The definitions of $d^{\mathrm{R}}(v)$ and $d^{\mathrm{R}}_{\pi^{\mathrm{R}}}(v,i)$ are similar. After calling $\textsc{RecCol}(G_{E^{\mathrm{L}}}, \mathcal{C})$, the obtained coloring π^{L} is nearly equitable for $G_{E^{\mathrm{L}}} = \left(V_{E^{\mathrm{L}}}, E^{\mathrm{L}}\right)$, and hence

$$\left| d^{\mathrm{L}}_{\pi^{\mathrm{L}}}(v,i) - d^{\mathrm{L}}_{\pi^{\mathrm{L}}}(v,j) \right| \leq 2 \tag{10}$$

holds for any vertex $v \in V_{E^{\mathrm{L}}}$ and any two colors $i, j \in \mathcal{C}$.

Since

$$\min_{i\in\mathcal{C}} \left(d^{\mathrm{L}}_{\pi^{\mathrm{L}}}(v,i) \right) \leq \frac{d^{\mathrm{L}}(v)}{k} \leq \max_{i\in\mathcal{C}} \left(d^{\mathrm{L}}_{\pi^{\mathrm{L}}}(v,i) \right),$$

we have

$$\left| d^{\mathrm{L}}_{\pi^{\mathrm{L}}}(v,i) - \frac{d^{\mathrm{L}}(v)}{k} \right| \leq 2 \tag{11}$$

for any vertex $v \in V_{E^{\mathrm{L}}}$ and any color $i \in \mathcal{C}$. For the same reason, the coloring π^{R} obtained by calling $\textsc{RecCol}(G_{E^{\mathrm{R}}}, \mathcal{C})$ satisfies

$$\left| d^{\mathrm{R}}_{\pi^{\mathrm{R}}}(v,i) - \frac{d^{\mathrm{R}}(v)}{k} \right| \leq 2, \tag{12}$$

for any vertex $v \in V_{E^{\mathrm{R}}}$ and any color $i \in \mathcal{C}$. Since $d_\pi(v,i) = d^{\mathrm{L}}_{\pi^{\mathrm{L}}}\left(v, \left(\sigma^{\mathrm{L}}\right)^{-1}(i)\right) + d^{\mathrm{R}}_{\pi^{\mathrm{R}}}\left(v, \left(\sigma^{\mathrm{R}}\right)^{-1}(i)\right)$ and $d(v) = d^{\mathrm{L}}(v) + d^{\mathrm{R}}(v)$ hold by definition, (11) and (12) imply that the coloring π before calling $\textsc{ChkRec}$ satisfies

$$\left| d_\pi(v,i) - \frac{d(v)}{k} \right| \leq 4. \tag{13}$$

Hence we have $\Phi_\pi \leq \hat{\Phi}_\pi \leq 4kn$. □

4.3 Running Time of Algorithm RecCol

We now consider the running time of algorithm RecCol. As discussed in Section 4.2, the running time of RecCol except Line 4 is dominated by the computation time of ChkRec in Line 7, which is determined by the running time of Recolor and the number of calls to Recolor. Lemma 5-(i) and Lemma 2 imply that Recolor always runs in $O\left(\max_{i\in\mathcal{C}}|E_\pi(i)|\right) = O(m/k)$ time, while Lemma 5-(ii) and Lemma 3 imply that the number of calls to Recolor is bounded by $\Phi_\pi = O(kn)$. Thus, the running time of ChkRec becomes $O(kn \times (m/k)) = O(mn)$.

Let $T(m,n,k)$ denote the total running time of algorithm RecCol. Then we have

$$T(m,n,k) \quad = \quad \begin{cases} O(m^2/k) & \text{if } m \le kn \\ O(mn) + 2T(m/2,n,k) & \text{otherwise.} \end{cases}$$

Since $m^2/k = O(mn)$ holds for $m \le kn$, we have $T(m,n,k) = O\left(mn\log\left(m/(kn)+1\right)\right)$ for $m > kn$. For $m \le kn$, $T(m,n,k) = O(m^2/k)$ is obvious. Below we show that $m^2/k = \Theta\left(mn\log\left(m/(kn)+1\right)\right)$ holds for $m \le kn$, which implies that $T(m,n,k) = O(mn\log(m/(kn)+1))$ holds for all range of m. We also show that $mn\log(m/(kn)+1) = O(m^2/k)$, which means that algorithm RecCol is never slower than BalCol. For these, we use the following lemma, whose proof is straightforward and is omitted.

