A linear time algorithm for L(2,1)-labeling of trees

Hasunuma, Toru

Ishii, Toshimasa

Ono, Hirotaka
九州大学大学院システム情報科学研究院 : 助教

Uno, Yushi

出版情報：2008-10
バージョン：
権利関係：

A linear time algorithm for $L(2,1)$-labeling of trees

Toru Hasunuma [*], Toshimasa Ishii [†], Hirotaka Ono [‡] and Yushi Uno [§]

Abstract

An $L(2,1)$-labeling of a graph G is an assignment f from the vertex set $V(G)$ to the set of nonnegative integers such that $|f(x) - f(y)| \geq 2$ if x and y are adjacent and $|f(x) - f(y)| \geq 1$ if x and y are at distance 2, for all x and y in $V(G)$. A k-$L(2,1)$-labeling is an assignment $f : V(G) \to \{0, \ldots, k\}$, and the $L(2,1)$-labeling problem asks the minimum k, which we denote by $\lambda(G)$, among all possible assignments. It is known that this problem is NP-hard even for graphs of treewidth 2, and tree is one of a very few classes for which the problem is polynomially solvable. The running time of the best known algorithm for trees had been $O(\Delta^{4.5}n)$ for more than a decade, however, an $O(n^{1.75})$-time algorithm has been proposed recently, which substantially improved the previous one, where Δ is the maximum degree of T and $n = |V(T)|$. In this paper, we finally establish a linear time algorithm for $L(2,1)$-labeling of trees.

[*]Department of Mathematical and Natural Sciences, The University of Tokushima, Tokushima 770–8502 Japan. hasunuma@ias.tokushima-u.ac.jp

[†]Department of Information and Management Science, Otaru University of Commerce, Otaru 047-8501, Japan. ishii@res.otaru-uc.ac.jp

[‡]Department of Computer Science and Communication Engineering, Kyushu University, Fukuoka 812-8581, Japan. ono@csce.kyushu-u.ac.jp

[§]Department of Mathematics and Information Sciences, Graduate School of Science, Osaka Prefecture University, Sakai 599-8531, Japan. uno@mi.s.osakafu-u.ac.jp

1 Introduction

Let G be an undirected graph. An $L(2,1)$-*labeling* of a graph G is an assignment f from the vertex set $V(G)$ to the set of nonnegative integers such that $|f(x)-f(y)| \geq 2$ if x and y are adjacent and $|f(x)-f(y)| \geq 1$ if x and y are at distance 2, for all x and y in $V(G)$. A k-$L(2,1)$-labeling is an assignment $f : V(G) \to \{0, \ldots, k\}$, and the $L(2,1)$-*labeling problem* asks the minimum k among all possible assignments. We call this invariant, the minimum value k, the $L(2,1)$-*labeling number* and is denoted by $\lambda(G)$. Notice that we can use $k+1$ different labels when $\lambda(G) = k$ since we can use 0 as a label for conventional reasons.

The original notion of $L(2,1)$-labeling can be seen in the context of frequency/channel assignment, where 'close' transmitters must receive different frequencies and 'very close' transmitters must receive frequencies that are at least two frequencies apart so that they can avoid interference. Due to its practical importance, the $L(2,1)$-labeling problem has been widely studied. From the graph theoretical point of view, since this is a kind of vertex coloring problem, it has attracted a lot of interest [3, 7, 9, 12]. In this context, $L(2,1)$-labeling is generalized into $L(p,q)$-labeling for arbitrary nonnegative integers p and q, and in fact, we can see that $L(1,0)$-labeling ($L(p,0)$-labeling, actually) is equivalent to the classical vertex coloring. We can find a lot of related results on $L(p,q)$-labelings in comprehensive surveys by Calamoneri [2] and Yeh [13].

Related Work: There are also a number of studies on the $L(2,1)$-labeling problem from the algorithmic point of view [1, 5, 11]. It is known to be NP-hard for general graphs [7], and it still remains NP-hard for some restricted classes of graphs, such as planar graphs, bipartite graph, chordal graphs [1], and recently it turned out to be NP-hard even for graphs of treewidth 2 [4]. In contrast, only a few graph classes are known to have polynomial time algorithms for this problem, e.g., we can determine the $L(2,1)$-labeling number of paths, cycles, wheels within polynomial time [7].

As for trees, Griggs and Yeh [7] showed that $\lambda(T)$ is either $\Delta+1$ or $\Delta+2$ for any tree T, and also conjectured that determining $\lambda(T)$ is NP-hard, however, Chang and Kuo [3] disproved this by presenting a polynomial time algorithm for computing $\lambda(T)$. Their polynomial time algorithm exploits the fact that $\lambda(T)$ is either $\Delta+1$ or $\Delta+2$ for any tree T. Its running time is $O(\Delta^{4.5}n)$, where Δ is the maximum degree of a tree T and $n = |V(T)|$. Recently, an $O(\min\{n^{1.75}, \Delta^{1.5}n\})$ time algorithm has been proposed [8], which substantially improves the previous result.

Both algorithms are based on dynamic programming (DP) approach, which checks whether $(\Delta+1)$-$L(2,1)$-labeling is possible or not, from leaf vertices to a root vertex in the original tree strcucture, and the principle of optimality requires to solve at each vertex of the tree the assignments of labels to subtrees. The assignments are formulated as the maximum matching in a bipartite graph with $O(\Delta)$ vertices and $O(\Delta^2)$ edges, that is, it takes $O(\Delta^{2.5})$ time [10]. Since the assignment at each vertex should be solved Δ^2 times to fill the DP table up, the total running time of Chang and Kuo's algorithm is $O(\Delta^{4.5}n)$. The $O(n^{1.75})$-time algorithm in [8] circumvents this heavy computation by two ways: one is that it is sufficient to solve the bipartite matching not Δ^2 times but essentially Δ times at each vertex; the other is an amortized analysis, in which we can show that the bipartite matching problems should be solved roughly only $O(n/\Delta)$ times in average with respect to degrees. Note that the observation is derived from the fact that the assignments near leaf vertices can be solved efficiently. We give a concise review of these two algorithms in Sections 2 and 3, respectively.

Our Contributions: In this paper, we present a linear time algorithm for $L(2,1)$-labeling of trees. It is based on the similar DP approach to the preceding two polynomial time algorithms [3, 8]. In our new algorithm, besides using their ideas, we introduce the notion of "label compatibility", which indicates how we flexibly change labels with preserving its $(\Delta+1)$-$L(2,1)$-labeling. Interestingly, we can show that only $O(\log_\Delta n)$ labels are essential for $L(2,1)$-labeling in any input tree by using this notion. By utilizing this fact, we can replace the bipartite matching of graphs with the maximum flow of much smaller networks as the engine to find the assignments. Furthermore, the argument of the label compatibility explains that, in the DP, a subtree with size at most Δ^c (c is some constant) can be considered a so-called generalized leaf. This enables us to do a more detailed amortized analysis than the previous factor $O(n/\Delta)$. Consequently, our algorithm finally achieves its linear running time.

Organization of this Paper: The rest of this paper is organized as follows. Section 2 gives basic definitions and introduces as a warm-up the idea of Chang and Kuo's $O(\Delta^{4.5}n)$ time algorithm based on dynamic programming. The improved $O(n^{1.75})$-time algorithm and its ideas are described in Section 3. Section 4 introduces the notion of label compatibility that can bundle a set of compatible vertices and reduce the size of the graph constructed for computing bipartite matchings. Moreover, this allows us to use maximum-flow based computation for them. In Section 5, we give precise analyses to achieve linear running time.

2 Preliminaries

2.1 Definitions and Notations

A graph G is an ordered set of its vertex set $V(G)$ and edge set $E(G)$ and is denoted by $G = (V(G), E(G))$. We assume throughout this paper that all graphs are undirected, simple and connected, unless otherwise stated. Therefore, an edge $e \in E(G)$ is an unordered pair of vertices u and v, which are *end vertices* of e, and we often denote it by $e = (u, v)$. Two vertices u and v are *adjacent* if $(u, v) \in E(G)$. A graph $G = (V(G), E(G))$ is called *bipartite* if the vertex set $V(G)$ can be divided into two disjoint sets V_1 and V_2 such that every edge in $E(G)$ connects a vertex in V_1 and one in V_2; such G is denoted by (V_1, V_2, E).

For a graph G, the (*open*) *neighborhood* of a vertex $v \in V(G)$ is the set $N_G(v) = \{u \in V(G) \mid (u, v) \in E(G)\}$, and the *closed neighborhood* of v is the set $N_G[v] = N_G(v) \cup \{v\}$. The *degree* of a vertex v is $|N_G(v)|$, and is denoted by $d_G(v)$. We use $\Delta(G)$ to denote the maximum degree of a graph G. A vertex whose degree is $\Delta(G)$ is called *major*. We often drop G in these notations if there are no confusions. A vertex whose degree is 1 is called a *leaf vertex*, or simply a *leaf*. A *path* in G is a sequence $v_1, v_2, \ldots, v_\ell$ of vertices such that $(v_i, v_{i+1}) \in E$ for $i = 1, 2, \ldots, \ell - 1$, or equivalently, a sequence $(v_1, v_2), (v_2, v_3), \ldots, (v_{\ell-1}, v_\ell)$ of edges (v_i, v_{i+1}) for $i = 1, 2, \ldots, \ell - 1$. A path $v_1, v_2, \ldots, v_\ell$ is a *cycle* if $v_1 = v_\ell$. A graph is a *tree* if it is connected and has no cycle.

In describing algorithms, it is convenient to regard the input tree to be rooted at a leaf vertex r. Then we can define the parent-child relationship on vertices in the usual way. For a rooted tree, its *height* is the length of the longest path from the root to a leaf. For any vertex v, the set of its children is denoted by $C(v)$. For a vertex v, define $d'(v) = |C(v)|$.

2.2 Chang and Kuo's Algorithm

We first review some significant properties on $L(2, 1)$-labeling of graphs or trees that have been used so far for designing $L(2, 1)$-labeling algorithms. We can see that $\lambda(G) \geq \Delta + 1$ holds for any graph G. Griggs and Yeh [7] observed that any major vertex in G must be labeled 0 or $\Delta + 1$ when $\lambda(G) = \Delta + 1$, and that if $\lambda(G) = \Delta + 1$, then $N_G[v]$ contains at most two major vertices for any $v \in V(G)$. Furthermore, they showed that $\lambda(T)$ is either $\Delta + 1$ or $\Delta + 2$ for any tree T. By using this fact, Chang and Kuo [3] presented an $O(\Delta^{4.5}n)$ time algorithm for computing $\lambda(T)$.

Next we review the idea of Chang and Kuo's dynamic programming algorithm (CK algorithm) for the $L(2, 1)$-labeling problem of trees, since our linear time algorithm also depends on the same formula of the principle of optimality. The algorithm determines if $\lambda(T) = \Delta + 1$, and if so, we can easily construct the labeling with $\lambda(T) = \Delta + 1$.

