Contributions to Mathematics
www.shahindp.com/locate/cm

Contrib. Math. **11** (2025) 14–21
DOI: 10.47443/cm.2025.013

Research Article

Local antimagic labelling for trees*

Subhabrata Paul†, Soumen Raul

Department of Mathematics, Indian Institute of Technology Patna, Bihta 801106, Bihar, India

(Received: 5 February 2025. Received in revised form: 29 March 2025. Accepted: 7 April 2025. Published online: 14 April 2025.)

Abstract

Let $G(V, E)$ be a graph, where V and E are the vertex set and edge set, respectively. A local antimagic labelling of G is a bijection $f : E \to \{1, 2, \ldots, |E|\}$ such that for every edge $uv \in E$, the condition $\sum_{e\in I(u)} f(e) \neq \sum_{e\in I(v)} f(e)$ holds, where $I(u)$ represents the set of edges incident to u. Bensmail, Senhaji, and Lyngsie [*Discrete Math. Theor. Comput. Sci.* **19** (2017) #21] proposed a recursive algorithm for local antimagic labelling of trees. In this article, we propose a linear-time iterative algorithm for computing a local antimagic labelling for trees. A major tool used in this algorithm is a specific partitioning scheme, introduced by Kaplan, Lev, and Roditty [*Discrete Math.* **309** (2009) 2010–2014]. We believe that our technique can be generalised to achieve local antimagic labelling for some superclasses of trees.

Keywords: local antimagic labelling; tree; linear-time algorithm.

2020 Mathematics Subject Classification: 05C78, 05C85.

1. Introduction

Since its introduction in 1960s, graph labelling has been a major area of research in graph theory due to its enormous applications in science and engineering. By graph labelling, we mean an assignment of labels (generally integers) to the vertices, edges, or both of a graph, satisfying few predetermined conditions that depend on the type of labelling. Over the years, many different variations have been introduced and studied from both algorithmic and structural point of view. Some important variations of graph labelling are graceful labelling, harmonious labelling, magic labelling, prime labelling, radio labelling, geometric labelling, etc. We refer readers to the survey [7] for details.

Hartsfield and Ringel [8] introduced a novel concept of graph labelling known as *antimagic labelling*. This labelling scheme involves assigning distinct integers to the edges of a graph so that the sums of the labels on edges incident to each vertex are unique. The sums of the labels of the edges incident to the vertex v are referred to as the *vertex sum* and denoted by $sum(v)$. Specifically, for a graph $G(V, E)$ with $|E| = m$, antimagic labelling is a bijection $f : E(G) \to \{1, 2, \ldots, m\}$ such that $sum(u) \neq sum(v)$ for every pair of vertices u and v in G. Clearly, if a graph contains a component isomorphic to K_2, then it does not admit any antimagic labelling. Therefore, we study antimagic labelling for graphs that do not have any component isomorphic to K_2. We call such a graph *nice graph*. In [8], the authors conjectured that "every nice graph is antimagic". Although antimagic labelling has been studied for many graph classes, including Cayley graphs, regular graphs, Cartesian product of graphs, split graphs, generalised Petersen graphs, this conjecture remains unresolved even for trees. Kaplan et al. [11] proved that trees with at most one vertex of degree 2 are antimagic. However, their proof contained an error, which was later corrected by Liang et al. [13]. Chawathe and Krishna [5] proved that all complete m-ary trees are antimagic. Alon et al. [1] used probabilistic methods and techniques from analytic number theory to support that the conjecture holds for graphs with n vertices and a minimum degree of $\Omega(\log n)$. Hafetz et al. [10] proved that if G is a graph with $n = 3^k$ vertices, where $k \in \mathbb{N}$, and admits a K_3-factor, then it is antimagic. Cranston et al. [6] proved that all regular bipartite graphs with a minimum degree of 2 are antimagic.

