An Efficient Algorithm for the Nearly Equitable Edge Coloring Problem

Xuzhen Xie, Mutsunori Yagiura, Takao Ono, Tomio Hirata, Uri Zwick

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 \(O(mn \log (m/(kn) + 1))\) 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 \(\vartheta > 1\).
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 $C = \{1, 2, \ldots, k\}$, 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 $\Delta$ 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 $\Delta$ 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 $O(m n \log (m/(kn) + 1))$ 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 $\pi$, ChkRec repeats the following until $\pi$ becomes nearly equitable: It checks whether the current edge coloring $\pi$ is nearly equitable or not, and if $\pi$ is not nearly equitable, it invokes an algorithm called Recolor to modify $\pi$. Let $E_\pi(i)$ denote the set of edges colored with $i$. Xie et al. introduced a potential $\Phi_\pi$ and showed that Recolor runs in
results lead to the new time complexity $O(|E_\pi(i) \cup E_\pi(j)|)$ time for relevant colors $i$ and $j$ and decreases $\Phi_\pi$ by at least one for each call to it. It is therefore preferable to construct a coloring $\pi$ such that potential $\Phi_\pi$ 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 $\pi$ 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 $\pi^L$ and $\pi^R$ of $E^L$ and $E^R$, respectively. It then constructs a coloring $\pi$ of $G$ by merging the two colorings $\pi^L$ and $\pi^R$ after permuting them. We show that the resulting edge coloring $\pi$ 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 $O(mn \log (m/(kn) + 1))$.

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 $\pi$: $E \to \mathcal{C}$; 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 \( C = \{1, 2, \ldots, k\} \), the nearly equitable edge coloring \( \pi \) 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 \( i, j \in C \), \( |d_\pi(v, i) - d_\pi(v, j)| \leq 2 \).

Definition 2 ([4]) An edge coloring \( \pi \) 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 \textsc{BalCol} and basic procedures \textsc{ChkRec} and \textsc{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 \textsc{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 \textsc{BalCol}(G, C)

\textbf{Input:} a graph \( G = (V, E) \) and a \( k \)-color set \( C = \{1, 2, \ldots, k\} \)

\textbf{Output:} a nearly equitable edge coloring \( \pi \) for \( G \)

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

Input: a graph \(G = (V, E)\), a \(k\)-color set \(C = \{1, 2, \ldots, k\}\), and an edge coloring \(\pi\)

Output: a nearly equitable edge coloring \(\pi\) for \(G\)

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

Algorithm Recolor\((G, \alpha, \beta, \pi)\)

Input: a graph \(G = (V, E)\), colors \(\alpha\) and \(\beta\), and an edge coloring \(\pi\)

Task: modify the edge coloring \(\pi\) for \(G\)

1. Let \(\hat{V} \leftarrow V_\pi(\alpha, \beta) \cup \{\hat{v}\}\) (\(\hat{v} \not\in V\)) 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
7. Draw two parallel edges between \(v\) and \(\hat{v}\), and add them into \(\hat{E}\).
8. else
9. Let \(v\) be an arbitrary vertex in \(H\). Draw two parallel edges between \(v\) and \(\hat{v}\), and add them into \(\hat{E}\).
10. Let \(\hat{G} \leftarrow (\hat{V}, \hat{E})\).
11. 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 \(t = 1, 2, \ldots, l\).
12. 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 \(\pi\) and two colors \(\alpha, \beta \in C\), let \(\pi'\) be the coloring after invoking Recolor\((G, \alpha, \beta, \pi)\). 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 \( \text{CHKREC} \). 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 \( \text{CHKREC} \), whenever the “while” loop in Line 2 of \( \text{CHKREC} \) terminates, condition \( [4] \) holds with \( v = u \), and this condition will never be violated again as mentioned above. Hence the coloring \( \pi \) output by \( \text{CHKREC} \) satisfies \( [1] \) for all vertices \( v \in V \).

We now consider the running time of algorithm \( \text{BALCOL} \). As Line 1 takes \( O(m) \) time, its running time is dominated by the running time of \( \text{CHKREC} \). Note that, in \( \text{CHKREC} \), algorithm \( \text{RECOLOR} \) 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 \( \alpha \) and \( \beta \) in Line 3 of \( \text{CHKREC} \) 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 \( \text{CHKREC} \) can be kept in \( O(m + \xi) \), where \( \xi \) is the total number of calls to \( \text{RECOLOR} \) from \( \text{CHKREC} \). 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 \( d \) with a nonempty list. Then \( \alpha \) and \( \beta \) 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 \( \alpha \) and \( \beta \) from the current position and insert them to appropriate positions after executing \( \text{RECOLOR} \). After modifying the positions of \( \alpha \) and \( \beta \), 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 \( O(\sum_{u \in V} d(u) + \xi) = O(m + \xi) \) time during the whole execution of \( \text{CHKREC} \). Care should be taken in initializing the above data structure when \( \text{CHKREC} \) begins the computation for a new vertex \( u \), because if this part is naively implemented, \( \Omega(kn) \) 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 \( \Phi_\pi \), which is defined as follows. For all vertices \( v \in V \), let \( \bar{d}(v) = \lfloor d(v)/k \rfloor \). Define