Before explaining the idea, we introduce some notations. We assume for explanation that T is rooted at some leaf vertex r. Given a vertex v, we denote the subtree of T rooted at v by $T(v)$. Let $T(u, v)$ be a tree rooted at u that forms $T(u,v) = (\{u\} \cup V(T(v)), \{(u,v)\} \cup E(T(v)))$. Note that this u is just a virtual vertex for explanation and $T(u,v)$ is uniquely determined by $T(v)$. For $T(u,v)$, we define

$$\delta((u,v),(a,b)) = \begin{cases} 1, & \text{if } \lambda(T(u,v) \mid f(u)=a, f(v)=b) \le \Delta+1, \\ 0, & \text{otherwise,} \end{cases}$$

where $\lambda(T(u,v) \mid f(u)=a, f(v)=b)$ denotes the $L(2,1)$-labeling number on $T(u,v)$ under the condition that $f(u)=a$ and $f(v)=b$, i.e., the minimum k of k-$L(2,1)$-labeling on $T(u,v)$ satisfying $f(u)=a$ and $f(v)=b$. This δ function satisfies the following formula:

$$\delta((u,v),(a,b)) = \begin{cases} 1, & \text{if there is an injective assignment } g : \{w_1, w_2, \dots, w_{d'(v)}\} \to \{0,1,\dots,\Delta+1\} \\ & -\{a, b-1, b, b+1\} \text{ such that } \delta((v,w_i),(b,g(w_i))) = 1 \text{ for each } i, \\ 0, & \text{otherwise,} \end{cases}$$

where $w_1, \dots, w_{d'(v)}$ are the children of v. The existence of such an injective assignment g is formalized as the maximum matching problem: For a bipartite graph $G(u,v,a,b) = (C(v), X, E(u,v,a,b))$, where $C(v) = \{w_1, w_2, \dots, w_{d'(v)}\}$, $X = \{0,1,\dots,\Delta,\Delta+1\}$ and $E(u,v,a,b) = \{(w,c) \mid \delta((v,w),(b,c)) = 1, c \in X - \{a\}, w \in C(v)\}$, we can see that there is an injective assignment g: $\{w_1, w_2, \dots, w_{d'(v)}\} \to \{0,1,\dots,\Delta+1\} - \{a, b-1, b, b+1\}$ if there exists a matching of size $d'(v)$ in $G(u,v,a,b)$. Namely, for $T(u,v)$ and two labels a and b, we can easily (i.e., in polynomial time) determine the value of $\delta((u,v),(a,b))$ if the values of δ function for $T(v,w_i)$ and any two pairs of labels are given. Now let $t(v)$ be the time for calculating $\delta((u,v),(*,*))$ for vertex v. CK algorithm solves the bipartite matching problems of $O(\Delta)$ vertices and $O(\Delta^2)$ edges $O(\Delta^2)$ times for each v, in order to obtain δ-values for all combinations of labels a and b. This amounts $t(v) = O(\Delta^{2.5}) \times O(\Delta^2) = O(\Delta^{4.5})$, where the first $O(\Delta^{2.5})$ is the time complexity of the bipartite matching problem [10]. Thus the total running time is $\sum_{v \in V} t(v) = O(\Delta^{4.5} n)$.

3 An $O(n^{1.75})$-time Algorithm and its Ideas

In this section, we review the ideas of the $O(n^{1.75})$-time algorithm presented in [8]. The algorithm is described in Algorithm 1, and it contains two subroutines (Algorithms 2 and 3).

Algorithm 1 Label-Tree

1: Do Preprocessing (Algorithm 2).
2: If $N[v]$ contains at least three major vertices for some vertex $v \in V$, output "No". Halt.
3: If the number of major vertices is at most $\Delta - 6$, output "Yes". Halt.
4: For $T(u,v)$ with $v \in V_Q$ (its height is 2), let $\delta((u,v),(a,0)) := 1$ for each label $a \neq 0,1$, $\delta((u,v),(a,\Delta+1)) := 1$ for each label $a \neq \Delta, \Delta+1$, and $\delta((u,v),(*,*)) := 0$ for any other pair of labels. Let $h := 3$.
5: For all $T(u,v)$ of height h, compute $\delta((u,v),(*,*))$ (Compute $\delta((u,v),(*,b))$ for each b by Maintain-matching$(G(u,v,-,b))$ (Algorithm 3).
6: If $h = h^*$ where h^* is the height of root r of T, then goto Step 7. Otherwise let $h := h+1$ and goto Step 4.
7: If $\delta((r,v),(a,b)) = 1$ for some (a,b), then output "Yes". Otherwise output "No". Halt.

Algorithm 2 Preprocessing

1: Check if there is a leaf v whose unique neighbor u has degree less than Δ. If so, remove v and edge (u,v) from T until such a leaf does not exist.
2: Check if there is a path component whose size is at least 4, say $v_1, v_2, \dots, v_\ell$, and let v_0 and $v_{\ell+1}$ be the unique adjacent vertices of v_1 and v_ℓ other than v_2 and $v_{\ell-1}$, respectively. If it exists, assume T is rooted at v_1, divide T into $T_1 := T(v_1, v_0)$ and $T_2 := T(v_4, v_5)$, and remove v_2 and v_3. Continue this operation until such a path component does not exist.

Algorithm 3 MAINTAIN-MATCHING($G(u, v, -, b)$)

1: Find a maximum bipartite matching M of $G(u, v, -, b)$. If $G(u, v, -, b)$ has no matching of size $d'(v)$, output $\delta((u, v), (*, b))$ as $\delta((u, v), (i, b)) = 0$ for every label i.

2: Let X' be the set of unmatched vertices under M. For each label vertex i that is reachable from a vertex in X' via M-alternating path, let $\delta((u, v), (i, b)) = 1$. For the other vertices j, let $\delta((u, v), (j, b)) = 0$. Output $\delta((u, v), (*, b))$.

The running time $O(n^{1.75})$ is roughly achieved by two strategies. One is that the problem can be solved by a simple linear time algorithm if $\Delta = \Omega(\sqrt{n})$. The other is that we can solve the problem for any input trees in $O(\Delta^{1.5} n)$ time. The ideas used in the latter strategy play key roles also in our new linear time algorithm. In the following subsections, we will see more detailed ideas.

Efficient Search of Augmenting Paths in Solving Bipartite Matchings

Recall that CK algorithm computes the maximum bipartite matching to calculate $\delta((u, v), (a, b))$ for every pair of labels a and b; the bipartite matching is solved Δ^2 times per $\delta((u, v), (*, *))$. The first idea of the speedup is that, we do not solve the bipartite matching problems every time from scratch, but reuse the obtained matching structure. We focus on the fact that the graphs $G(u, v, a, b)$ and $G(u, v, a', b)$ has almost the same topology except edges from vertices corresponding to a or a'. To utilize this fact, we solve the bipartite matching problem for $G(u, v, -, b)$, where $E(u, v, -, b) = \{(w, c) \mid \delta((v, w), (b, c)) = 1, c \in X, w \in C(v)\}$, instead of $G(u, v, a, b)$ for a specific a. A maximum matching of this $G(u, v, -, b)$ satisfies the following properties:

Property 1 (Lemma 3 [8]) *If $G(u, v, -, b)$ has no matching of size $d'(v)$, then $\delta((u, v), (i, b)) = 0$ for any label i.* □

Property 2 (Lemma 4 [8]) $\delta((u, v), (i, b)) = 1$ *if and only if vertex i can be reached by an M-alternating path from some vertex in X' in $G(u, v, -, b)$, where M denotes a maximum matching of $G(u, v, -, b)$ (of size $d'(v)$).* □

From these properties, we can see that Algorithm MAINTAIN-MATCHING (Algorithm 3) correctly computes $\delta((u, v), (*, b))$. Since Step 2 of MAINTAIN-MATCHING is performed by a single graph search, the total running time of MAINTAIN-MATCHING is $O(\Delta^{1.5} d'(v)) + O(\Delta d'(v)) = O(\Delta^{1.5} d'(v))$ (for solving the bipartite matching of $G(u, v, -, b)$, which has $O(\Delta)$ vertices and $O(\Delta d'(v))$ edges, and for a single graph search). Since this calculation is done for all b, we have $t(v) = O(\Delta^{2.5} d'(v))$, which improves the running time $t(v) = O(\Delta^{4.5})$ of the original CK algorithm.

Preprocessing Operations and Amortized Analysis

The other technique of the speedup is based on preprocessing operations and amortized analysis. We introduce preprocessing operations for an input tree (PREPROCESSING, Algorithm 2), where a sequence of consecutive vertices $v_1, v_2, \ldots, v_\ell$ is called a *path component* if $(v_i, v_{i+1}) \in E$ for all $i = 1, 2, \ldots, \ell - 1$ and $d(v_i) = 2$ for all $i = 1, 2, \ldots, \ell$, and ℓ is called the *size* of the path component. They are carried out (1) to remove the vertices that are 'irrelevant' to the $L(2, 1)$-labeling number, and (2) to divide T into several subtrees that preserve the $L(2, 1)$-labeling number. It is easy to show that neither of the operations affects the $L(2, 1)$-labeling number. Note that, these operations may not reduce the size of the input tree, but more importantly, they restrict the shape of input trees, which enables an amortized analysis.

After the preprocessing operations, the input trees satisfy the following properties.

Property 3 *All vertices connected to a leaf vertex are major vertices.* □

Property 4 *The size of any path component of T is at most* 3. $\square$

By Property 3, if we go down the resulting tree from a root, then we reach a major vertex whose children are all leaves. For the set V_Q of such vertices, we can observe the following: (1) for $v \in V_Q$ $\delta((u,v),(a,b)) = 1$ if and only if $b = 0$ or $\Delta + 1$ and $|a - b| \geq 2$, (2) $|V_Q| \leq n/\Delta$. Note that (1) implies that we do not need to solve the bipartite matching to obtain δ-values. Also (2) and Property 4 imply that $|V - V_Q - V_L| = \mathrm{O}(n/\Delta)$, where V_L is the set of all leaf vertices. (This can be obtained by pruning leaf vertices and regarding V_Q vertices as new leaves.) Since we do not have to compute bipartite matchings for $v \in V_L \cup V_Q$, and this implies that $\sum_{v \in V} t(v) = \mathrm{O}(\sum_{v \in V - V_L - V_Q} t(v))$, which turned out to be $\mathrm{O}(\Delta^{2.5} \sum_{v \in V - V_L - V_Q} d''(v))$, where $d''(v) = |C(v) - V_L|$. Since $\sum_{v \in V - V_L - V_Q} d''(v) = |V - V_L - V_Q| + |V_Q| - 1 = \mathrm{O}(n/\Delta)$, we obtain $\sum_{v \in V - V_L - V_Q} t(v) = \mathrm{O}(\Delta^{1.5} n)$.