Lemma 6 *$x \le \lg(x+1)$ holds for any $x \in [0,1]$, while $\lg(x+1) \le 2x$ holds for any $x \ge 0$, where $\lg = \log_2$.*

Lemma 6 implies that

$$mn\lg\left(\frac{m}{kn}+1\right) \ge mn\frac{m}{kn} = \frac{m^2}{k}$$

holds for $0 < m/(kn) \le 1$, while

$$mn\lg\left(\frac{m}{kn}+1\right) \le mn\frac{2m}{kn} = \frac{2m^2}{k}$$

holds for any m, n and k. Thus, we conclude that $T(m,n,k) = O\left(mn\log\left(m/(kn)+1\right)\right)$ always holds, and algorithm RecCol is never asymptotically slower than algorithm BalCol. The results are summarized in the following theorem.

Theorem 2 *Algorithm* RecCol *solves the nearly equitable edge coloring problem in*

$$O\left(mn\log\left(\frac{m}{kn}+1\right)\right)$$

time, where m, n and k are the numbers of edges, vertices and given colors, respectively. Moreover, the edge coloring π satisfies the balanced constraint; i.e., $||E_\pi(i)| - |E_\pi(j)|| \le 1$ holds for any two colors $i, j \in \mathcal{C}$.

When $k = O(1)$, the time complexity of all known algorithms becomes $O(m^2)$, which implies that the time complexity, derived from the constructive proof of Hilton and de Werra [1], remains to be the best. We now consider the ratio

$$\frac{mn\log(m/n)}{m^2} = \frac{\log(m/n)}{m/n}.$$

Then we have

$$\lim_{m/n\to\infty} \frac{\log(m/n)}{m/n} = 0,$$

which implies that algorithm RecCol needs less running time than $O(m^2)$ when m/n grows to infinity; e.g., $m = n^{\vartheta}$ ($\vartheta > 1$ is an arbitrary constant).

5 Concluding Remarks

In this paper, we presented a deterministic algorithm for computing nearly equitable edge colorings. The algorithm runs in $O\left(mn\log\left(m/(kn)+1\right)\right)$ time, where m, n and k are the numbers of edges, vertices and given colors, respectively. The time complexity of the recursive algorithm is better than the previous best known time bound $O(m^2/k)$ presented by Xie et al. When $k = O(1)$, the time complexity of all known algorithms becomes $O(m^2)$, which implies that this time complexity remains to be the best for more than twenty years since 1982 when Hilton and de Werra gave a constructive proof for the existence of a nearly equitable edge coloring for any graph. Our recursive algorithm is the first that improves this time complexity when m/n grows to infinity; e.g., $m = n^{\vartheta}$ ($\vartheta > 1$ is an arbitrary constant).

Acknowledgments

The authors wish to thank the referees for several valuable comments and useful suggestions.

References

[1] A. J. W. Hilton and D. de Werra. Sufficient conditions for balanced and for equitable edge-colouring of graphs. O. R. Working paper 82/3, Dépt. of Math., École Polytechnique Fédérate de Lausanne, Switzerland, 1982.

[2] A. J. W. Hilton and D. de Werra. A sufficient condition for equitable edge-colourings of simple graphs. *Discrete Mathematics*, 128:179–201, 1994.

[3] I. Holyar. The NP-completeness of edge-coloring. *SIAM Journal on Computing*, 10:718–720, 1981.

[4] S. Nakano, Y. Suzuki, and T. Nishizeki. An algorithm for the nearly equitable edge-coloring of graphs (in Japanese). *IEICE Transactions on Information and Systems*, J78-D-I:437–444, 1995.

