RA-002

木の $L(2,1)$-ラベリングのための $O(n \log^2 n)$ 時間アルゴリズム

An $O(n \log^2 n)$ Algorithm for $L(2,1)$-labeling of Trees

蓮沼 徹 † 石井 利昌 ‡ 小野 廣隆 § 宇野 裕之 ¶
Toru Hasunuma Toshimasa Ishii Hirotaka Ono Yushi Uno

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 is considered 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. On the other hand, this problem is also attractive from the graph theoretical point of view since it is a kind of vertex coloring problem. In this context, $L(2,1)$-labeling is generalized into $L(h,k)$-labeling for arbitrary nonnegative integers h and k, and in fact, we can see that $L(1,0)$ (or $L(h,0)$)-labeling is equivalent to the classical vertex coloring.

Related Work: There are also a number of studies about the $L(2,1)$-labeling problem from the algorithmic point of view. It is known to be NP-hard for general graphs [4], and it still remains NP-hard for some restricted classes of graphs, such as planar graphs, bipartite graphs and chordal graphs, and recently it turned out to be NP-hard even for graphs of treewidth 2 [2]. In contrast, only a few graph classes are known to have polynomial time algorithms for this problem. Among those, Chang and Kuo [1] established a polynomial time algorithm for the $L(2,1)$-labeling problem for trees. Their algorithm fully exploits the fact that $\lambda(T)$ is either $\Delta+1$ or $\Delta+2$ for any tree T. It is based on dynamic programming and runs in $O(\Delta^{4.5}n)$ time, where Δ is the maximum degree of a tree T and $n = |V(T)|$. Very recently, the authors of this paper have improved the result by presenting an $O(\min\{\Delta^{1.5}n, n^{1.75}\})$-time algorithm [5].

Our Contributions: In this paper, we present an $O(\min\{\Delta^{1.5}n, n\log^2 n\})$-time algorithm for $L(2,1)$-labeling of trees, which improves Chang and Kuo's $O(\Delta^{4.5}n)$ algorithm [1] and our previous $O(n^{1.75})$ algorithm, both of which are based on dynamic programming. We achieve the improvement by focusing on the fact that, in a labeling, some labels can often be replaced with other labels without violating the $L(2,1)$-labeling constraint. While both of the previous algorithms use the bipartite matching for filling DP tables up, our new algorithm uses the maximum flow for smaller graphs by utilizing the above fact.

2 Preliminaries

2.1 Definitions and Notations

In this paper, we assume that the reader has a basic knowledge of graph theory. 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. 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.

In describing algorithms, it is convenient to regard the input tree to be rooted at an arbitrary vertex r of degree 1. Then we can define the parent-child relationship on vertices in the usual way. For any vertex v, the set of its children is denoted by $C(v)$. For v, also define $d'(v) = |C(v)|$.

2.2 Chang and Kuo's Algorithm

We first review a significant result on $L(2,1)$-labeling of trees. We can see that $\lambda(G) \geq \Delta + 1$ holds for any graph G. Griggs and Yeh [4] observed that any major vertex in G must be labeled 0 or $\Delta+1$ when $\lambda(G) = \Delta+1$. Furthermore, they show the following lemma for trees.

Lemma 1 [4] *For any tree T, $\lambda(T)$ is either $\Delta+1$ or $\Delta+2$.*

Concerning this lemma, Griggs and Yeh [4] conjectured the problem of determining if $\lambda(T)$ is $\Delta + 1$ or $\Delta + 2$ is NP-hard. Chang and Kuo [1] disproved this by presenting a polynomial time algorithm for computing $\lambda(T)$, whose running time is $O(\Delta^{4.5}n)$. In this subsection, we review a basic idea of Chang and Kuo's algorithm.

Before explaining the idea of the algorithm, 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 decided for $T(v)$ in a sense.

* † 徳島大学, The University of Tokushima
‡ 小樽商科大学, Otaru University of Commerce
§ 九州大学, Kyushu Univerisity
¶ 大阪府立大学, Osaka Prefecture University

For a rooted tree, we call the length of the longest path from the root to a leaf its *height*. For $T(u,v)$, we define

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

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 assumption that $f(u)=a$ and $f(v)=b$, that is, 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:

$$\delta((u,v),(a,b)) = \begin{cases} 1 & \text{if there is a distinct assignment } c_1, \\ & \ldots, c_{d'(v)} \text{ on } w_1,\ldots,w_{d'(v)}, \text{ where} \\ & c_i \text{ is different from } a, b, b-1, b+1, \\ & \text{and } \delta((v,w_i),(b,c_i)) = 1 \text{ for each } i, \\ 0 & \text{otherwise,} \end{cases}$$