4 Label Compatibility and Flow-based Computation of δ

As mentioned in Section 3, we can achieve an efficient computation of δ-values by reusing the matching structures, which is one of keys of the running time $\mathrm{O}(n^{1.75})$ of Algorithm 1. In this section, we introduce another technique of the speedup of the computation of δ-values. The faster computation of δ-values is based on a maximum flow algorithm instead of a maximum matching algorithm used in Section 3. Seemingly, this sounds a bit strange, because the time complexity of the maximum flow problem is larger than the one of the bipartite matching problem. The trick is that the new flow-based computation uses a smaller network (graph) than the graph $G(u, v, -, b)$, which is used in the bipartite matching. To this end, we introduce the notion of label compatibility (Subsection 4.1), which enables us to treat several labels equivalently under the computation of δ-values (Subsection 4.2).

4.1 Label Compatibility and Neck/head Levels

Let $L_h = \{h, h+1, \ldots, \Delta - h, \Delta - h + 1\}$. Let T be a tree rooted at v, and $u \notin V(T)$. We say that T is *head-L_h-compatible* if $\delta((u,v),(a,b)) = \delta((u,v),(a',b))$ for all $a, a' \in L_h$ and $b \in L_0$ with $|a - b| \geq 2$ and $|a' - b| \geq 2$. Analogously, we say that T is neck-L_h-compatible if $\delta((u,v),(a,b)) = \delta((u,v),(a,b'))$ for all $a \in L_0$ and $b, b' \in L_h$ with $|a - b| \geq 2$ and $|a - b'| \geq 2$. The neck and head levels of T are defined as follows:

Definition 1 *Let T be a tree rooted at v, and $u \notin V(T)$.*
(i) *The* neck level (*resp.,* head level) *of T is* 0 *if T is neck-L_0-compatible* (*resp., head-L_0-compatible*).
(ii) *The* neck level (*resp.,* head level) *of T is h* (≥ 1) *if T is not neck-L_{h-1}-compatible* (*resp., head-L_{h-1}-compatible*) *but neck-L_h-compatible* (*resp., head-L_h-compatible*).

An intuitive explanation of neck-L_h-compatibility (resp., head-L_h-compatibility) of T is that if for $T(u,v)$, a label in L_h is assigned to v (resp., u) under $(\Delta+1)$-$L(2,1)$-labeling of $T(u,v)$, the label can be replaced with another label in L_h without violating a proper $(\Delta+1)$-$L(2,1)$-labeling; labels in L_h are compatible. The neck and head levels of T represent the bounds of L_h-compatibility of T.

For the relationship between the neck/head levels and the tree size, we can show the following lemma, although the proof has been moved to the Appendix, due to space limitation:

Lemma 2 *Let T' be a subtree of T. If $|V(T')| \leq (\Delta - 3 - 2h)^{h/2} - 1$ and $\Delta - 2h \geq 10$, then the head level and neck level of T' are both at most h.*

By this lemma, we obtain the following theorem:

Theorem 3 For a tree T, both the head and neck levels of T are $\mathrm{O}(\log |V(T)| / \log \Delta)$.

4.2 Flow-based Computation of δ

Now we are ready to explain the faster computation of δ-values. Recall that $\delta((u, v), (a, b)) = 1$ holds if there exists a matching of $G(u, v, a, b)$ in which all $C(v)$ vertices are just matched; which vertex is matched to a vertex in X does not matter. From this fact, we can treat vertices in X corresponding to L_h equally in computing δ, if T is neck- and head-L_h-compatible. The idea of the fast computation of δ-values is that, by bundling compatible vertices in X of G, we reduce the size of a graph (or a network) to compute the assignments of labels, which is no longer the maximum matching; the maximum flow.

Algorithm Maintain-Matching in Section 3 computes δ-values not by solving the maximum matchings of $G(u, v, a, b)$ for all pairs of a and b but by finding a maximum matching M of $G(u, v, -, b)$ once and then searching M-alternating paths. In the new flow-based computation, we adopt the same strategy; for a tree $T(v)$ whose head and neck levels are at most $h(v)$, we prepare not a network for a specific pair of (a, b), say $\mathcal{N}(u, v, a, b)$, but a general network $\mathcal{N}(u, v, -, b) = (\{s, t\} \cup C(v) \cup X_{h(v)}, E(v) \cup E_X \cup E_\delta, cap)$, where $X_{h(v)} = \{c \in (L_0 - L_{h(v)}) \cup \{h(v)\}\}$, $E(v) = \{(s, w) \mid w \in C(v)\}$, $E_X = \{(c, t) \mid c \in X_{h(v)}\}$, $E_\delta = \{(w, c) \mid w \in C(v), c \in X_{h(v)}\}$, and $cap(e)$ function is defined as follows: for $\forall e \in E(v)$, $cap(e) = 1$, for $e = (w, c) \in E_\delta$, $cap(e) = 1$ if $\delta((v, w), (b, c)) = 1$, $cap(e) = 0$ otherwise, and for $e = (c, t) \in E_X$, $cap(e) = 1$ if $c \neq h(v)$, $cap(e) = |L_{h(v)} - \{b, b + 1, b - 1\}|$ if $c = h(v)$. See Figure 1 for an example of $\mathcal{N}(u, v, -, b)$ in Appendix.

For a maximum flow $\psi : e \to R^+$, we redefine X' as $\{c \in X_h \mid cap((c, t)) - \psi((c, t)) \geq 1\}$. By the flow integrality and arguments similarly to Properties 1 and 2, we can obtain the following properties:

Lemma 4 *If $\mathcal{N}(u, v, -, b)$ has no flow of size $d'(v)$, then $\delta((u, v), (i, b)) = 0$ for any label i.* □

Lemma 5 *$\delta((u, v), (i, b)) = 1$ if and only if vertex i can be reached by an ψ-alternating path from some vertex in X' in $\mathcal{N}(u, v, -, b)$.* □

Here, ψ-alternating path is defined as follows: Given a flow ψ, a path is called *ψ-alternating* if its edges alternately satisfy $cap(e) - \psi(e) \geq 1$ and $\psi(e) \geq 1$. By these lemmas, we can obtain $\delta((u, v), (*, b))$-values for b by solving the maximum flow of $\mathcal{N}(u, v, -, b)$ once and then applying a single graph search.

The current fastest algorithm for solving the maximum flow problem runs in $O(\min\{m^{1/2}, n^{2/3}\} m \log(n^2/m) \log U) = O(n^{2/3} m \log n \log U)$ time, where U, n and m are the maximum capacity of edges, the number of vertices and edges, respectively [6]. Thus the running time of calculating $\delta((u, v), (a, b))$ for a pair (a, b) is

$$O((h(v) + d''(v))^{2/3}(h(v)d''(v)) \log(h(v) + d''(v)) \log \Delta) = O(\Delta^{2/3}(h(v)d''(v)) \log^2 \Delta),$$

since $h(v) \leq \Delta$ and $d''(v) \leq \Delta$ (recall that $d''(v) = |C(v) - V_L|$). By using a similar technique of updating matching structures introduced in [8], we can obtain $\delta((u, v), (*, b))$ in $O(\Delta^{2/3}(h(v)d''(v)) \log^2 \Delta) + O(h(v)d''(v)) = O(\Delta^{2/3}(h(v)d''(v)) \log^2 \Delta)$ time. Since the number of candidates for b is also bounded by $h(v)$ from the neck/head level property, we have the following lemma.

Lemma 6 *$\delta((u, v), (*, *))$ can be computed in $O(\Delta^{2/3}(h(v))^2 d''(v) \log^2 \Delta)$ time, i.e., $t(v) = O(\Delta^{2/3}(h(v))^2 d''(v) \log^2 \Delta)$.* □

Combining this with $\sum_{v \in V_\delta} d''(v) = O(n/\Delta)$ shown in Section 3, we can show that the total running time for computing $L(2, 1)$-labeling number is $O(n(\max\{h(v)\})^2(\Delta^{-1/3} \log^2 \Delta))$. By applying Theorem 3, we have the following theorem:

Theorem 7 *For trees, the $L(2, 1)$-labeling problem can be solved in $O(\min\{n \log^2 n, \Delta^{1.5} n\})$ time. Furthermore, if $n = O(\Delta^{poly(\log \Delta)})$, it can be solved in $O(n)$ time.* □

Only by directly applying Theorem 3 (actually Lemma 2), we obtain much faster running time than the previous one. In the following section, we present a linear time algorithm, in which Lemma 2 is used in a different way.

5 Proof of Linear Running Time

As mentioned in Section 3, in [8], one of keys of the achievement of the running time $O(\Delta^{1.5}n) = O(n^{1.75})$ is equation $\sum_{v \in V_\delta} d''(v) = O(n/\Delta)$, where V_δ is the set of vertices in which δ-values should be computed via the matching-based algorithm; since the computation of δ-values for each v is done in $O(\Delta^{2.5}d''(v))$ time, it takes $\sum_{v \in V_\delta} O(\Delta^{2.5}d''(v)) = O(\Delta^{1.5}n)$ time in total. This equation is derived from the fact that in leaf vertices we do not need to solve the matching to compute δ-values, and any vertex with height 1 has $\Delta - 1$ leaves as its children after the preprocessing operation.

In our new algorithm, we generalize this idea: By replacing leaf vertices with subtrees with size at least Δ^4 in the above argument, we can obtain $\sum_{v \in V_\delta} d''(v) = O(n/\Delta^4)$, and in total, the running time $\sum_{v \in V_\delta} O(\Delta^{2.5}d''(v)) = O(n)$ is roughly achieved. Actually, this argument contains a cheating, because a subtree with size at most Δ^4 is not always connected to a major vertex, whereas a leaf is, which is well utilized to obtain $\sum_{v \in V_\delta} d''(v) = O(n/\Delta)$. Also, whereas we can neglect leaves to compute δ-values, we cannot neglect such subtrees. We resolve these problems by best utilizing the properties of neck/head levels and the maximum flow techniques introduced in Section 4.

5.1 Efficient Assignment of Labels for Computing δ

In this section, by compiling observations and techniques for assigning labels in the computation of $\delta((u, v), (*, *))$ for $v \in V$, given in Sections 2–4, we will design an algorithm to run in linear time within the DP framework. Throughout this section, we assume that an input tree T satisfies Properties 3 and 4. Below, we first partition the vertex set V into five types of subsets defined later, and give a linear time algorithm for computing the value of δ functions, specified for each type.