Arumugam et al. [2] and Bensmail et al. [4] independently introduced a variation of antimagic labelling, known as *local antimagic labelling*. A labelling that assigns each edge a unique label from the set $\{1, 2, \ldots, m\}$ is called a *local antimagic labelling* if, for every edge uv, the $sum(u)$ and $sum(v)$ are distinct. It is clear that every antimagic labelling is also local antimagic labelling. Furthermore, local antimagic labelling creates a proper vertex coloring of G, where each vertex v gets a color based on its vertex sum. The minimum number of distinct vertex sums induced by local antimagic labellings of a graph G is known as the chromatic number of local antimagic and is denoted by $\chi_{\text{la}}(G)$.

*Both authors contributed equally to this work.
†Corresponding author (subhabrata@iitp.ac.in).

In [2], Arumugam et al. conjectured that every nice connected graph is local antimagic. Bensmail et al. [4] proposed a slightly stronger form of this conjecture, which states that every nice graph has local antimagic labelling. In [9], Haslegrave proved this conjecture concerning local antimagic labelling of connected graphs using the probabilistic method. In [12] and [16], the exact value of the local antimagic chromatic number was determined for various types of join graphs. Nazula et al. [14] determined the local antimagic chromatic number for some unicyclic graphs, which are connected graphs with exactly one cycle. Bača et al. [3] showed that the local antimagic chromatic number for all complete full t-ary trees is exactly $l+1$ when t is odd, where l represents the number of leaves. Sarath et al. [15] calculated the local antimagic chromatic number for trees with a diameter of 3, certain classes of trees with a diameter of 4, and complete bipartite graphs $K_{m,n}$ when m and n have different parities. Readers are referred to [7] for more details.

In this paper, we present a linear-time algorithm for local antimagic labelling of trees in an iterative approach. Although Bensmail et al. [4] proposed an algorithm for local antimagic labelling of trees, it is a recursive process, however. We believe that our work presents a straightforward alternative to the method proposed by Bensmail et al. [4]. We also believe that our technique can be generalised to achieve local antimagic labelling for some superclasses of trees.

2. Local antimagic labelling of trees

In this section, we design an iterative algorithm to obtain a local antimagic labelling of trees. It is known that for every nice graph G, with maximum degree 2, any arbitrary labelling from $\{1,2,\ldots,m\}$ gives a local antimagic labelling. Therefore, we assume that the input tree has a maximum degree of at least 3.

First, we give an overview of the algorithm. Given a tree T, we first fix a specific vertex as the root and consider the tree as a rooted tree. With a slight abuse of notation, we refer to both the tree and the rooted tree by T. We then assign labels to some edges of the rooted tree according to the labelling scheme used in [11]. Kaplan et al. showed that for every even integer k, we can partition the set $\{1,2,...,k\}$ into 2-subsets and 3-subsets following some properties that help in achieving antimagic labelling of a specific type of trees [11]. As we use this labelling in our algorithm, we restate the result in the following lemma.

Lemma 2.1 (see [11]). *Let k be an even integer which is of the form $k = 2s + 3\cdot 2l$, where s and l are two non-negative integers. Then the set $\{1,2,...,k\}$ can be partitioned into $s+2l$ subsets, say $A_1, A_2, ..., A_{s+2l}$, such that the following hold:*

1. *There are s subsets having two elements (2-subsets) and the sum of those elements is $k+1$.*
2. *There are l subsets having three elements (3-subsets) and the sum of those elements is $k+1$.*
3. *There are l subsets having three elements (3-subsets) and the sum of those elements is $2(k+1)$.*

First, we fix a specific vertex as root in such a way that allows us to use the labelling technique described in Lemma2.1. Our main goal here is to set the value of k as $k = m$, when m is even and $k = m-1$ otherwise. Then, in the rooted tree T, for every vertex having at least two descendant edges, we label those descendant edges using the labelling described in Lemma 2.1 as follows. If a vertex has odd number of descendant edges, three of those descendant edges are labelled using a 3-subset, and the remaining descendant edges are labelled using some 2-subsets. For a vertex having an even number of descendant edges, we label those descendant edges using some 2-subsets. After that, we label all the remaining edges arbitrarily using the remaining labels from $\{1,2,\ldots,m\}$. This arbitrary labelling might create some conflicting edges, that is, an edge uv where the vertex sums of u and v are the same. Finally, we describe a method to resolve these conflicts.