where $w_1,\ldots,w_{d'(v)}$ are the children of v. The existence of an assignment $c_1,\ldots,c_{d'(v)}$ on $w_1,\ldots,w_{d'(v)}$ as above is formalized as the maximum bipartite matching problem: For a bipartite graph $G(u,v,a,b) = (V(v), X, E(u,v,a,b))$, where $V(v) = \{w_1, w_2, \ldots, w_{d'(v)} \in C(v)\}$, $X = \{0, 1, \ldots, \Delta, \Delta+1\}$ and $E(u,v,a,b) = \{(w,c) \mid \delta((v,w),(b,c)) = 1, c \in X - \{a\}, w \in V(v)\}$, we can see that an assignment $c_1, c_2, \ldots, c_{d'(v)}$ on $w_1, w_2, \ldots, w_{d'(v)}$ is feasible if there exists a matching with size $d'(v)$ of $G(u,v,a,b)$. 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. Chang and Kuo's 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 combination of labels a and b. Thus the total running time is $O(\Delta^{2.5}) \times O(\Delta^2) \times O(n) = O(\Delta^{4.5}n)$, where the first $O(\Delta^{2.5})$ is the time complexity of the bipartite matching problem. The algorithm in [5] also utilizes the same scheme, but it achieves a better running time $O(\min\{\Delta^{1.5}n, n^{1.75}\})$ by introducing another algorithm for large Δ and an efficient computation of δ values with amortized analyses.

3 Label Compatibility Preserving $\lambda(T) = \Delta+1$

In this section, we introduce the notion of label compatibility, which enables us to treat several labels equivalently under the computation of δ values in Subsection 2.2.

Definition 1 We say the *neck level* (resp., *head level*) of $T(u,v)$ is h if for any label a the following holds:

Case $h = 0$: for any $b \in \{0,1,\ldots,\Delta,\Delta+1\} - \{a, a-1, a+1\}$ (resp., for any $a \in \{0,1,\ldots,\Delta,\Delta+1\} - \{b, b-1, b+1\}$), $\delta((u,v),(a,b)) = 1$,
Case $h > 0$: for any $b \in \{h, h+1, \ldots, \Delta-h, \Delta+1-h\} - \{a, a-1, a+1\}$ (resp., for any $a \in \{h, h+1, \ldots, \Delta-h, \Delta+1-h\} - \{b, b-1, b+1\}$), $\delta((u,v),(a,b)) \neq \delta((u,v),(a,h-1))$.

An intuitive explanation of neck (resp., head) level h of $T(u,v)$ is that if v (resp., u) is assigned to a label in $\{h,\ldots,\Delta+1-h\}$ under $(\Delta+1)$-$L(2,1)$-labeling of $T(u,v)$, the label can be replaced with another label in $\{h,\ldots,\Delta+1-h\}$ without violating a proper $(\Delta+1)$-$L(2,1)$-labeling; labels in $\{h,\ldots,\Delta+1-h\}$ are compatible. For the neck/head levels, we can show the following theorem, though the detail is omitted.

Theorem 1 For $T(u,v)$, both the head and neck levels of $T(u,v)$ are at most $\log|T(u,v)|$. □

4 Flow-based Algorithm

We can compute δ values by a maximum flow algorithm instead of the bipartite matching, though usually it is meaningless because the time complexity of the maximum flow problem is larger than the one of the bipartite matching problem. In this case, however, a maximum flow algorithm can achieve a faster running time because Theorem 1 can reduce the sizes of target graphs. We consider a network $\mathcal{N}(u,v,a,b) = (\{s,t\} \cup V(v) \cup X_h, E(v) \cup E_X \cup E_\delta, cap)$ instead of $G(u,v,a,b)$, where $V(v)$ is defined in Section 2.2, $X_h = X - \{h+1,\ldots,\Delta-h\}$, $E(v) = \{(s,w) \mid w \in V(v)\}$, $E_X = \{(c,t) \mid c \in X_h\}$, $E_\delta = \{(w,c) \mid w \in V(v), c \in X_h\}$, and $cap(e)$ function is defined as follows: for $\forall e \in E(v)$, $cap(e) = 1$, for $e = (w,c) \in E_\delta$, $cap(e) = \delta((v,w),(b,c))$, and for $e = (c,t) \in E_X$, $cap(e) = 1$ if $c \neq h$, $cap(e) = |\{h,\ldots,\Delta+1-h\} - \{a,b,b+1,b-1\}|$ if $c = h$. It is not difficult to show that the maximum flow on $\mathcal{N}(u,v,a,b)$ computes δ value by the label compatibility. Since $\mathcal{N}(u,v,a,b)$ has $O(d'(v)+h)$ vertices, $O(d'(v)h)$ edges and at most Δ units of cap, the network flow is solved in $O(\Delta^{2/3}d'(v)h\log^2\Delta)$ [3]. Thus $\delta((u,v),(a,b))$ is computed in $O(\Delta^{2/3}d'(v)\log n\log^2\Delta)$ time by Theorem 1.

The efficient update of matching structures introduced in [5] can also be applied to the network structures, which implies that δ values for all pairs of (a,b) with $a,b \le h$ are computed in $O(\Delta^{2/3}d'(v)\log^2 n\log^2\Delta)$ time. Also, we can obtain $\sum_v d'(v) = O(n/\Delta)$ by a similar amortized analysis. Combining these, we achieve the running time $O(n\log^2 n)$.

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

参考文献

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

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

[3] A. V. Goldberg and S. Rao, Beyond the flow decomposition barrier, *J. ACM*, **45-5**, 783–797 (1998).

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

[5] T. Hasunuma, T. Ishii, H. Ono and Y. Uno, An $O(n^{1.75})$ Algorithm for $L(2,1)$-labeling of Trees, to appear in *Proc. SWAT 2008.*