We here start with defining such five types of subsets V_i ($i = 1, \ldots, 5$). Throughout this section, for a tree T', we may denote $|V(T')|$ simply by $|T'|$. Let V_M be the set of vertices $v \in V$ such that $T(v)$ is a "maximal" subtree of T with $|T(v)| \leq \Delta^5$; i.e., for the parent u of v, $|T(u)| > \Delta^5$. Divide V_M into two sets $V_M^{(1)} := \{v \in V_M \mid |T(v)| \geq (\Delta - 19)^4\}$ and $V_M^{(2)} := \{v \in V_M \mid |T(v)| < (\Delta - 19)^4\}$ (notice that $V_L \subseteq \cup_{v \in V_M} V(T(v))$). Define $\tilde{d}(v) := |C(v) - V_M^{(2)}|$ $(= d'(v) - |C(v) \cap V_M^{(2)}|)$. Let

$$
\begin{aligned}
V_1 &:= \cup_{v \in V_M} V(T(v)),\\
V_2 &:= \{v \in V - V_1 \mid \tilde{d}(v) \geq 2\},\\
V_3 &:= \{v \in V - V_1 \mid \tilde{d}(v) = 1, C(v) \cap (V_M^{(2)} - V_L) = \emptyset\},\\
V_4 &:= \{v \in V - (V_1 \cup V_3) \mid \tilde{d}(v) = 1, \textstyle\sum_{w \in C(v) \cap (V_M^{(2)} - V_L)} |T(w)| \leq \Delta(\Delta - 19)\},\\
V_5 &:= \{v \in V - (V_1 \cup V_3) \mid \tilde{d}(v) = 1, \textstyle\sum_{w \in C(v) \cap (V_M^{(2)} - V_L)} |T(w)| > \Delta(\Delta - 19)\}.
\end{aligned}
$$

Notice that $V = V_1 \cup V_2 \cup V_3 \cup V_4 \cup V_5$, and that $V_i \cap V_j = \emptyset$ for each i, j with $i \neq j$.

Here we describe an outline of the algorithm for computing $\delta((u, v), (*, *))$, $v \in V$, named Compute-$\delta(v)$, which can be regarded as a subroutine of the DP framework. In the subsequent subsections, we show that for each V_i, $\delta((u, v), (*, *))$, $v \in V_i$ can be computed in linear time in total; i.e., $O(\sum_{v \in V_i} t(v)) = O(n)$. Namely, we have the following theorem.

Theorem 8 *For trees, the $L(2,1)$-labeling problem can be solved in linear time.*

The proof for V_3 and the proofs of several lemmas have been moved to the Appendix.

5.2 Computation of δ-value for V_1

For each $v \in V_M$, we have $O(\sum_{w \in T(v)} t(w)) = O(|T(v)|)$, by Theorem 7 and $|T(v)| = O(\Delta^5)$. Hence, we have $O(\sum_{v \in V_1} t(v)) = O(|V_1|)$.

Algorithm 4 COMPUTE-$\delta(v)$

1: /** Assume that the head and neck levels of $T(v)$ are at most h. **/

2: If $v \in V_1 \cup V_2$, then for each $b \in (L_0 - L_h) \cup \{h\}$, compute $\delta((u, v), (*, b))$ by the max-flow computation in the network $\mathcal{N}(u, v, -, b)$ defined in Subsection 4.2.

3: If $v \in V_3$, execute the following procedure for each $b \in L_0$ in the case of $C(v) \cap V_L = \emptyset$, and for each $b \in \{0, \Delta + 1\}$ in the case of $C(v) \cap V_L \neq \emptyset$. /** Let w^* denote the unique child of v not in $V_M^{(2)}$.**/

3-1: If $|\{c \mid \delta((v, w^*), (b, c)) = 1\}| \geq 2$, then let $\delta((u, v), (*, b)) := 1$.

3-2: If $\{c \mid \delta((v, w^*), (b, c)) = 1\} = \{c^*\}$, then let $\delta((u, v), (c^*, b)) := 0$ and $\delta((u, v), (a, b)) := 1$ for all other labels $a \notin \{b - 1, b, b + 1\}$.

3-3: If $|\{c \mid \delta((v, w^*), (b, c)) = 1\}| = 0$, then let $\delta((u, v), (*, b)) := 0$.

4: If $v \in V_4 \cup V_5$, then similarly to the case of $v \in V_1 \cup V_2$, compute $\delta((u, v), (*, *))$ by the max-flow computation in a network such as $\mathcal{N}(u, v, -, b)$ specified for this case (details will be described in Subsec. 5.4 and 5.5).

5.3 Computation of δ-value for V_2

By Lemma 6, we can observe that $\sum_{v\in V_2} t(v) = O(\sum_{v\in V_2} \Delta^{2/3} d'(v) h^2 \log^2 \Delta) = O(\Delta^{8/3} \log^2 \Delta \sum_{v\in V_2} d'(v))$ (note that $h \leq \Delta$ and $d''(v) \leq d'(v)$). Now, we have $d'(v) \leq \tilde{d}(v) + \Delta \leq \Delta\tilde{d}(v)$. It follows that $\sum_{v\in V_2} t(v) = O(\Delta^{11/3} \log^2 \Delta \sum_{v\in V_2} \tilde{d}(v))$. Below, in order to show that $\sum_{v\in V_2} t(v) = O(n)$, we prove that $\sum_{v\in V_2} \tilde{d}(v) = O(n/\Delta^4)$.

By definition, there is no vertex whose all children are vertices in $V_M^{(2)}$, since if there is such a vertex v, then for each $w \in C(v)$, we have $|T(w)| < (\Delta - 19)^4$ and hence $|T(v)| < \Delta^5$, which contradicts the maximality of $T(w)$. It follows that each leaf vertex belongs to $V_M^{(1)}$ in the tree T' obtained from T by deleting all vertices in $V_1 - V_M^{(1)}$ (note that $V(T') = V_M^{(1)} \cup V_2 \cup V_3 \cup V_4 \cup V_5$). Hence,

$$\begin{aligned}
|V(T')| - 1 = |E(T')| &= \tfrac{1}{2} \textstyle\sum_{v\in V(T')} d_{T'}(v) \\
&= \tfrac{1}{2}(|V_M^{(1)}| + \textstyle\sum_{v\in V_2\cup V_3\cup V_4\cup V_5}(\tilde{d}(v) + 1) - 1) \\
&= \tfrac{1}{2}(|V_M^{(1)}| + \textstyle\sum_{v\in V_2}(\tilde{d}(v) + 1) + 2|V_3| + 2|V_4| + 2|V_5| - 1) \\
&\geq \tfrac{1}{2}|V_M^{(1)}| + \tfrac{3}{2}|V_2| + |V_3| + |V_4| + |V_5| - \tfrac{1}{2}
\end{aligned}$$

(the last inequality follows from $\tilde{d}(v) \geq 2$ for all $v \in V_2$). Thus, $|V_M^{(1)}| - 1 \geq |V_2|$. Therefore, we can observe that $\sum_{v\in V_2} \tilde{d}(v) = |E(T')| - |V_3| - |V_4| - |V_5| = |V_M^{(1)}| + |V_2| - 1 \leq 2|V_M^{(1)}| - 2$. Notice that the first equality follows from $|E(T')| = \sum_{v\in V_2\cup V_3\cup V_4\cup V_5} \tilde{d}(v) = \sum_{v\in V_2} \tilde{d}(v) + |V_3| + |V_4| + |V_5|$ and that the second equality follows from $|E(T')| = |V(T')| - 1 = |V_M^{(1)}| + |V_2| + |V_3| + |V_4| + |V_5| - 1$. It follows by $|V_M^{(1)}| = O(n/\Delta^4)$ that $\sum_{v\in V_2} \tilde{d}(v) = O(n/\Delta^4)$.

5.4 Computation of δ-value for V_4

We first claim that $|V_4| = O(n/\Delta)$. Since each leaf is incident to a major vertex by Property 3, note that

$$\text{for each } w \in V_M - V_L, \text{ we have } |T(w)| \geq \Delta. \tag{1}$$

Now, by definition, we have $C(v) \cap (V_M^{(2)} - V_L) \neq \emptyset$ and hence $|V_4| = O(n/\Delta)$, which is derived from the fact that the total number of descendants of a child in $V_M^{(2)} - V_L$ of each vertex in V_4 is at least $\Delta|V_4|$.

Here, we first observe several properties of vertices in V_4 before describing an algorithm for computing δ-values. Let $v \in V_4$. By (1) and $\sum_{w\in C(v)\cap(V_M^{(2)}-V_L)} |T(w)| \leq \Delta(\Delta - 19)$, we have

$$|C(v) \cap (V_M^{(2)} - V_L)| \leq \Delta - 19. \tag{2}$$

Intuitively, (2) indicates that the number of labels to be assigned for $V_M^{(2)} - V_L$ is relatively small. Now,

$$\text{the head and neck levels of } T(w) \text{ are at most 8 for each } w \in V_M^{(2)} \tag{3}$$

by Lemma 2 and $|T(w)| < (\Delta - 19)^4$ (note that we assume that $\Delta \geq 26$, since otherwise the original CK algorithm is already a linear time algorithm). Hence, we can observe that there are many possible feasible assignments for $V_M^{(2)} - V_L$; i.e., we can see that if we can assign labels to some restricted children of v properly, then there exists a proper assignment for the whole children of v, as observed in the following Lemma 9. For a label b, we divide $C(v) \cap (V_M^{(2)} - V_L)$ into two subsets $C_1(b) := \{w \in C(v) \cap (V_M^{(2)} - V_L) \mid \delta((v,w),(b,c)) = 1$ for all $c \in L_8 - \{b-1, b, b+1\}\}$ and $C_2(b) := \{w \in C(v) \cap (V_M^{(2)} - V_L) \mid \delta((v,w),(b,c)) = 0$ for all $c \in L_8 - \{b-1, b, b+1\}\}$ (notice that by (3), the neck level of $T(w)$, $w \in C(v) \cap (V_M^{(2)} - V_L)$, is at most 8, and hence we have $C(v) \cap (V_M^{(2)} - V_L) = C_1(b) \cup C_2(b)$). Let w^* be the unique child of v in $C(v) - V_M^{(2)}$. By the following property, we have only to consider the assignments for $\{w^*\} \cup C_2(b)$.

Lemma 9 *Let a and b be labels with $|b - a| \geq 2$ such that $b \in L_0$ if $C(v) \cap V_L = \emptyset$ and $b \in \{0, \Delta + 1\}$ otherwise. Then, $\delta((u,v),(a,b)) = 1$ if and only if there exists an injective assignment $g : \{w^*\} \cup C_2(b) \to L_0 - \{a, b-1, b, b+1\}$ such that $\delta((v,w),(b,g(w))) = 1$ for each $w \in \{w^*\} \cup C_2(b)$.*

Below, we show how to compute $\delta((u,v),(*,b))$ for a fixed b, where $b \in L_0$ if $C(v) \cap V_L = \emptyset$ and $b \in \{0, \Delta + 1\}$ otherwise. If $|C_2(b)| \geq 17$, then $\delta((u,v),(*,\ b)) = 0$ because in this case, there exists some $w \in C_2(b)$ to which any label in $L_0 - L_8$ cannot be assigned since $|L_0 - L_8| = 16$. Assume that $|C_2(b)| \leq 16$. There are the following three possible cases: (Case-1) $\delta((v,w^*),(b,c_i)) = 1$ for at least two labels $c_1, c_2 \in L_8$, (Case-2) $\delta((v,w^*),(b,c_1)) = 1$, for exactly one label $c_1 \in L_8$, and (Case-3) otherwise.