Now, we describe each step of the algorithm in detail. First, we introduce some notations that are helpful in describing the algorithm. Let D_{even} denote the vertices in the rooted tree T that have an even number of descendant edges, $D_{odd,3}$ denote the vertices in the rooted tree T that have degree at least 3 and also have an odd number of descendant edges, and finally, $D_{odd,1}$ denote the vertices in the rooted tree T that have exactly 1 descendant edge.

Step 1: Root identification

In the first step, we fix a specific vertex as the root. That specific vertex is chosen depending on the parity of m and the number of degree 2 vertices in the tree. Let n_2 denote the number of degree 2 vertices in T.

Algorithm 1 Root identification

if m is odd and $n_2 \neq 1$ **then**
    Choose a pendant vertex v (i.e., $deg(v) = 1$) as the root.
else if m is even and $n_2 = 1$ **then**
    Choose the two-degree vertex as the root.
else
    Select a vertex v as the root such that $deg(v) \geq 3$.

Step 2: Setting the values of k, l and s

Next, we set the values for k, s, and l, so that we can apply Lemma 2.1 for some edges. Since, in Lemma 2.1, k is even, we set k based on the parity of m. The value of l is set based on the parity of $|D_{odd,3}|$. If $|D_{odd,3}|$ is even, then we set $2l = |D_{odd,3}|$; otherwise, we set $2l = |D_{odd,3}| + 1$. Finally, we set s according to $k = 2s + 3 \cdot 2l$.

Algorithm 2 Setting the values of k, l and s

if m is odd **then**
    $k = m - 1$.
else
    $k = m$.
if $|D_{odd,3}|$ is even **then**
    $2l = |D_{odd,3}|$.
else
    $2l = |D_{odd,3}| + 1$.
$s = (k - 3 \cdot 2l)/2$

To prove the correctness of Algorithm 2, we have the following lemma (its proof is given in the next section):

Lemma 2.2. *The value of s is always non-negative.*

Step 3: Preliminary labelling of the tree

Now, we apply Lemma 2.1 to assign labels to some of the edges of the tree. The rules for assigning labels are given in the following algorithm.

Algorithm 3 Preliminary labelling of the tree

for each vertex v of T **do**
    if v is the root and it is a pendant vertex **then**
        Label the edge incident to v by $k + 1$.
    if $v \in D_{odd,3}$ **then**
        Label 3 of those descendant edges by one 3-subset and label the remaining edges using some 2-subsets.
    if $v \in D_{even}$ **then**
        Label all the descendant edges using some 2-subsets.
for each unlabeled edge e **do**
    Label e by assigning any unused number from $\{1, 2, ..., m\}$.

Step 4: Identify conflicting edges

The labelling described in Step 3 may not be a local antimagic labelling. An edge e is called a *conflicting edge* if the vertex sums of the end points are the same. There are two types of conflicting edges, namely, conflicting edges of type-I and type-II. Now, we describe these two types of conflicting edges. Let $e = uv$ be a conflicting edge with u being the parent of v in the rooted tree T.

Conflicting edge of type-I: The edge $e = uv$ is called a *conflicting edge of type-I* if degree of u is 2, the parent of u is the root vertex having degree 1 and sum of descendant edges of v is $k + 1$.

Conflicting edge of type-II: Let e_1 and e_2 be the edges that are above u and below v, respectively. Also, let the labels of e_1, e and e_2 are x, y and z, respectively. The edge $e = uv$ is called a *conflicting edge of type-II* if the degree of v is 2, the sum of labels of the descendant edges of u is $k + 1$, and $y + z - x = k + 1$. We consider the edge $e = uv$ as a conflicting edge of type-II even if u is the root vertex; in that case, e_1 does not exist, and hence, x is considered to be 0.

In the next section, we show that except for these two types of conflicts, no other type of conflict can occur by proving the following lemma:

Lemma 2.3. *After Step 3, only conflicting edges of type-I and type-II can occur.*

Step 5: Resolve conflicts

Once we have identified the conflicting edges, we describe how to resolve them. We describe two resolution techniques for the two types of conflicting edges.

Resolution for a conflicting edge of type-I: We interchange the labels of the edge uv and its above edge.