[5] H. Song, J. Wu, and G. Liu. The equitable edge-coloring of series-parallel graphs. In *Proceedings of the International Conference on Computational Science (ICCS 2007), Part III; Lecture Notes in Computer Science* 4489, pages 457–460, 2007.

[6] D. de Werra. Equitable colorations of graphs. *Revue française d'Informatique et de Recherche Operationelle*, R-3:3–8, 1971.

[7] D. de Werra. An extension of bipartite multigraphs. *Discrete Mathematics*, 14:133–138, 1976.

[8] X. Xie, T. Ono, S. Nakano, and T. Hirata. An improved algorithm for the nearly equitable edge-coloring problem. *IEICE Transactions on Fundamentals*, E87-A:1029–1033, 2004.

[9] X. Xie, M. Yagiura, T. Ono, and T. Hirata. Analysis of an edge coloring algorithm using Chernoff bounds. *Information Technology Letters*, 6:13–16, 2007.

[10] X. Xie, M. Yagiura, T. Ono, T. Hirata, and U. Zwick. New bounds for the nearly equitable edge coloring problem. In *Proceedings of the 18th International Symposium on Algorithms and Computation (ISAAC 2007); Lecture Notes in Computer Science* 4835, pages 280–291, 2007.

Appendix

A Some Details Omitted in Section 3

Below we give some details of the implementation issues for finding α and β in algorithm ChkRec, and the proofs of lemmas in Section 3.

A.1 Time for Finding α and β

As mentioned in Section 3, two colors α and β in Line 3 of ChkRec can be found in $O(1)$ time for each iteration if appropriately implemented, and additional time needed for maintaining data structures for this computation during the whole execution of ChkRec can be kept in $O(m+\xi)$, where ξ is the total number of calls to Recolor from ChkRec. Below is an example of such data structures. For the current vertex u and for each $d = 0, 1, 2, \ldots, d(u)$, prepare a doubly linked list of those colors $i \in \mathcal{C}$ satisfying $d_\pi(u, i) = d$, and keep pointers to the maximum and minimum d, respectively, among those d with a nonempty list. Then α and β can be found in the first cells of the linked lists identified by these two pointers, which takes $O(1)$ time. To maintain the above lists efficiently, prepare an array of size k whose ith entry contains the pointer to the cell with color i in the linked list so that the cell for i in the linked list can be accessed in $O(1)$ time whenever a color i is specified. Then it takes $O(1)$ time to remove the cells with colors α and β from the current position and insert them to appropriate positions after executing Recolor. After modifying the positions of α and β, we may need to update the pointers to the maximum and minimum d with a nonempty list. Because of (1) and (2), these pointers are modified $O(d(u))$ times during the whole execution of the "while" loop for each vertex u; hence this part takes $O\left(\sum_{u \in V} d(u) + \xi\right) = O(m + \xi)$ time during the whole execution of ChkRec.

Care should be taken in initializing the above data structure when ChkRec begins the computation for a new vertex u, because if this part is naively implemented, $\Omega(k)$ time may be spent for each u, which results in spending $\Omega(kn)$ time during the whole computation of ChkRec. To avoid this, prepare a linked list of all colors for $d = 0$ at the beginning of ChkRec, which takes $O(k) = O(m)$ time (recall that we assume $k \leq m$). Whenever a new vertex u is examined, initialize the lists by moving only those cells with the colors assigned to edges incident to u; i.e., those cells with color i with $d_\pi(u, i) = 0$ are not scanned and stay in the list for $d = 0$. This can be done in $O(d(u))$ time by scanning the list of the edges incident to u and computing the values of $d_\pi(u, i)$ for all $i \in \mathcal{C}$ with $d_\pi(u, i) \geq 1$ accordingly. Then, whenever the loop for a u is over, move all the cells in the lists for $d \geq 1$ to the list for $d = 0$, which also takes $O(d(u))$ time. In computing the values of $d_\pi(u, i)$ for all $i \in \mathcal{C}$ with $d_\pi(u, i) \geq 1$, similar care should be taken to avoid spending $\Omega(k)$ time and execute this in $O(d(u))$ time. Consequently, such initialization takes $O(d(u))$ time for each u; hence the total time for this part is $O\left(\sum_{u \in V} d(u)\right) = O(m)$ during the whole execution of ChkRec.