(Case-1) By assumption, for any a, $\delta((v,w^*),(b,c)) = 1$ for some $c \in L_8 - \{a\}$. This, together with Lemma 9, implies that we have only to check whether there exists an injective assignment $g : C_2(b) \to L_0 - L_8 - \{a, b-1, b, b+1\}$ such that $\delta((v,w),(b,g(w))) = 1$ for each $w \in C_2(b)$ (note that by definition, $\delta((v,w),(b,c)) = 0$ for any $w \in C_2(b)$, $c \in L_8$). According to Section 4.2, this can be done by utilizing the maximum flow computation on the subgraph $\mathcal{N}'$ of $\mathcal{N}(u,v,-,b)$ induced by $\{s,t\} \cup C_2(b) \cup X'$ where $X' = \{0, 1, \ldots, 7, \Delta - 6, \Delta - 5, \ldots, \Delta + 1\}$. Obviously, the size of $\mathcal{N}'$ is O(1) and it follows that its time complexity is O(1).

(Case-2) For all $a \neq c_1$, the value of $\delta((u,v),(a,b))$ can be computed similarly to Case-1. Consider the case where $a = c_1$. In this case, if $\delta((v,w^*),(b,c)) = 1$ holds, then it turns out that $c \in L_0 - L_8$. Hence, by Lemma 9, it suffices to check whether there exists an injective assignment $g : \{w^*\} \cup C_2(b) \to L_0 - L_8 - \{b-1, b, b+1\}$ such that $\delta((v,w),(b,g(w))) = 1$ for each $w \in \{w^*\} \cup C_2(b)$. Similarly to Case-1, this can be done in O(1) time, by utilizing the subgraph $\mathcal{N}''$ of $\mathcal{N}(u,v,-,b)$ induced by $\{s,t\} \cup (C_2(b) \cup \{w^*\}) \cup X'$.

(Case-3) By assumption, if $\delta((v,w^*),(b,c)) = 1$ holds, then it turns out that $c \in L_0 - L_8$. Similarly to the case of $a = c_1$ in Case-2, by using $\mathcal{N}''$, we can compute the values of $\delta((u,v),(*,b))$ in O(1) time.

Finally, we analyze the time complexity for computing $\delta((u,v),\ (*,*))$. It is dominated by that for computing $C_1(b)$, $C_2(b)$, and $\delta((u,v),(*,b))$ for each $b \in L_0$. By (3), we have $C_i(b) = C_i(b')$ for all $b, b' \in L_8$ and $i = 1, 2$. It follows that the computation for $C_1(b)$ and $C_2(b)$, $b \in L_0$ can be done in $O(|C(v) \cap (V_M^{(2)} - V_L)|)$ time. On the other hand, the values of $\delta((u,v),(*,b))$ can be computed in constant time in each case of Cases-1, 2 and 3 for a fixed b. Thus, we can observe that the time complexity for computing $\delta((u,v),(*,*))$ is $O(|C(v) \cap (V_M^{(2)} - V_L)| + \Delta) = O(\Delta)$. This together with $|V_4| = O(n/\Delta)$ implies that the values of $\delta((u,v),(*,*))$ for vertices in V_4 can be computed in $O(n)$ time.

5.5 Computation of δ-value for V_5

In the case of V_4, since the number of labels to be assigned is relatively small, we can properly assign labels to $C_1(b)$ even after determining labels for $\{w^*\} \cup C_2(b)$; we just need to solve the maximum flow of a small network in which vertices corresponding to $C_1(b)$ are omitted. In the case of V_5, however, the number of labels to be assigned is not small enough, and actually it is sometimes tight. Thus we need to assign labels carefully, that is, we need to solve the maximum flow problem in which the network contains vertices corresponding to $C_1(b)$. Instead of that, we can utilize the inequality $|V_5| \le \frac{n}{\Delta(\Delta-19)} = \mathrm{O}(n/\Delta^2)$, which is derived from $\sum_{w \in C(v)\cap(V_M^{(2)}-V_L)} |T(w)| > \Delta(\Delta-19)$.

Below, in order to show that $\mathrm{O}(\sum_{v\in V_5} t(v)) = \mathrm{O}(n)$, we prove that the values of $\delta((u,v),(*,b))$ can be computed in $\mathrm{O}(\Delta)$ time for a fixed b. A key is that the children $w \in C(v) \cap V_M^{(2)}$ of v can be classified into 2^{17} ($=\mathrm{O}(1)$) types, depending on its δ-values $(\delta((v,w),(b,i)) \mid i \in (L_0 - L_8) \cup \{8\})$, since for each such w, the head and neck levels of $T(w)$ are at most 8, as observed in (3). We denote the characteristic vector $(\delta((v,w),(b,i)) \mid i \in (L_0 - L_8) \cup \{8\})$ by (x_w). Furthermore, by the following lemma, we can see that $\delta((u,v),(*,b))$ can be obtained by checking $\delta((u,v),(a,b))$ for O(1) candidates of a, where we let w^* be the unique child of v not in $V_M^{(2)}$; namely, we have only to check $a \in (L_0 - L_8) \cup \{8\}$ if $|\{c \in L_8 - \{b-1,b,b+1\} \mid \delta((v,w^*),(b,c)) = 1\}| \ne 1$, and $a \in (L_0 - L_8) \cup \{c'\}$ where $c' \in L_8 - \{b-1,b,b+1,c^*\}$ if $\{c \in L_8 - \{b-1,b,b+1\} \mid \delta((v,w^*),(b,c)) = 1\} = \{c^*\}$.

Lemma 10 *If $\delta((u,v),(a_1,b)) \ne \delta((u,v),(a_2,b))$ for some $a_1, a_2 \in L_8 - \{b-1,b,b+1\}$ (say, $\delta((u,v),(a_1,b)) = 1$), then we have $\delta((v,w^*),(b,a_2)) = 1$ and $\delta((v,w^*),(b,a)) = 0$ for all $a \in L_8 - \{a_2, b-1, b, b+1\}$, and moreover, $\delta((u,v),(a,b)) = 1$ for all $a \in L_8 - \{a_2, b-1, b, b+1\}$.*

From these observations, it suffices to show that $\delta((u,v),(a,b))$ can be computed in $\mathrm{O}(\Delta)$ time for a fixed pair a,b. We will prove this by applying the maximum flow techniques observed in Section 4.2 to the network $\mathcal{N}'(u,v,a,b)$ with O(1) vertices, O(1) edges, and $\mathrm{O}(\Delta)$ units of capacity, defined as follows:

$$\mathcal{N}'(u,v,a,b) = (\{s,t\} \cup U_8 \cup \{w^*\} \cup X_8, E_8 \cup E'_X \cup E'_\delta, cap),$$

where $U_8 = \{(x_w) \mid w \in C(v)\}$, $X_8 = \{0,1,\ldots,7,\Delta-6,\Delta-5,\ldots,\Delta+1\} \cup \{8\}$, $E_8 = \{s\} \times (U_8 \cup \{w^*\})$, $E'_\delta = (U_8 \cup \{w^*\}) \times X_8$, $E'_X = X_8 \times \{t\}$. We note that $(x_w)_c = \delta((v,w),(b,c))$. For example, for w, vector $(x_w) = (101....1)$ means that w satisfies $(x_w)_0 = \delta((v,w),(b,0)) = 1, (x_w)_1 = \delta((v,w),(b,1)) = 0, (x_w)_2 = \delta((v,w),(b,2)) = 1, \ldots, (x_w)_{\Delta+1} = \delta((v,w),(b,\Delta+1)) = 1$. Notice that U_8 is a set of 0-1 vector with length 17, and its size is bounded by a constant. When we refer a vertex $i \in X_8$, we sometimes use c_i instead of i to avoid a confusion. For the vetex sets and the edge sets, its $cap(e)$ function is defined as follows: For $e = (s,(x)) \in \{s\} \times U_8 \subset E_8$, $cap(e) = |\{w \mid (x) = (x_w)\}|$. For $(s,w^*) \in E_8$, $cap(e) = 1$. If $a \in X_8 - \{c_8\}$, then for $e = (u',a) \in E'_\delta$ with $u' \in U_8 \cup \{w^*\}$, $cap(e) = 0$. For $e = ((x_w),c) \in U_8 \times (X_8 - \{a, c_8\}) \subset E'_\delta$, $cap(e) = (x_w)_c$, and for $e = (w^*,c) \in E'_\delta$ with $c \in X_8 - \{a,c_8\}$, $cap(e) = \delta((v,w^*),(b,c))$. For $e = ((x_w),c_8) \in U_8 \times \{c_8\} \subset E'_\delta$, $cap(e) = |L_8 - \{a,b-1,b,b+1\}|$ if $(x_w)_8 = 1$, and $cap(e) = 0$ otherwise. For $(w^*,c_8) \in E'_\delta$, $cap(e) = 1$ if $\delta((v,w^*),(b,c)) = 1$ for some $c \in L_8 - \{a,b-1,b,b+1\}$, and $cap(e) = 0$ otherwise. For $e = (c,t) \in (X_8 - \{c_8\}) \times \{t\} \subset E'_X$, $cap(e) = 1$. For $e = (c_8,t) \in E'_X$, $cap(e) = |L_8 - \{a,b-1,b,b+1\}|$.

This network is constructed differently from $\mathcal{N}(u,v,-,b)$ in two points. One is that in the new $\mathcal{N}'(u,v,a,b)$, not only label vertices but also $C(v)$ vertices are bundled to U_8. For each arc $e = ((x_w),c_8) \in E'_\delta$, $cap(e)$ is defined by $|L_8 - \{a,b-1,b,b+1\}|$ if $(x_w)_8 = 1$ and 0 otherwise. This follows from the neck level of $T(w)$ for $w \in C(v) - \{w^*\}$ is at most 8; i.e., we have $\delta((v,w),(b,c)) = \delta((v,w),(b,c'))$ for all $c,c' \in L_8 - \{b-1,b,b+1\}$. The other is that the arc (w^*,c_8) is set in a different way from the ones in $\mathcal{N}(u,v,-,b)$. Notice that neck level of $T(w^*)$ may not be at most 8; $cap((w^*,c_8)) = 1$ does not imply that $\delta((v,w^*),(b,c))$ is equal for any $c \in L_8$. Despite the difference of the definition of cap functions, we can see that $\delta((u,v),(a,b)) = 1$ if and only if there exists a maximum flow from s to t with size $d'(v)$ in $\mathcal{N}'(u,v,a,b)$. Indeed, even for a maximum flow ψ with size $d'(v)$ such that $\psi(w^*,c_8) = 1$ (say, $\delta((v,w^*),(b,c^*)) = 1$ for $c^* \in L_8 - \{a,b-1,b,b+1\}$), there exists an injective assignment $g : C(v) \to L_0 - \{a,b-1,b,b+1\}$ such that $\delta((v,w),(b,g(w))) = 1$ for each $w \in C(v)$, because we can assign injectively the remaining labels in L_8 (i.e., $L_8 - \{a,b-1,b,b+1,c^*\}$) to all vertices corresponding to x_w with $\psi((x_w),c_8) > 0$.