Resolution for a conflicting edge of type-II: We interchange the labels of the edge uv and one descendant edge of u.

Note that there is at most one edge in T that can be a conflicting edge of type-I. If such an edge is present, then we apply the corresponding resolution technique once. We prove (in the next section) the following lemma that shows that after applying resolution for a conflicting edge of type-I, similar conflict cannot occur.

Lemma 2.4. *The resolution for a conflicting edge of type-I does not create any other conflicting edge of type-I.*

The resolution for a conflicting edge of type-II does not work in the same way; it might create a new conflicting edge of type-II. But, the following lemma (to be proved in the next section) confirms that the newly created conflicting edge of type-II must have a higher level than the current conflicting edge of type-II.

Lemma 2.5. *Let e be a conflicting edge of type-II and the corresponding resolution be applied on e. This resolution for e can create another conflicting edge of type-II, say e'. In that case, level of e' is 1 more than the level of e.*

Because of Lemma 2.5, we need to apply the resolution for conflicting edges of type-II in a specific order. We order the edges of T from top to bottom, keeping all the edges in the same level together. We refer such an ordering of edges as *level-wise top-down ordering*.

Algorithm 4 An algorithm for identifying conflict and resolve

Input: Rooted tree $T = (V, E)$ and the preliminary labelling of T after Step 3.
Output: A local antimagic of T.

1: **if** there is a conflicting edge of type-I **then**
2: Apply Resolution for conflicting edge of type-I.
3: Let σ be the level-wise top-down ordering of the edges in T.
4: **for** each edge $e = uv$ in σ **do**
5: **if** e is a conflicting edge of type-II **then**
6: Apply Resolution for conflicting edge of type-II.

3. Proof of correctness and running time

In this section, we discuss the running time of the proposed algorithm and show the correctness of this algorithm. First, we provide proofs of the lemmas stated in the previous section.

Proof of Lemma 2.2. To prove that s is always non-negative, we show $k \geq 3 \cdot 2l$. Note that the total number of descendant edges of the vertices in $D_{odd,3}$ is at least $3 \cdot |D_{odd,3}|$. If $|D_{odd,3}|$ is even, then we have set $2l = |D_{odd,3}|$. Hence, $m \geq 3 \cdot 2l$. Now, when m is even, we have set $k = m$ and hence $k \geq 3 \cdot 2l$. On the other hand, if m is odd, then $(m-1) \geq 3 \cdot 2l$ as $3 \cdot 2l$ is an even number. As we set $k = m - 1$, we have $k \geq 3 \cdot 2l$.

If $|D_{odd,3}|$ is odd, then we set $2l = |D_{odd,3}| + 1$. This implies that the total number of edges descending from the vertices of $D_{odd,3}$ is at least $3 \cdot (2l - 1)$. To complete the proof, it is sufficient to show that the number of edges descending from the vertices of $D_{odd,3}$ is at most $k - 3$. Since $|D_{odd,3}|$ is odd, the total number of edges descending from the vertices of $D_{odd,3}$ is odd. Therefore, the total number of edges descending from the vertices of $D_{odd,3} \cup D_{even}$ is also odd. This implies that if m is even, then $|D_{odd,1}|$ is odd; otherwise $|D_{odd,1}|$ is even. The rest of the proof is divided into three cases based on the root vertex.

Case 1. The root is a pendant vertex.

Note that m is odd and $n_2 \neq 1$. Since m is odd, we have $k = m - 1$, and $|D_{odd,1}|$ is even. As the root vertex belongs to $D_{odd,1}$, which implies that $|D_{odd,1}|$ is at least 2. If $|D_{odd,1}| = 2$, it would follow that $n_2 = 1$, which contradicts the condition $n_2 \neq 1$. Therefore, $|D_{odd,1}|$ is at least 4. The number of edges descending from vertices of $D_{odd,3} \cup D_{odd,1}$ is at most m. So, the number of edges descending from vertices of $D_{odd,3}$ is at most $m - 4$. Since $k = m - 1$, the number of edges descending from vertices of $D_{odd,3}$ is at most $k - 3$. Hence, $k \geq 3 \cdot 2l$.