In summary, the time for finding α and β, as well as the time for maintaining relevant data structures, is $O(m + \xi)$ during the whole execution of ChkRec.

A.2 Proofs of Basic Lemmas

In this section, we give the proof of Lemma 1. We also include the proofs of Lemmas 2 and 3 to make the paper self-contained. Here we note that these proofs are simpler than those in [8]. Recall that $d_\pi(v, \alpha)$ is the number of edges incident to v and colored with α in a coloring π.

Lemma 1 *For any coloring π and two colors $\alpha, \beta \in \mathcal{C}$, let π' be the coloring after invoking* Recolor(G, α, β, π). *Then we have*

$$d_{\pi'}(v, \alpha) + d_{\pi'}(v, \beta) = d_\pi(v, \alpha) + d_\pi(v, \beta), \tag{14}$$

$$|d_{\pi'}(v, \alpha) - d_{\pi'}(v, \beta)| \leq |d_\pi(v, \alpha) - d_\pi(v, \beta)| \tag{15}$$

hold for all vertices $v \in V_\pi(\alpha, \beta)$.

Proof: The equation (14) is obvious, and the remaining is to show that (15) holds for all vertices $v \in V_\pi(\alpha, \beta)$. We denote by $d_{\hat{\pi}}(v, \alpha)$ the number of edges incident to v and colored with α in the augmented graph $\hat{G}$ after executing Line 9 of Recolor. Let $\eta(v)$ be the number of edges between v and the newly added vertex $\hat{v}$. Then $\eta(v) = 0$, 1 or 2 by the definition of $\hat{G}$. For any $v \in V_\pi(\alpha, \beta)$, $d_{\hat{\pi}}(v, \alpha) = d_{\hat{\pi}}(v, \beta)$ holds because we color edges alternately along an Euler circuit; hence we have

$$|d_{\pi'}(v, \alpha) - d_{\pi'}(v, \beta)| \leq \eta(v). \tag{16}$$

If $\eta(v) = 0$, then (15) is immediate from (16). If $\eta(v) = 1$, then $|d_\pi(v, \alpha) - d_\pi(v, \beta)|$ is odd, and hence is at least 1, which implies (15). For the case with $|d_\pi(v, \alpha) - d_\pi(v, \beta)| \geq 2$, (15) is also immediate from (16) and $\eta(v) \leq 2$. The remaining is the case with $\eta(v) = 2$ and $|d_\pi(v, \alpha) - d_\pi(v, \beta)| = 0$ (when $\eta(v)$ is even, $|d_\pi(v, \alpha) - d_\pi(v, \beta)|$ is even and cannot be 1). This happens only if the vertex v is connected to $\hat{v}$ in Line 7 of Recolor. In this case, $d_\pi(w, \alpha) = d_\pi(w, \beta)$ holds for all vertices w in the connected component of $G_\pi(\alpha, \beta)$ that includes v, which implies that the number of edges in this component is even. Hence the two parallel edges between the v and $\hat{v}$ receive different colors, and therefore we have (15). □

Lemma 2 [8]: *For any colors $\alpha, \beta \in \mathcal{C}$ and a coloring π, let π' be the coloring after calling* Recolor(G, α, β, π). *Then* Recolor *runs in $O(|E_\pi(\alpha) \cup E_\pi(\beta)|)$ time and the coloring π' satisfies $||E_{\pi'}(\alpha)| - |E_{\pi'}(\beta)|| \leq 1$.*