To construct $\mathcal{N}'(u,v,a,b)$, it takes $\mathrm{O}(d'(v))$ time. Since $\mathcal{N}'(u,v,a,b)$ has O(1) vertices, O(1) edges and at most Δ units of capacity, the maximum flow itself can be solved in $\mathrm{O}(\log\Delta)$ time. Thus, the values of $\delta((u,v),(*,b))$ can be computed in $\mathrm{O}(d'(v) + \log\Delta) = \mathrm{O}(\Delta)$ time.

References

[1] H. L. Bodlaender, T. Kloks, R. B. Tan and J. van Leeuwen. Approximations for λ-coloring of graphs. *The Computer Journal* 47, 193–204 (2004).

[2] T. Calamoneri. The $L(h,k)$-labelling problem: A survey and annotated bibliography. *The Computer Journal* 49, 585–608 (2006).

[3] G. J. Chang and D. Kuo. The $L(2,1)$-labeling problem on graphs. *SIAM J. Disc. Math.* 9, 309–316 (1996).

[4] J. Fiala, P. A. Golovach and J. Kratochvíl. Distance constrained labelings of graphs of bounded treewidth. *Proc. 32th ICALP*, 360–372 (2005).

[5] J. Fiala, T. Kloks and J. Kratochvíl. Fixed-parameter complexity of λ-labelings. *Discr. Appl. Math.* 113, 59–72 (2001).

[6] A. V. Goldberg and S. Rao. Beyond the flow decomposition barrier. *J. ACM* 45, pp. 783–797 (1998).

[7] J. R. Griggs and R. K. Yeh. Labelling graphs with a condition at distance 2. *SIAM J. Disc. Math.* 5, 586–595 (1992).

[8] T. Hasunuma, T. Ishii, H. Ono and Y. Uno. An $O(n^{1.75})$ algorithm for $L(2,1)$-labeling of trees. *Proc. 11th SWAT*, 185–197 (2008).

[9] F. Havet, B. Reed and J.-S. Sereni. $L(2,1)$-labelling of graphs. *Proc. 19th SIAM-SODA*, 621–630 (2008).

[10] J. E. Hopcroft, R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.* 2, 225–231 (1973).

[11] J. Kratchvíl, D. Kratsch and M. Liedloff. Exact algrithms for $L(2,1)$-labeling of graphs. *Proc. 32nd MFCS*, 513–524 (2007).

[12] W.-F. Wang. The $L(2,1)$-labelling of trees. *Discr. Appl. Math.* 154, 598–603 (2006).

[13] R. K. Yeh. A survey on labeling graphs with a condition at distance two. *Discr. Math.* 306, 1217–1231 (2006).

A APPENDIX

This appendix provides the proofs of the results that have been omitted due to space reasons. They may be read to the discretion of the program committee.

A.1 Figure in Section 4

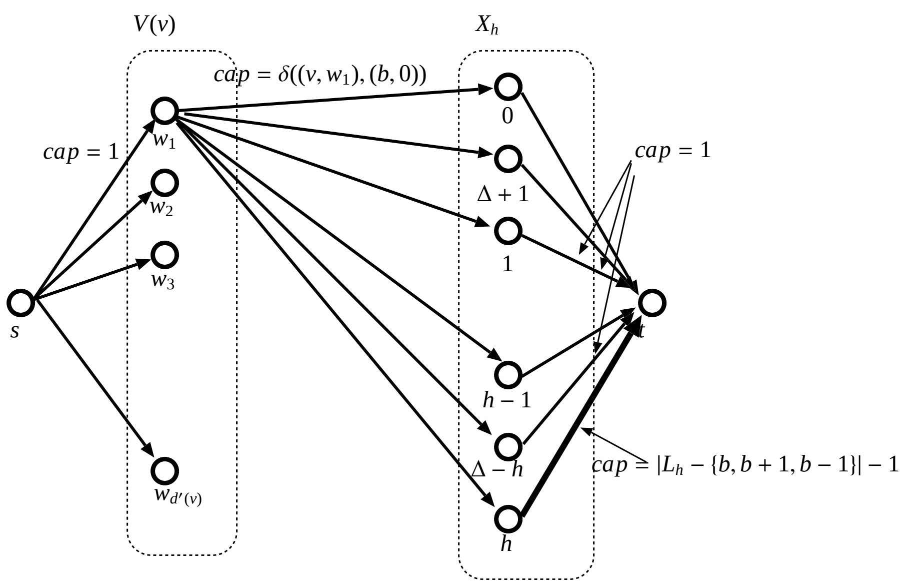

Figure 1: An example of $\mathcal{N}(u, v, -, b)$ where $h = h(v)$.

A.2 Proofs of Lemma 2 and Theorem 3

For a tree T' rooted at v, denote by $T' + (u, v)$ the tree obtained from T' by adding a vertex $u \notin V(T')$ and an edge (u, v). This is similar to $T(u, v)$ defined in Subsection 2.2, however, for $T(u, v)$, u is regarded as a virtual vertex, while for $T' + (u, v)$, u may be an existing vertex.

Proof of Lemma 2: When $h = 1, 2$, we have $|T'| \le \Delta - 6$ and hence $\Delta(T' + (u, v)) \le \Delta - 6$, where v denotes the root of T'. It follows that $T' + (u, v)$ can be labeled by using at most $\Delta(T' + (u, v)) + 3 \le \Delta - 3$ labels. Thus, in these cases, the head and neck levels are both 0.

Now, we assume by contradiction that this lemma does not hold. Let T_1 be such a counterexample with the minimum size, i.e., T_1 satisfies the following properties (4)–(7):

$$\cdot\ |T_1| \le (\Delta - 3 - 2h)^{h/2} - 1. \tag{4}$$

$\cdot$ The neck or head level of T_1 is at least $h + 1$. (5)

$\cdot$ $h \ge 3$ (from the arguments of the previous paragraph). (6)

$\cdot$ For each tree T' with $|T'| < |T_1|$, the lemma holds. (7)

By (5), there are two possible cases (Case-I) the head level of T_1 is at least $h + 1$ and (Case-II) the neck level of T_1 is at least $h + 1$. Let v_1 denote the root of T_1.

(Case-I) In this case, in $T_1 + (u, v_1)$ with $u \notin V(T_1)$, for some label b, there exist two labels $a, a' \in L_h$ with $|b - a| \geq 2$ and $|b - a'| \geq 2$ such that $\delta((u, v_1), (a, b)) = 1$ and $\delta((u, v_1), (a', b)) = 0$. Let f be a $(\Delta + 1)$-$L(2, 1)$-labeling with $f(u) = a$ and $f(v_1) = b$. If any child of v_1 does not have label a' in the labeling f, then the labeling obtained from f by changing the label for u from a to a' is also feasible, which contradicts $\delta((u, v_1), (a', b)) = 0$.

Consider the case where some child w of v_1 satisfies $f(w) = a'$. Then by $|T(w)| < |T_1|$ and (7), then the neck level of $T(w)$ is at most h. Hence, we have $\delta((v_1, w), (b, a')) = \delta((v_1, w), (b, a))$ $(= 1)$ by $a, a' \in L_h$. Let f_1 be a $(\Delta + 1)$-$L(2, 1)$-labeling on $T(w) + (v_1, w)$ achieving $\delta((v_1, w), (b, a)) = 1$. Now, note that any vertex $v \in C(v_1) - \{w\}$ satisfies $f(v) \notin \{a, a'\}$, since f is feasible. Thus, we can observe that the labeling f_2 satisfying the following (i)–(iii) is a $(\Delta+1)$-$L(2, 1)$-labeling on $T_1 + (u, v_1)$: (i) $f_2(u) := a'$, (ii) $f_2(v) := f_1(v)$ for all vertices $v \in V(T(w))$, and (iii) $f_2(v) := f(v)$ for all other vertices. This also contradicts $\delta((u, v_1), (a', b)) = 0$.

(Case-II) By the above arguments, we can assume that the head level of T_1 is at most h. In $T_1 + (u, v_1)$ with $u \notin V(T_1)$, for some label a, there exist two labels $b, b' \in L_h$ with $|b - a| \geq 2$ and $|b' - a| \geq 2$ such that $\delta((u, v_1), (a, b)) = 1$ and $\delta((u, v_1), (a, b')) = 0$. Similarly to Case-I, we will derive a contradiction by showing that $\delta((u, v_1), (a, b')) = 1$. Now there are the following three cases: (II-1) there exists such a pair b, b' with $b' = b - 1$, (II-2) the case (II-1) does not hold and there exists such a pair b, b' with $b' = b + 1$, and (II-3) otherwise.

First we show that we have only to consider the case of (II-1); namely, we can see that if

$$\text{(II-1) does not occur for any } a, b \text{ with } a \notin \{b-2, b-1, b, b+1\} \text{ and } \{b-1, b\} \subseteq L_h, \tag{8}$$

then (II-2) does not occur and that (II-3) does not occur. Assume that (8) holds. Consider the case (II-2). Then, since we have $\delta((u, v_1), (\Delta+1-a, \Delta+1-b)) = \delta((u, v_1), (a, b)) = 1$ and $\delta((u, v_1), (\Delta+1-a, \Delta-b)) = \delta((u, v_1), (a, b + 1)) = 0$, which contradicts (8). Consider the case (II-3); there is no pair b, b' such that $|b-b'| = 1$. Namely, in this case, for some $a \in L_{h+2}$, we have $\delta((u, v_1), (a, h)) \neq \delta((u, v_1), (a, \Delta+1-h))$ and $\delta((u, v_1), (a, b_1)) = \delta((u, v_1), (a, b_2))$ only if (i) $b_i \leq a - 2$, $i = 1, 2$ or (ii) $b_i \geq a + 2$, $i = 1, 2$. Then, since the head level of T_1 is at most h, it follows by $a \in L_{h+2}$ that $\delta((u, v_1), (a, h)) = \delta((u, v_1), (\Delta + 1 - a, h))$. By $\delta((u, v_1), (\Delta+1-a, h)) = \delta((u, v_1), (a, \Delta+1-h))$, we have $\delta((u, v_1), (a, h)) = \delta((u, v_1), (a, \Delta+1-h))$, a contradiction.

Below, in order to show (8), we consider the case of $b' = b - 1$. Let f be a $(\Delta + 1)$-$L(2, 1)$-labeling with $f(u) = a$ and $f(v_1) = b$. We first start with the labeling f, and change the label for v_1 from b to $b - 1$. Let f_1 denote the resulting labeling. If f_1 is feasible, then it contradicts $\delta((u, v_1), (a, b - 1)) = 0$. Here, we assume that f_1 is infeasible, and will show how to construct another $(\Delta + 1)$-$L(2, 1)$-labeling by changing the assignments for vertices in $V(T_1) - \{v_1\}$. Notice that since f_1 is infeasible, there are

$$\text{some child } w \text{ of } v_1 \text{ with } f_1(w) = b - 2, \text{ or} \tag{9}$$

$$\text{some grandchild } x \text{ of } v_1 \text{ with } f_1(x) = b - 1. \tag{10}$$

Now we have the following claim.