Case 2. The root is a vertex of degree 2.
Note that m is even and $n_2 = 1$. Since m is even, $|D_{odd,1}|$ must be odd. However, as the root has two descendant edges, it follows that the root vertex belongs to D_{even}, which implies $|D_{odd,1}|$ is 0, leading to a contradiction. Hence, if the root is a vertex of degree 2, $|D_{odd,3}|$ being odd is not possible.

Case 3. The degree of the root vertex is at least 3.
Note that all vertices of $D_{odd,1}$ must be vertices of degree 2. So, $n_2 = |D_{odd,1}|$. In this case, there are two possibilities.

Case 3.1. m is even and $n_2 \neq 1$.
Since the total number of edges of T is even, it follows that $k = m$ and $|D_{odd,1}|$ is odd. As $n_2 \neq 1$ and $|D_{odd,1}|$ is odd, it follows that $|D_{odd,1}|$ is at least 3. Since the total number of edges descending from vertices of $D_{odd,3} \cup D_{odd,1}$ is at most m, the number of edges descending from vertices of $D_{odd,3}$ is at most $m - 3$. As $k = m$, this implies that the number of edges descending from vertices of $D_{odd,3}$ is at most $k - 3$. Hence, $k \geq 3 \cdot 2l$.

Case 3.2. m is odd and $n_2 = 1$.
Since the total number of edges in T is odd, it follows that $|D_{odd,1}|$ must be even. However, in this case, $n_2 = |D_{odd,1}| = 1$, leading to a contradiction. Hence, if m is odd and $n_2 = 1$, $|D_{odd,3}|$ cannot be odd.

Therefore, the value of s is always non-negative. □

Proof of Lemma 2.3. We prove this result by examining every edge $e = uv$ of the tree T. Observe that if the degree of either u or v is 1, then $e = uv$ cannot be a conflicting edge. Hence, we assume that $deg(u) \geq 2$ and $deg(v) \geq 2$. We divide the proof based on the degrees of u and v. Let us consider that u' is the parent of u if u is not the root vertex. Also, recall that $sum(u)$ denotes the vertex sum of u.

Case 1. $deg(u) > 2$ and $deg(v) > 2$.
Since the degrees of u and v are at least 3, the number of descendant edges of u and v is at least 2. We label these descendant edges using either 2-subsets or 3-subsets, or both. Therefore, the sum of the labels of the descendant edges of u and v is a multiple of $(k + 1)$. Let the sum of the labels of the descendant edges from u and v be $p(k + 1)$ and $q(k + 1)$, respectively, where $p, q \in \mathbb{N}$ and the label of the edge uv is y, where $1 \leq y \leq k$. So, the vertex sum of v is $q(k + 1) + y$. The vertex sum of u depends on the edge label above u. Therefore, we analyze this case based on the edge label above u.

Case 1.1. The label of $u'u$ lies between 1 and k.
Let the label of edge $u'u$ be x. The vertex sum of u is $p(k+1)+x$. Thus, $sum(u) \equiv x \pmod{k+1}$ and $sum(v) \equiv y \pmod{k+1}$. Since labels are not repeated, it follows that $sum(u) \neq sum(v)$. Therefore, $e = uv$ is not a conflicting edge.

Case 1.2. The label of $u'u$ is $k + 1$, or u is the parent of the tree.
If the label of $u'u$ is $(k + 1)$, then the vertex sum of u becomes $(p + 1) \cdot (k + 1)$. If u is the root of the tree, the vertex sum of u is $p(k + 1)$. In both cases, $sum(u) \equiv 0 \pmod{k + 1}$. On the other hand, the vertex sum of v is $q(k + 1) + y$, which implies $sum(v) \not\equiv 0 \pmod{k + 1}$. So, $sum(u) \neq sum(v)$. Therefore, $e = uv$ is not a conflicting edge.