\[
\Phi_\pi(v, i) = \varphi^{(2)}_{\bar{d}(v) - 1}(d_\pi(v, i))
\]

\[
\Phi_\pi(v) = \sum_{i \in C} \Phi_\pi(v, i)
\]

\[
\Phi_\pi = \sum_{v \in V} \Phi_\pi(v),
\]

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

\[
\varphi^{(b)}_{\bar{d}(v)}(x) = \max\{x - a - b, a - x, 0\}
\]

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

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

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

The initial edge coloring \( \pi \) of algorithm BALCol satisfies \( |E_\pi(i)| - |E_\pi(j)| \leq 1 \) for any \( i, j \in C \), which implies that \( |E_\pi(i)| = O(m/k) \) holds for all \( i \in C \). Lemma 2 implies that after invoking Recolor, the edge coloring \( \pi' \) also satisfies \( |E_{\pi'}(i)| - |E_{\pi'}(j)| \leq 1 \) for any \( i, j \in C \); thus Recolor always runs in \( O(m/k) \) time. When Line 3 of algorithm ChkRec is executed, vertex \( u \) and colors \( \alpha \) and \( \beta \) satisfy the condition of Lemma 3. Since \( \Phi_\pi \geq 0 \) holds for any \( \pi \) by definition, Lemma 3 implies that the number of calls to Recolor is bounded by the value of \( \Phi_\pi \) for the initial coloring \( \pi \), 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 \( \pi \) 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 \text{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 \text{RecCol}.

4.1 Recursive Algorithm

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

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

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

4.2 Properties of Algorithm \text{RecCol}

This section analyzes the properties of algorithm \text{RecCol}. Since the edge coloring obtained by \text{RecCol} is the output of algorithm \text{BalCol} or \text{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 \(O(\max_{\pi \in \Pi} |E_\pi(i)|)\) and the number of calls to **Recolor** is bounded by the value of \(\Phi_\pi\), where \(E_\pi(i)\) and \(\Phi_\pi\) are those of the coloring \(\pi\) just before calling **ChkRec** in Line 7. We now show the following lemmas.

**Lemma 4** The edge coloring \(\pi\) 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 \(\pi\) 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 \(|E_L|\) and \(|E_R|\) are smaller than \(|E|\), after calling **RecCol** on both \(G_E\) and \(G_{E_R}\) in Line 4, the obtained two edge colorings \(\pi_L\) and \(\pi_R\) satisfy

\[
||E_{\pi_L}(i)| - |E_{\pi_L}(j)|| \leq 1 \quad (6)
\]
\[
||E_{\pi_R}(i)| - |E_{\pi_R}(j)|| \leq 1 \quad (7)
\]

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

\[
||E_\pi(i)| - |E_\pi(j)|| \leq 1 \quad (8)
\]

for any two colors \(i, j \in \mathcal{C}\), where \(\pi\) is the edge coloring just before calling **ChkRec**. Then, Lemma 3 implies that this condition still holds after calling algorithm **ChkRec**. \(\square\)

**Lemma 5** The edge coloring \(\pi\) 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:

$$
\hat{\Phi}_{\pi}(v, i) = \varphi_{d(v)/k}^{(0)}(d_{\pi}(v, i)) = \left| d_{\pi}(v, i) - \frac{d(v)}{k} \right| \quad (9)
$$

$$
\hat{\Phi}_{\pi}(v) = \sum_{i \in C} \hat{\Phi}_{\pi}(v, i)
$$

$$
\hat{\Phi}_{\pi} = \sum_{v \in V} \hat{\Phi}_{\pi}(v),
$$

where $\varphi_{d(v)/k}^{(0)}(d_{\pi}(v, i))$ 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 $a' + b' \leq a + b$, $\varphi_{a}^{(b)}(x) \leq \varphi_{a'}^{(b')}(x)$ holds for all $x$. Thus, for $d(v) - 1 \leq d(v)/k$ and $(d(v)/k) + 1 \leq d(v) - 1 + 2$, we obtain $\Phi_{\pi}(v, i) \leq \hat{\Phi}_{\pi}(v, i)$ for all vertices $v \in V$ and all colors $i \in C$, which imply $\Phi_{\pi} \leq \hat{\Phi}_{\pi}$.