Proof: We first consider the running time of Recolor. It is clear that the running time of Recolor is dominated by the time for finding an Euler circuit, which is linear to the number of edges in the augmented graph $\hat{G}$. Let us consider the number of newly added edges in $\hat{G}$, all of which are incident to the new vertex $\hat{v}$. For each vertex v in $V_\pi(\alpha, \beta)$, the number of new edges between v

and $\hat{v}$ is at most two. Moreover, $|V_\pi(\alpha,\beta)| \le \sum_{v\in V_\pi(\alpha,\beta)} (d_\pi(v,\alpha)+d_\pi(v,\beta)) = 2|E_\pi(\alpha)\cup E_\pi(\beta)|$ holds because each vertex v in $V_\pi(\alpha,\beta)$ must be an endpoint of an edge of color α or β. Hence the number of new edges is $O(|E_\pi(\alpha)\cup E_\pi(\beta)|)$, which implies that the total running time of RECOLOR is $O(|E_\pi(\alpha)\cup E_\pi(\beta)|)$.

We next prove that after an invocation of RECOLOR, $||E_{\pi'}(\alpha)| - |E_{\pi'}(\beta)|| \le 1$ holds. If $\hat{G}$ has odd number of edges, then the coloring along the Euler circuit begins and ends with α at the new vertex $\hat{v}$. For there are even number of edges incident to $\hat{v}$, $|E_{\pi'}(\alpha)| = |E_{\pi'}(\beta)| - 1$ holds after calling RECOLOR. Otherwise, $|E_{\pi'}(\alpha)| = |E_{\pi'}(\beta)|$ is obvious. □

To prove Lemma 3, we first show the following fact.

Lemma 7 *For four numbers p, q, r and s satisfying $p+q=r+s$ and $|p-q| \ge |r-s|$, if a function f satisfies $f((1-\lambda)p+\lambda q) \le (1-\lambda)f(p)+\lambda f(q)$ for any $\lambda \in (0,1)$, then $f(r)+f(s) \le f(p)+f(q)$. Moreover, the last inequality is proper if the first two inequalities are proper.*

Proof: For four numbers p, q, r and s satisfying $p+q=r+s$ and $|p-q| > |r-s|$, suppose by symmetry that $p < r \le s < q$. Then there exists a constant $\lambda \in (0, 1/2]$ that satisfies $r = (1-\lambda)p+\lambda q$ and $s = \lambda p + (1-\lambda)q$. If f satisfies $f((1-\lambda)p+\lambda q) < (1-\lambda)f(p)+\lambda f(q)$ for any $\lambda \in (0,1)$, then we have

$$\begin{aligned} f(r)+f(s) &= f((1-\lambda)p+\lambda q) + f(\lambda p + (1-\lambda)q) \\ &< (1-\lambda)f(p)+\lambda f(q) + \lambda f(p) + (1-\lambda)f(q) \\ &= f(p)+f(q). \end{aligned} \tag{17}$$

By a similar argument, we have $f(r)+f(s) \le f(p)+f(q)$ if p, q, r and s satisfy $p+q=r+s$ and $|p-q| \ge |r-s|$, and f satisfies $f((1-\lambda)p+\lambda q) \le (1-\lambda)f(p)+\lambda f(q)$ for all $\lambda \in (0,1)$. □

Lemma 3 [8]: *Assume that there exist a vertex u and colors α, β such that $d_\pi(u,\alpha) \ge \bar{d}(u)+1$, $d_\pi(u,\beta) \le \bar{d}(u)$ and $d_\pi(u,\alpha)-d_\pi(u,\beta) \ge 3$ for a coloring π, and let π' be the coloring after calling* RECOLOR(G,α,β,π). *Then $\Phi_{\pi'} \le \Phi_\pi - 1$ holds.*