Claim 1 *Let f' be a $(\Delta+1)$-$L(2,1)$-labeling on T_1 and $T(v)$ be a subtree of T_1. There are at most $\Delta-2h-4$ children w of v with $f'(w) \in L_h$ and $|T(w)| \geq (\Delta - 2h + 1)^{(h-2)/2}$.*

Proof. Let $C'(v)$ be the set of children w of v with $f'(w) \in L_h$ and $|T(w)| \geq (\Delta-2h+1)^{(h-2)/2}$. If this claim would not hold, then we would have $|T_1| \geq |T(v)| \geq 1+\sum_{w \in C'(v)} |T(w)| \geq 1+(\Delta-2h-3)(\Delta-2h+1)^{(h-2)/2} > 1 + (\Delta - 2h - 3)^{h/2} > |T_1|$, a contradiction. □

This claim indicates that given a feasible labeling f' on T', for each vertex $v \in V(T')$, there exist at least two labels $\ell_1, \ell_2 \in L_h$ such that ℓ_i is not assigned to any vertex in $\{v, p(v)\} \cup C(v)$ (i.e., $\ell_i \notin \{f'(v') \mid v' \in \{v, p(v)\} \cup C(v)\}$) or assigned to a child $w_i \in C(v)$ with $|T(w_i)| \leq (\Delta - 2h + 1)^{(h-2)/2} - 1$, since

$|L_h - \{f'(p(v)), f'(v)-1, f'(v), f'(v)+1\}| \geq \Delta - 2h - 2$, where $p(v)$ denotes the parent of v. For each vertex $v \in V(T_1)$, denote such labels by $\ell_i(v; f')$ and such children by $c_i(v; f')$ (if exists) for $i = 1, 2$. We note that by (7), if $c_i(v; f')$ exists, then the head and neck levels of $T(c_i(v; f'))$ are at most $h-2$.

First consider the case where the vertex of (9) exists; denote such vertex by w_1. We consider this case by dividing into two cases (II-1-1) $b \geq h+2$ and (II-1-2) $b \leq h+1$, i.e., $b = h+1$ by $b-1 \in L_h$.

(II-1-1) Suppose that we have $\ell_1(v_1; f) \neq b-2$, and $c_1(v_1; f)$ exists (other cases can be treated similarly). By (7), the head and neck levels of $T(w)$ are at most h for each $w \in C(v_1)$, and especially, the head and neck levels of $T(c_1(v_1; f))$ are at most $h-2$. Hence, we have

$$\begin{aligned} \delta((v_1, w_1), (b, b-2)) &= \delta((v_1, w_1), (b, \ell_1(v_1; f))) \\ &= \delta((v_1, w_1), (b-1, \ell_1(v_1; f))), \qquad (11) \\ \delta((v_1, c_1(v_1; f)), (b, \ell_1(v_1; f))) &= \delta((v_1, c_1(v_1; f)), (b-1, \ell_1(v_1; f))) \\ &= \delta((v_1, c_1(v_1; f)), (b-1, b+1)), \qquad (12) \end{aligned}$$

since $\ell_1(v_1; f) \notin \{b-2, b-1, b, b+1\}$ and $\{b-2, b-1, b, \ell_1(v_1; f)\} \subseteq L_h$ (note that in the case where $c_1(v_1; f)$ does not exist, (12) is not necessary). Notice that $b+1 \notin L_h$ may hold, however we have $b+1 \in L_{h-1}$ by $b \in L_h$. By these observations, there exits labelings f'_1 and f'_2 on $T(w_1) + (v_1, w_1)$ and $T(c_1(v_1; f)) + (v_1, c_1(v_1; f))$, achieving $\delta((v_1, w_1), (b-1, \ell_1(v_1; f))) = 1$ and $\delta((v_1, c_1(v_1; f)), (b-1, b+1)) = 1$, respectively. Let f^* be the labeling such that $f^*(v_1) = b-1$, $f^*(v) = f'_1(v)$ for all $v \in V(T(w_1))$, $f^*(v) = f'_2(v)$ for all $v \in V(T(c_1(v_1; f)))$, and $f^*(v) = f(v)$ for all other vertices.

(II-1-2) In this case, we have $h+2, h+3 \in L_h$ by $\Delta - 2h \geq 10$. Now we have the following claim.

Claim 2 *For $T(w_1) + (v_1, w_1)$, we have $\delta((v_1, w_1), (h, h+2)) = 1$.*

Proof. Let f_1 be the labeling such that $f_1(v_1) := h$, $f_1(w_1) := h+2$, and $f_1(v) := f(v)$ for all other vertices v. Assume that f_1 is infeasible to $T(w_1) + (v_1, w_1)$ since otherwise the claim is proved. Hence, (A) there exist some child x of w_1 with $f_1(x) \in \{h+2, h+3\}$ or (B) some grandchild y of w_1 with $f_1(y) = h+2$ (note that any child x' of w_1 has neither label h nor $h+1$ by $f(w_1) = h-1$ and $f(v_1) = h+1$).

First, we consider the case where vertices of (A) exist. Suppose that there are two children $x_1, x_2 \in C(w_1)$ with $f_1(x_1) = h+2$ and $f_1(x_2) = h+3$, we have $\{\ell_1(w_1; f), \ell_2(w_1; f)\} \cap \{h+2, h+3\} = \emptyset$, and both of $c_1(w_1; f)$ and $c_2(w_1; f)$ exist (other cases can be treated similarly). Let $b'' \in L_h - \{h, h+1, h+2, h+3, h+4, \ell_1(w_1; f)-1, \ell_1(w_1; f), \ell_1(w_1; f)+1, \ell_2(w_1; f)-1, \ell_2(w_1; f), \ell_2(w_1; f)+1\}$ (such b'' exists by $\Delta + 2 - 2h \geq 12$).

Now by (7), the head and neck levels of $T(x_i)$ (resp., $T(c_i(w_1; f))$) is at most h (resp., $h-2$) for $i = 1, 2$. Hence, we have

$$\begin{aligned} \delta((w_1, x_1), (h-1, h+2)) &= \delta((w_1, x_1), (h-1, \ell_1(w_1; f))) \qquad (13) \\ \delta((w_1, c_1(w_1; f)), (h-1, \ell_1(w_1; f))) &= \delta((w_1, c_1(w_1; f)), (b'', \ell_1(w_1; f))) \\ &= \delta((w_1, c_1(w_1; f)), (b'', h-2)) \\ &= \delta((w_1, c_1(w_1; f)), (h+2, h-2)), \qquad (14) \\ \delta((w_1, x_2), (h-1, h+3)) &= \delta((w_1, x_2), (h-1, \ell_2(w_1; f))) \qquad (15) \\ \delta((w_1, c_2(w_1; f)), (h-1, \ell_2(w_1; f))) &= \delta((w_1, c_2(w_1; f)), (b'', \ell_2(w_1; f))) \\ &= \delta((w_1, c_2(w_1; f)), (b'', h-1)) \\ &= \delta((w_1, c_2(w_1; f)), (h+2, h-1)), \qquad (16) \end{aligned}$$

since $\{\ell_1(w_1; f), \ell_2(w_1; f)\} \cap \{h-2, h-1, h, h+1, h+2, h+3\} = \emptyset$. Notice that $h-2 > 0$ by (6). By (13)–(16), there exist $(\Delta+1)$-$L(2,1)$-labelings f'_1, f'_2, f'_3, and f'_4 on $T(x_1) + (w_1, x_1)$, $T(c_1(w_1; f)) + (w_1, c_1(w_1; f))$, $T(x_2) + (w_1, x_2)$, and $T(c_2(w_1; f)) + (w_1, c_2(w_1; f))$, achieving $\delta((w_1, x_1), (h-1, \ell_1(w_1; f))) = 1$, $\delta((w_1,$

$c_1(w_1; f)), (h+2, h-2)) = 1$, $\delta((w_1, x_2), (h-1, \ell_2(w_1; f)))) = 1$, and $\delta((w_1, c_2(w_1; f)), (h+2, h-1)) = 1$, respectively. Let f_2 be the labeling on $T(w_1) + (v_1, w_1)$ such that $f_2(v_1) = h$, $f_2(w_1) = h+2$, $f_2(v) = f_1'(v)$ for all $v \in V(T(x_1))$, $f_2(v) = f_2'(v)$ for all $v \in V(T(c_1(w_1; f)))$, $f_2(v) = f_3'(v)$ for all $v \in V(T(x_2))$, $f_2(v) = f_4'(v)$ for all $v \in V(T(c_2(w_1; f)))$, and $f_2(v) = f(v)$ for all other vertices. Observe that we have $f_2(x_1) = \ell_1(w_1; f)$, $f_2(c_1(w_1; f)) = h-2$, $f_2(x_2) = \ell_2(w_1; f)$, and $f_2(c_2(w_1; f)) = h-1$, and $f_2(x) \notin \{h, h+1, h+2, h+3\}$ for all $x \in C(w_1)$, every two labels in $C(w_1)$ are pairwise disjoint, and f_2 is a $(\Delta+1)$-$L(2,1)$-labeling on each subtree $T(x)$ with $x \in C(w_1)$.

Assume that f_2 is still infeasible. Then, there exists some grandchild y of w_1 with $f_2(y) = h+2$. Observe that from $f(w_1) = h-1$, no sibling of such a grandchild y has label $h-1$ in the labeling f_2, while such y may exist in the subtree $T(x)$ with $x \in C(w_1) - \{c_1(w_1; f), c_2(w_1; f)\}$. Also note that for the parent $x_p = p(y)$ of such y, we have $f(x_p) \notin \{h-2, h-1, h\}$. Suppose that $\ell_1(x_p; f_2) \neq h+2$ holds and $c_1(x_p; f_2)$ exists (other cases can be treated similarly). Now, by (7), the neck level of $T(y)$ (resp., $T(c_1(x_p; f_2)))$ is at most h (resp., $h-2$). Hence, we have $\delta((x_p, y), (f_2(x_p), h+2)) = \delta((x_p, y), (f_2(x_p), \ell_1(x_p; f_2)))$ and $\delta((x_p, c_1(x_p; f_2)), (f_2(x_p), \ell_1(x_p; f_2))) = \delta((x_p, c_1(x_p; f_2)), (f_2(x_p), h-1)))$. It follows that there exist $(\Delta+1)$-$L(2,1)$-labelings f_1'' and f_2'' on $T(y) + (x_p, y)$ and $T(c_1(x_p; f_2)) + (x_p, c_1(x_p; f_2))$ which achieves $\delta((x_p, y), (f_2(x_p), \ell_1(x_p; f_2))) = 1$ and $\delta((x_p, c_1(x_p; f_2)), (f_2(x_p), h-1))) = 1$, respectively. It is not difficult to see that the labeling f'' such that $f''(v) = f_1''(v)$ for all $v \in V(T(y))$, $f''(v) = f_2''(v)$ for all $v \in V(T(c_1(x_p; f_2)))$, and $f''(v) = f_2(v)$ for all other vertices is a $(\Delta+1)$-$L(2,1)$-labeling on $T(x_p) + (w_1, x_p)$.