Let $d_{L}(v)$ denote the number of edges in $E_{L}$ incident to the vertex $v$ and $d_{nL}(v, i)$ denote the number of edges in $E_{L}$ that are incident to $v$ and colored with $i$ under the coloring $\pi_{L}$. The definitions of $d_{R}(v)$ and $d_{nR}(v, i)$ are similar. After calling RECCL($G_{E_{L}}, C$), the obtained coloring $\pi_{L}$ is nearly equitable for $G_{E_{L}} = (V_{E_{L}}, E_{L})$, and hence

$$
|d_{nL}(v, i) - d_{nL}(v, j)| \leq 2 \quad (10)
$$

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

Since

$$
\min_{i \in C} \left( d_{nL}(v, i) \right) \leq \frac{d_{L}(v)}{k} \leq \max_{i \in C} \left( d_{nL}(v, i) \right),
$$

we have

$$
\left| d_{nL}(v, i) - \frac{d_{L}(v)}{k} \right| \leq 2 \quad (11)
$$

for any vertex $v \in V_{E_{L}}$ and any color $i \in C$. For the same reason, the coloring $\pi_{R}$ obtained by calling RECCL($G_{E_{R}}, C$) satisfies

$$
\left| d_{nR}(v, i) - \frac{d_{R}(v)}{k} \right| \leq 2 \quad (12)
$$

for any vertex $v \in V_{E_{R}}$ and any color $i \in C$. Since $d_{\pi}(v, i) = d_{nL}(v, (\sigma^{L})^{-1}(i)) + d_{nR}(v, (\sigma^{R})^{-1}(i))$ and $d(v) = d_{L}(v) + d_{R}(v)$ hold by definition, (11) and (12) imply that the coloring $\pi$ before calling CHKREC satisfies

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

Hence we have $\Phi_{\pi} \leq \hat{\Phi}_{\pi} \leq 4kn$. \qed
4.3 Running Time of Algorithm \textsc{RecCol}

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

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

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

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

\textbf{Lemma 6} $x \leq \lg(x + 1)$ holds for any $x \in [0,1]$, while $\lg(x + 1) \leq 2x$ holds for any $x \geq 0$, where $\lg = \log_2$.

Lemma 6 implies that

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

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

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

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

\textbf{Theorem 2} Algorithm \textsc{RecCol} solves the nearly equitable edge coloring problem in

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

time, where $m$, $n$ and $k$ are the numbers of edges, vertices and given colors, respectively. Moreover, the edge coloring $\pi$ satisfies the balanced constraint; i.e., $|E_\pi(i)| - |E_\pi(j)| \leq 1$ 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 $O(mn \log (m/(kn) + 1))$ 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 Fédérale 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 $\alpha$ and $\beta$ in algorithm ChkRec, and the proofs of lemmas in Section 3.

A.1 Time for Finding $\alpha$ and $\beta$

As mentioned in Section 3, two colors $\alpha$ and $\beta$ 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 $\xi$ 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 $i \in C$ satisfying $d_\pi(u, i) = d$, and keep pointers to the maximum and minimum $d$, respectively, among those $d$ with a nonempty list. Then $\alpha$ and $\beta$ 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 $i$th 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 $\alpha$ and $\beta$ from the current position and insert them to appropriate positions after executing Recolor. After modifying the positions of $\alpha$ and $\beta$, 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 $i \in C$ 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 $i \in C$ 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 $\alpha$ and $\beta$, 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 $\alpha$ in a coloring $\pi$.

**Lemma 1** For any coloring $\pi$ and two colors $\alpha, \beta \in C$, let $\pi'$ be the coloring after invoking RECOLOR($G, \alpha, \beta, \pi$). Then we have

\[
    d_\pi(v, \alpha) + d_{\pi'}(v, \beta) = d_\pi(v, \alpha) + d_\pi(v, \beta),
\]

(14)

\[
    |d_\pi(v, \alpha) - d_{\pi'}(v, \beta)| \leq |d_\pi(v, \alpha) - d_\pi(v, \beta)|
\]

(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 $d_\pi(v, \alpha)$ the number of edges incident to $v$ and colored with $\alpha$ 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 $\eta(v) \geq 0, 1$ or 2 by the definition of $\hat{G}$. For any $v \in V_\pi(\alpha, \beta)$, $d_\pi(v, \alpha) = d_\pi(v, \beta)$ holds because we color edges alternately along an Euler circuit; hence we have

\[
    |d_\pi(v, \alpha) - d_{\pi'}(v, \beta)| \leq \eta(v).
\]

(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, $d_\pi(v, \alpha) = d_{\pi'}(v, \beta)$ 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). \hfill $\square$