Proof: Recall that the definition of $\Phi_\pi(v)$ is

$$\Phi_\pi(v) = \sum_{i\in\mathcal{C}} \varphi^{(2)}_{\bar{d}(v)-1}\left(d_\pi(v,i)\right),$$

where

$$\varphi^{(2)}_{\bar{d}(v)-1}\left(d_\pi(v,i)\right) = \max\{d_\pi(v,i) - (\bar{d}(v)+1), (\bar{d}(v)-1) - d_\pi(v,i), 0\}$$

is a convex piecewise linear function of $d_\pi(v,i)$. From the convexity, we have

$$\begin{aligned} &\varphi^{(2)}_{\bar{d}(v)-1}\left((1-\lambda)d_\pi(v,\beta)+\lambda d_\pi(v,\alpha)\right) \\ &\qquad \le (1-\lambda)\varphi^{(2)}_{\bar{d}(v)-1}\left(d_\pi(v,\beta)\right) + \lambda\varphi^{(2)}_{\bar{d}(v)-1}\left(d_\pi(v,\alpha)\right) \end{aligned}$$

for all $\lambda \in (0,1)$. Moreover, we have (14) and (15) by Lemma 1. Hence Lemma 7 implies

$$\begin{aligned}\Phi_{\pi'}(v) - \Phi_{\pi}(v) &= \left(\varphi^{(2)}_{\bar{d}(v)-1}\left(d_{\pi'}(v,\beta)\right) + \varphi^{(2)}_{\bar{d}(v)-1}\left(d_{\pi'}(v,\alpha)\right)\right) \\ &\quad - \left(\varphi^{(2)}_{\bar{d}(v)-1}\left(d_{\pi}(v,\beta)\right) + \varphi^{(2)}_{\bar{d}(v)-1}\left(d_{\pi}(v,\alpha)\right)\right) \leq 0 \quad (18)\end{aligned}$$

for all vertices $v \in V_\pi(\alpha,\beta)$.

By the assumption of the lemma, there is a vertex u that satisfies $d_\pi(u,\alpha) \geq \bar{d}(u)+1$, $d_\pi(u,\beta) \leq \bar{d}(u)$ and $d_\pi(u,\alpha) - d_\pi(u,\beta) \geq 3$. Then, two points

$$\left(d_\pi(u,\beta), \varphi^{(2)}_{\bar{d}(u)-1}\left(d_\pi(u,\beta)\right)\right) \text{ and } \left(d_\pi(u,\alpha), \varphi^{(2)}_{\bar{d}(u)-1}\left(d_\pi(u,\alpha)\right)\right)$$

cannot be on the same line segment of the convex piecewise linear function $\varphi^{(2)}_{\bar{d}(u)-1}(\cdot)$. Hence

$$\begin{aligned}&\varphi^{(2)}_{\bar{d}(u)-1}\left((1-\lambda)d_\pi(u,\beta) + \lambda d_\pi(u,\alpha)\right) \\ &\qquad < (1-\lambda)\varphi^{(2)}_{\bar{d}(u)-1}\left(d_\pi(u,\beta)\right) + \lambda\varphi^{(2)}_{\bar{d}(u)-1}\left(d_\pi(u,\alpha)\right)\end{aligned}$$

holds for all $\lambda \in (0,1)$. As $|d_{\pi'}(u,\alpha) - d_{\pi'}(u,\beta)| \leq 2 < 3 \leq |d_\pi(u,\alpha) - d_\pi(u,\beta)|$, Lemma 7 implies

$$\begin{aligned}\Phi_{\pi'}(u) - \Phi_{\pi}(u) &= \left(\varphi^{(2)}_{\bar{d}(u)-1}\left(d_{\pi'}(u,\beta)\right) + \varphi^{(2)}_{\bar{d}(u)-1}\left(d_{\pi'}(u,\alpha)\right)\right) \\ &\quad - \left(\varphi^{(2)}_{\bar{d}(u)-1}\left(d_{\pi}(u,\beta)\right) + \varphi^{(2)}_{\bar{d}(u)-1}\left(d_{\pi}(u,\alpha)\right)\right) < 0. \quad (19)\end{aligned}$$

Recall that Φ_π is an integer and $\Phi_\pi(v)$ does not change for the vertices not in $G_\pi(\alpha,\beta)$. Thus, (18) and (19) imply $\Phi_{\pi'} \leq \Phi_\pi - 1$. □