Thus, by repeating these observations for each grandchild y of w_1 with $f_2(y) = h+2$, we can obtain a $(\Delta+1)$-$L(2,1)$-labeling f_3 on $T(w_1) + (v_1, w_1)$ with $f_3(v_1) = h$ and $f_3(w_1) = h+2$. □

Let f^* be the labeling such that $f^*(v) = f_3(v)$ for all $v \in \{v_1\} \cup V(T(w_1))$ and $f^*(v) = f(v)$ for all other vertices.

Thus, in both cases (II-1-1) and (II-1-2), we have constructed a labeling f^* such that we have $f^*(u) = a$, $f^*(v_1) = b-1$, and $f^*(w) \notin \{a, b-2, b-1, b\}$ for all $w \in C(v_1)$, every two labels in $C(v_1)$ are pairwise disjoint, and f^* is a $(\Delta+1)$-$L(2,1)$-labeling on each subtree $T(w)$ with $w \in C(v_1)$.

Assume that f^* is still infeasible. Then, there exists some grandchild x of v_1 of (10). Notice that for each vertex $v \in \{p(x)\} \cup V(T(x))$, we have $f^*(v) = f(v)$ from the construction; $f^*(p(x)) \notin \{b-1, b, b+1\}$ and $f^*(x') \neq b$ for any sibling x' of x. Moreover, by (7), the neck level of $T(x)$ is at most h; $\delta((p(x), x), (f^*(p(x)), b-1)) = \delta((p(x), x), (f^*(p(x)), b)) = 1$. Hence, there exists a $(\Delta+1)$-$L(2,1)$-labeling f' on $T(x) + (p(x), x)$ which achieves $\delta((p(x), x), (f^*(p(x)), b)) = 1$. It follows that the labeling f'' such that $f''(v) = f'(v)$ for all $v \in V(T(x))$ and $f''(v) = f^*(v)$ otherwise, is a $(\Delta+1)$-$L(2,1)$-labeling on $T(p(x)) + (v_1, p(x))$. Thus, by repeating these observations for each grandchild of v_1 of (10), we can obtain a $(\Delta+1)$-$L(2,1)$-labeling f^{**} for $T_1 + (u, v_1)$ with $f^{**}(u) = a$ and $f^{**}(v_1) = b-1$. This contradicts $\delta((u, v_1), (a, b-1)) = 0$. □

Proof of Theorem 3: The case of $\Delta = O(\log n / \log \Delta)$ is clear. Consider the case where $\Delta > \frac{8 \log n}{\log(\Delta/2)} + 6$. Then, for $h = \frac{2 \log n}{\log(\Delta/2)}$, we have

$$\begin{aligned}(\Delta - 3 - 2h)^{\frac{h}{2}} &> \left\{\frac{\Delta}{2} + \left(\frac{4\log n}{\log\frac{\Delta}{2}} + 3\right) - 3 - \frac{4\log n}{\log\frac{\Delta}{2}}\right\}^{\frac{\log n}{\log\frac{\Delta}{2}}} \\ &= n.\end{aligned}$$

Now note that $\Delta - 2h > \frac{4 \log n}{\log(\Delta/2)} + 6 \geq 10$. Hence, by Lemma 2, it follows that the head and neck levels of T are both at most $\frac{2 \log n}{\log(\Delta/2)}$. □

A.3 Proofs of Lemmas in Section 5

Proof of Lemma 9: The only if part is clear. We show the if part. Assume that there exists an injective assignment $g_1 : \{w^*\} \cup C_2(b) \to L_0 - \{a, b-1, b, b+1\}$ such that $\delta((v, w), (b, g_1(w))) = 1$ for each

$w \in \{w^*\} \cup C_2(b)$. Notice that by definition of $C_2(b)$, all $w \in C_2(b)$ satisfies $g_1(w) \in L_0 - L_8$. Hence, there exist at least $|L_8 - \{a, b-1, b, b+1, g_1(w^*)\}|$ $(= \Delta - 19)$ labels which are not assigned by g_1. By (2), we can assign such remaining labels to all vertices in $C_1(b)$ injectively; let g_2 be the resulting labeling on $C_1(b)$. Notice that for all $w \in C_1(b)$, we have $\delta((v, w), (b, g_2(w))) = 1$ by definition of $C_1(b)$ and $g_2(w) \in L_8$. It follows that the function $g_3 : C_1(b) \cup C_2(b) \cup \{w^*\} \to L_0 - \{a, b-1, b, b+1\}$ such that $g_3(w) = g_1(w)$ for all $w \in C_2(b) \cup \{w^*\}$ and $g_3(w) = g_2(w)$ for all $w \in C_1(b)$ is injective and satisfies $\delta((v, w), (b, g_3(w))) = 1$ for all $w \in C_1(b) \cup C_2(b)$. Thus, if $C(v) \cap V_L = \emptyset$, then we have $\delta((u, v), (a, b)) = 1$.

Consider the case where $C(v) \cap V_L \neq \emptyset$. Let $b = 0$ without loss of generality. Then by Property 3, v is major. Hence, $|C(v) \cap V_L| = \Delta - 1 - |C_1(0)| - |C_2(0)| - |\{w^*\}|$. Notice that the number of the remaining labels (i.e., labels not assigned by g_3) is $|L_0 - \{0, 1, a\} - C_1(0) - C_2(0) - \{w^*\}| = \Delta - 2 - |C_1(0)| - |C_2(0)|$. Hence, we can see that by assigning the remaining labels to vertices in $C(v) \cap V_L$ injectively, we can obtain a proper labeling; $\delta((u, v), (a, b)) = 1$ holds also in this case. □

Proof of Lemma 10: Let f be a labeling on $(u, v) + T(v)$ with $f(u) = a_1$ and $f(v) = b$, achieving $\delta((u, v), (a_1, b)) = 1$. By $\delta((u, v), (a_2, b)) = 0$, there exists a child w_1 of v with $f(w_1) = a_2$, since otherwise the labeling from f by changing the label for u from a_1 to a_2 would be feasible.

Assume by contradiction that $w_1 \neq w^*$ (i.e., $w_1 \in V_M^{(2)}$). Then the neck level of $T(w_1)$ is at most 8 and we have $\delta((v, w_1), (b, a_1)) = \delta((v, w_1), (b, a_2)) = 1$ by $a_1, a_2 \in L_8$. This indicates that $\delta((u, v), (a_2, b)) = 1$ would hold. Indeed, the function $g : C(v) \to L_0 - \{a_2, b-1, b, b+1\}$ such that $g(w_1) = a_1$ and $g(w') = f(w')$ for all other children w' of v, is injective and satisfies $\delta((v, w), (b, g(w))) = 1$ for all $w \in C(v)$.

Hence, we have $w_1 = w^*$. Note that $\delta((v, w^*), (b, a_2)) = 1$ since f is feasible. Then, assume by contradiction that some $a_3 \in L_8 - \{a_2, b-1, b, b+1\}$ satisfies $\delta((v, w^*), (b, a_3)) = 1$ ($a_3 = a_1$ may hold). Suppose that there exists a child w_2 of v with $f(w_2) = a_3$ (other cases can be treated similarly). Notice that the neck level of $T(w_2)$ is at most 8 and $\delta((v, w_2), (b, a_1)) = \delta((v, w_2), (b, a_3)) = 1$. Then we can see that $\delta((u, v), (a_2, b)) = 1$ would hold. Indeed, the function $g : C(v) \to L_0 - \{a_2, b-1, b, b+1\}$ such that $g(w^*) = a_3$, $g(w_2) = a_1$ $g(w') = f(w')$ for all other children w' of v, is injective and satisfies $\delta((v, w), (b, g(w))) = 1$ for all $w \in C(v)$. Furthermore, similarly to these observations, we can see that $\delta((u, v), (a, b)) = 1$ for all $a \in L_8 - \{a_2, b-1, b, b+1\}$. □

A.4 Computation of δ-value for V_3

First, we show the correctness of the procedure in the case of $v \in V_3$ in algorithm Compute-$\delta(v)$. Let $v \in V_3$, u be the parent of v, w^* be the unique child of v not in $V_M^{(2)}$, and b be a label such that $b \in L_0$ if $v \in V_3^{(1)} := \{v \in V_3 \mid C(v) \cap V_L = \emptyset\}$, and $b \in \{0, \Delta+1\}$ if $v \in V_3^{(2)} := V_3 - V_3^{(1)}$. Notice that if $v \in V_3^{(2)}$ (i.e., $C(v) \cap V_L \neq \emptyset$), then by Property 3, v is major and hence $\delta((u, v), (a, b)) = 1$, $a \in L_0$ indicates that $b = 0$ or $b = \Delta + 1$; namely we have only to check the case of $b \in \{0, \Delta+1\}$. Then, if there is a label $c \in L_0 - \{b-1, b, b+1\}$ such that $\delta((v, w^*), (b, c)) = 1$, then for all $a \in L_0 - \{b-1, b, b+1, c\}$, we have $\delta((u, v), (a, b)) = 1$. It is not difficult to see that this observation shows the correctness of the procedure in this case.

Next, we analyze the time complexity of the procedure. Obviously, for each $v \in V_3$, we can check which case of 3-1, 3-2, and 3-3 in algorithm Compute-$\delta(v)$ holds, and determine the values of $\delta((u, v), (*, b))$, in $O(1)$ time. Therefore, the values of $\delta((u, v), (*, *))$ can be determined in $O(\Delta)$ time. Below, in order to show that $\sum_{v \in V_3} t(v) = O(n)$, we prove that $|V_3| = O(n/\Delta)$.

As observed above, each vertex in $v \in V_3^{(2)}$ is major and we have $d(v) = \Delta$. Thus, it holds that $|V_3^{(2)}| = O(n/\Delta)$.

Finally, we show that $|V_3^{(1)}| = O(n/\Delta)$ also holds. By definition, we can observe that for any $v \in V_3^{(1)}$, $d'(v) = \tilde{d}(v) = 1$ (i.e., $d(v) = 2$). By Property 4, the size of any path component of T is at most 3.

This means that at least $|V_3^{(1)}|/3$ vertices in $V_3^{(1)}$ are children of vertices in $V_2 \cup V_3^{(2)} \cup V_4 \cup V_5$. Thus, $|V_3^{(1)}|/3 \leq \sum_{v \in V_2 \cup V_3^{(2)} \cup V_4 \cup V_5} \tilde{d}(v)$. From the discussions in the previous subsections (and this subsection), $\sum_{v \in V_2} \tilde{d}(v) = \mathrm{O}(n/\Delta^4)$, $\sum_{v \in V_3^{(2)}} \tilde{d}(v) = \sum_{v \in V_3^{(2)}} 1 = |V_3^{(2)}| = \mathrm{O}(n/\Delta)$, $\sum_{v \in V_4} \tilde{d}(v) = \sum_{v \in V_4} 1 = |V_4| = \mathrm{O}(n/\Delta)$, and $\sum_{v \in V_5} \tilde{d}(v) = \sum_{v \in V_5} 1 = |V_5| = \mathrm{O}(n/\Delta^2)$. Therefore, $|V_3^{(1)}| = \mathrm{O}(n/\Delta)$.