**Lemma 2** [8]: For any colors $\alpha, \beta \in C$ and a coloring $\pi$, let $\pi'$ be the coloring after calling RECOLOR($G, \alpha, \beta, \pi$). Then RECOLOR runs in $O(|E_\pi(\alpha) \cup E_\pi(\beta)|)$ time and the coloring $\pi'$ 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, \(|V_\pi(\alpha, \beta)| \leq \sum_{v \in V_\pi(\alpha, \beta)} (d_\pi(v, \alpha) + d_\pi(v, \beta)) = 2|E_\pi(\alpha) \cup E_\pi(\beta)| \) holds because each vertex \( v \) in \( V_\pi(\alpha, \beta) \) must be an endpoint of an edge of color \( \alpha \) or \( \beta \). Hence the number of new edges is \( O(|E_\pi(\alpha) \cup E_\pi(\beta)|) \), which implies that the total running time of \textsc{Recolor} is \( O(|E_\pi(\alpha) \cup E_\pi(\beta)|) \).

We next prove that after an invocation of \textsc{Recolor}, \(||E_\pi'(\alpha)| - |E_\pi'(\beta)|| \leq 1 \) holds. If \( \hat{G} \) has odd number of edges, then the coloring along the Euler circuit begins and ends with \( \alpha \) 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 \textsc{Recolor}. Otherwise, \(|E_\pi'(\alpha)| = |E_\pi'(\beta)| \) is obvious. \( \square \)

To prove Lemma 7, we first show the following fact.

**Lemma 7** For four numbers \( p, q, r \) and \( s \) satisfying \( p + q = r + s \) and \(|p - q| \geq |r - s|\), if a function \( f \) satisfies \( f((1 - \lambda)p + \lambda q) \leq (1 - \lambda)f(p) + \lambda f(q) \) for any \( \lambda \in (0, 1) \), then \( f(r) + f(s) \leq f(p) + f(q) \). 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 \(|p - q| \geq |r - s|\), suppose by symmetry that \( p < r \leq s < q \). 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

\[
\begin{align*}
    f(r) + f(s) &= f((1 - \lambda)p + \lambda q) + f(\lambda p + (1 - \lambda)q) \\
    &< (1 - \lambda)f(p) + \lambda f(q) + \lambda f(p) + (1 - \lambda)f(q) \\
    &= f(p) + f(q). 
\end{align*}
\]

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

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

**Proof:** Recall that the definition of \( \Phi_{\pi}(v) \) is

\[
\Phi_{\pi}(v) = \sum_{i \in C} \varphi_{d(v) - 1}^{(2)}(d_\pi(v, i)),
\]

where

\[
\varphi_{d(v) - 1}^{(2)}(d_\pi(v, i)) = \max\{d_\pi(v, i) - (d(v) + 1), (d(v) - 1) - d_\pi(v, i), 0\}
\]

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

\[
\varphi_{d(v) - 1}^{(2)}((1 - \lambda)d_\pi(v, \beta) + \lambda d_\pi(v, \alpha)) \\
\leq (1 - \lambda)\varphi_{d(v) - 1}^{(2)}(d_\pi(v, \beta)) + \lambda \varphi_{d(v) - 1}^{(2)}(d_\pi(v, \alpha))
\]

for every \( \lambda \in (0, 1) \).
for all $\lambda \in (0, 1)$. Moreover, we have (14) and (15) by Lemma 1. Hence Lemma 7 implies
\[
\Phi_{\pi'}(v) - \Phi_\pi(v) = \left( \varphi_{d_{\pi'}}(v, \beta) + \varphi_{d_{\pi'}}(v, \alpha) \right) - \left( \varphi_{d_{\pi}}(v, \beta) + \varphi_{d_{\pi}}(v, \alpha) \right) \leq 0 \tag{18}
\]
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
\[
\left( d_\pi(u, \beta), \varphi_{d(u) - 1}(d_\pi(u, \beta)) \right) \text{ and } \left( d_\pi(u, \alpha), \varphi_{d(u) - 1}(d_\pi(u, \alpha)) \right)
\]
cannot be on the same line segment of the convex piecewise linear function $\varphi_{d(u) - 1}(\cdot)$. Hence
\[
\varphi_{d(u) - 1}(\{(1 - \lambda)d_\pi(u, \beta) + \lambda d_\pi(u, \alpha)\})
\]
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
\[
\Phi_{\pi'}(u) - \Phi_\pi(u) = \left( \varphi_{d(u) - 1}(d_\pi(u, \beta)) + \varphi_{d(u) - 1}(d_\pi(u, \alpha)) \right) - \left( \varphi_{d(u) - 1}(d_\pi(u, \beta)) + \varphi_{d(u) - 1}(d_\pi(u, \alpha)) \right) < 0. \tag{19}
\]
Recall that $\Phi_{\pi}$ is an integer and $\Phi_\pi(v)$ does not change for the vertices not in $G_{\pi}(\alpha, \beta)$. Thus, (18) and (19) imply $\Phi_{\pi'} \leq \Phi_\pi - 1$. \hfill \qed