Case 2. $deg(u) > 2$ and $deg(v) = 2$.
If v is the only two-degree vertex, then the descendant edge label of v is $k + 1$. By reasoning similar to the previous case, we can conclude that $sum(u) \neq sum(v)$.
As the number of descendant edges of u is at least 2, the sum of the labels of these edges is a multiple of $(k + 1)$. Let the labels of edge uv and the descendant edge of v be y and z, respectively, where $1 \leq y, z \leq k$. If $deg(u) \geq 5$, the number of descendant edges from u is at least 4, so the sum of the labels of u's descendant edges is at least $2(k + 1)$. Since the sum of y and z is at most $2k - 1$, the vertex sums of u and v will never conflict. Now, let us consider the case where $deg(u) \leq 4$. Next, we examine this case based on the edge label above u.

Case 2.1. The label of $u'u$ lies between 1 and k.
Let the label of the edge $u'u$ be x. If $\deg(u) = 4$, then u has 3 descendant edges, which are labeled using a 3-subset. If the sum of the labels of these edges is $2(k + 1)$, then $sum(u) \neq sum(v)$. On the other hand, if the sum of the labels of these edges is $k + 1$ (meaning the edges are labeled with a 3-subset whose sum is $k + 1$) or if $\deg(u) = 3$ (which implies that u has two descendant edges and labeled these edges using a 2-subset), then $sum(u) = (k + 1) + x$ and $sum(v) = y + z$. If $y + z - x = k + 1$, a conflict arises between u and v. Therefore, the edge $e = uv$ is a conflicting edge of type-II.

Case 2.2. Label of $u'u$ is $k+1$.
If the label of the edge $u'u$ is $k+1$, then the vertex sum of u is always at least $2(k+1)$, as the sum of the descendant edges of u is at least $k+1$, while the vertex sum of v is at most $2k-1$. So, $sum(u) \neq sum(v)$. Therefore, $e = uv$ is not a conflicting edge.

Case 2.3. u is the root.
If $deg(u) = 4$, this implies that the number of descendant edges of u is 4. To label these edges, we use two 2-subsets. As a result, the vertex sum of u is $2(k+1)$, ensuring that $sum(u) \neq sum(v)$. If $deg(u) = 3$ and the descendant edges of u are labeled with a 3-subset whose sum is $2(k+1)$, the vertex sums of u and v will not conflict. However, if these edges are labeled with a 3-subset whose sum is $k+1$ and $y+z = k+1$, then a conflict arises between u and v. Therefore, $e = uv$ is a conflicting edge of type-II.

Case 3. $deg(u) = 2$ and $deg(v) > 2$.
If u is not the only vertex of degree two, then we do not choose a vertex of degree two as the root. So, the conflict between u and v depends on the edge label above u and the sum of the labels of the descendant edges of v, as the label of the edge uv contributes equally to the vertex sums of both u and v. Therefore, we analyze this case based on the edge label above u.

Case 3.1. The label of $u'u$ lies between 1 and k.
The degree of v is at least three, which means the number of descendant edges of v is at least two. The sum of the labels of these descendant edges is at least $k+1$, as edges of a vertex with at least two descendants are labeled using either 2-subset or 3-subset, or both. Since the label of the edge $u'u$ lies between 1 and k and the sum of the labels of the descendant edges of v is at least $k+1$, it follows that the vertex sums of u and v are always distinct. Therefore, $e = uv$ is not a conflicting edge.

Case 3.2. The label of $u'u$ is $k+1$, that is, u' is the pendant root of the tree.
If $deg(v) \geq 5$, then the sum of the labels of the descendant edges of v is at least $2(k+1)$, ensuring that $sum(u) \neq sum(v)$. Suppose that $deg(v) = 4$, the sum of the labels of the descendant edges of v is either $2(k+1)$ or $k+1$, depending on the type of 3-subset used. When the descendant edges of v are labeled with a 3-subset whose sum is $2(k+1)$, the vertex sums of u and v do not conflict. However, if the descendant edges of v are labeled with a 3-subset whose sum is $k+1$, or if $deg(v) = 3$, implying v has two descendant edges labeled using a 2-subset, a conflict arises between u and v. Therefore, the edge $e = uv$ is a conflicting edge of type-I.

Case 3.3. The vertex u is the root.
When m is even and $n_2 = 1$, then only we root two-degree vertex. In this case, the vertex sum of u is $k+1$, that is,

$$sum(u) \equiv 0 \pmod{k+1}.$$

The sum of the labels of the descendant edges of v is a multiple of $k+1$, and the edge label of uv lies between 1 and k. This implies that

$$sum(v) \not\equiv 0 \pmod{k+1}.$$

Therefore, $sum(u) \neq sum(v)$, and $e = uv$ is not a conflicting edge.

Case 4. $deg(u) = 2$ and $deg(v) = 2$.
The labels of the edges that are above u and below v are distinct. Hence, the vertex sums of u and v do not conflict. Therefore, $e = uv$ is not a conflicting edge.

Therefore, after Step-3, we can have only two types of conflicting edges. □

Proof of Lemma 2.4. Suppose that $e = uv$ is a conflicting edge of type-I. Then, the degree of u is 2, the parent of u is the root vertex with degree 1, and the sum of the labels of the descendant edges of v is $k+1$. Since the root is a pendant vertex, the edge incident to the root is labelled as $k+1$, which makes the vertex sums of u and v equal. In the resolution for conflicting edge of type-I, the labels of uv and the edge above it are interchanged. After interchanging, the label of the incident edge of the root lies between 1 and k, and the label of the edge uv becomes $k+1$. As the label of the edge uv contributes equally to the vertex sums of both u and v, the label of the edge above uv lies between 1 and k, but the sum of the descendant edges of v remains $k+1$, which implies $sum(u) \neq sum(v)$. Therefore, the conflict at the edge uv is resolved. Moreover, since none of the labels of any edge below v is changed, this resolution for a conflicting edge of type-I does not create any other conflicting edge of type-I. □

Proof of Lemma 2.5. Suppose that $e = uv$ is a conflicting edge of type-II. Also assume that e_1 is the edge above u and e_2 is the edge below v of the tree T and the labels assigned to e_1, e, and e_2 in Step 3 are x, y, and z, respectively. Since the edge $e = uv$ is a conflicting edge of type-II if the degree of v is 2, the sum of the labels of the descendant edges of u is $k + 1$, and the relationship $y + z - x = k + 1$ holds. If u is the root vertex, then e_1 does not exist. In such a case, x is considered to be 0, and $e = uv$ is still considered to be a conflicting edge of type-II. To resolve this type of conflict, the labels of the edge uv and another descendant edge of u, say uv', are interchanged. After this interchange, the vertex sums of u and v become distinct. Since the vertex sum of u remains the same, resolving the conflict of e cannot create any conflicting edge of type-II that is above the edge $e = uv$.

Now, we show that resolving the conflict of e cannot create another conflict at the level of e. To prove this, we show that $sum(u) \neq sum(v')$. Consider the case when $deg(v') = 2$. In this scenario, $sum(u) \neq sum(v')$ because the vertex sum of u remains unchanged and the label of the descendant edges of v and v' are distinct. Now, if $deg(v') > 2$, we still have $sum(u) \neq sum(v')$. This is because the sum of the labels of the descendant edges of u and v' is a multiple of $k + 1$, and the labels of the above edge of u and v' are distinct and within the range 1 to k. However, a conflicting edge of type-II may occur at the edge $v'v''$, where v'' is the child of v' and the degree of v'' is 2. If the sum of the labels of the descendant edges of v' is $k + 1$, and $a + b - y = k + 1$, where the labels of the edges $v'v''$ and the edge below v'' are a and b respectively, and $a, b \in [1, k]$, then a conflicting edge of type-II arises at $v'v''$. Therefore, this resolution for e, can create another conflicting edge of type-II, say $e' = v'v''$. In that case, the level of e' is 1 more than the level of e. □

Next, we discuss the running time of our algorithm. Firstly, in Step 1, identifying the root needs information about the number of degree 2 vertices of the graph. This can be calculated in $O(n)$ time. Therefore, we can fix the root of the tree in linear time. Once the root is fixed, we can calculate $|D_{odd,3}|$ in linear time, and hence Step 2 also takes linear time. The preliminary labelling assignment can also be done in linear time; we have to traverse the vertices in BFS/DFS order and assign 2-subsets and/or 3-subsets as required. Finding conflicting edges and their resolution can be done in linear time. If a conflicting edge is of type-I, then applying the corresponding resolution only once is sufficient because of Lemma 2.4. However, if the conflicting edge is of type-II, then we have to apply the corresponding resolution on edges in a specific order. Since each resolution is just an interchange of labels, it takes constant time, and hence, the whole resolution process in Step 5 takes linear time. In view of this discussion, we have the following main result:

Theorem 3.1. *Given a tree T, a local antimagic labelling can be obtained in a iterative way in linear time.*

Remark 3.1. *We can use the proposed algorithm to construct a local antimagic labelling of a forest. First, we pick a pendant vertex from every component of the forest and glue these vertices together into a single vertex to construct a tree. Then, we apply the algorithm for trees to obtain a local antimagic labelling of the tree. Clearly, this labelling is also a local antimagic labelling of the forest.*

4. Conclusion

In this paper, we have given a linear-time algorithm to achieve a local antimagic labelling for trees (forests). We believe that our technique can be generalised to achieve a local antimagic labelling for some superclasses of trees. Designing an algorithm for local antimagic labelling of other well-known classes of graphs, including block graphs and unicyclic graphs, seems to be an interesting problem.

Acknowledgement

The work of Soumen Raul is supported by the University Grants Commission (NTA Ref. No.: 211610022458), Government of India.

References

[1] N. Alon, G. Kaplan, A. Lev, Y. Roditty, R. Yuster, Dense graphs are antimagic, *J. Graph Theory.* **47** (2004) 297–309.
[2] S. Arumugam, K. Premalatha, M. Bača, A. Semaničová-Feňovčíková, Local antimagic vertex coloring of a graph, *Graphs Combin.* **33** (2017) 275–285.
[3] M. Bača, A. Semaničová-Feňovčíková, R. T. Lai, T. Wang, On local antimagic vertex coloring for complete full t-ary trees, *Fundam. Inform.* **26** (2022) 99–113.
[4] J. Bensmail, M. Senhaji, K. S. Lyngsie, On a combination of the 1-2-3 conjecture and the antimagic labelling conjecture, *Discrete Math. Theor. Comput. Sci.* **19** (2017) #21.
[5] P. D. Chawathe, V. Krishna, Antimagic labelings of complete m-ary trees, In: A. K. Agarwal, B. C. Berndt, C. F. Krattenthaler, G. L. Mullen, K. Ramachandra, M. Waldschmidt (Eds.), *Number Theory and Discrete Mathematics*, Birkhäuser, 2002, 77–80.
[6] D. W. Cranston, Regular bipartite graphs are antimagic, *J. Graph Theory.* **60** (2009) 173–182.
[7] J. A. Gallian, A dynamic survey of graph labeling, *Electron. J. Combin.* **6** (2022) 4–623.
[8] N. Hartsfield, G. Ringel, *Pearls in Graph Theory*, Academic Press, 1994.

[9] J. Haslegrave, Proof of a local antimagic conjecture, *Discrete Math. Theor. Comput. Sci.* **20** (2018) #18.

[10] D. Hefetz, Anti-magic graphs via the combinatorial nullstellensatz, *J. Graph Theory.* **50** (2005) 263–272.

[11] G. Kaplan, A. Lev, Y. Roditty, On zero-sum partitions and anti-magic trees, *Discrete Math.* **309** (2009) 2010–2014.

[12] G. C. Lau, K. Premalatha, S. Arumugam, W. C. Shiu, On local antimagic chromatic number of cycle-related join graphs II, *Discrete Math. Algorithms Appl.* **16** (2024) #2350022.

[13] Y. Liang, T. Wong, X. Zhu, Anti-magic labeling of trees, *Discrete Math.* **331** (2014) 9–14.

[14] N. H. Nazula, S. Slamin, D. Dafik, Local antimagic vertex coloring of unicyclic graphs, *Indones. J. Combin.* **2** (2018) 30–34.

[15] V. S. Sarath, A. V. Prajeesh, Local antimagic chromatic number of certain classes of trees, *Second International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), Trichirappalli, India*, 2023, 1–6.

[16] X. Yang, H. Bian, H. Yu, D. Liu, The local antimagic chromatic numbers of some join graphs, *Math. Comput. Appl.* **26** (2021) #80.