U.P.B. Sci. Bull., Series A, Vol. 80, Iss. 3, 2018 ISSN 1223-7027

COMPUTING EDGE IRREGULARITY STRENGTH OF COMPLETE M-ARY TREES USING ALGORITHMIC APPROACH

A. AHMAD[1], M. A. ASIM[1,3], M. BAČA[2], R. HASNI[3]

Algorithms help in solving many problems, where other mathematical solutions are very complex or impossible. Computations help in tackling numerous issues, where other numerical arrangements are extremely perplexing or incomprehensible. In this paper, the edge irregularity strength of a complete binary tree ($T_{2,h}$), complete ternary tree ($T_{3,h}$) and generalized for complete m-ary tree are computed using the algorithmic approach.

Keywords: edge irregularity strength, complete m-ary tree $T_{m,h}$, algorithms.

1. Introduction

A graph $G(V,E)$ with vertex set V and edge set E is connected, if there exists a relationship between any pair of vertices in G. For a graph G, the degree of a vertex v is the number of edges incident with v and denoted by $d(v)$. A graph can be represented by a numeric number, a polynomial, a sequence of numbers or a matrix that represents the entire graph, and these representations are aimed to be uniquely defined for that graph.

A tree is also a type of graph and can be defined in terms of edges and vertices. To be precise a rooted tree is a Directed Acyclic Graph (DAG) [11]. Tree structures are concretely complacent in computer science and are used in copious range of algorithms. For instance, trees are habituated to construct efficient algorithms for storing and locating items from a list. Considering the examples of B trees and B+ trees that may have many children because of "branching factor" but can locate any data efficiently in $O(\lg n)$ time. Another prominent use of trees is Huffman coding, that construct efficient codes for data transmission and storage. Apart from these applications, trees are being used in traversing of sorted data, workflow for compositing digital images for visual effects and path determination algorithms in networks.

[1]College of Computer Science & Information Systems, Jazan University, Jazan, KSA, e-mail: ahmadsms@gmail.com, mr.ahsan.asim@gmail.com
[2]Department of Applied Mathematics and Informatics, Technical University, Košice, Slovakia, e-mail: martin.baca@tuke.sk
[3]School of Informatics and Applied Mathematics, University Malaysia Terengganu, Kuala Terengganu, Terengganu, Malaysia, e-mail: hroslan@umt.edu.my

To enhance the usability of trees in interdisciplinary research, vertices of tree can be labeled using mathematical definitions, in similar way of graph labeling. On graph labeling lot of work has been done and related scripts are covering the research gaps.

Chartrand *et al.* in [10] introduced an edge *k*-labeling ϕ of a graph *G* such that $w_\phi(x) \neq w_\phi(y)$ for all vertices $x, y \in V(G)$ with $x \neq y$ where weight of a vertex $x \in V(G)$ is $w_\phi(x) = \sum \phi(xy)$ and the sum is over all vertices *y* adjacent to *x*. Such labelings were called *irregular assignments* and the *irregularity strength s*(*G*) of a graph *G* is known as the minimum *k* for which *G* has an irregular assignment using labels at most *k*. This parameter has attracted much attention [5,6,9,12].

In 2007, Bača *et al.* in [8] started to investigate two modifications of the irregularity strength of graphs, namely a *total edge irregularity strength*, denoted by *tes*(*G*), and a *total vertex irregularity strength*, denoted by *tvs*(*G*). Some results on total edge irregularity strength and total vertex irregularity strength can be found in [2-4,7,13,14].

Motivated by these papers, Ahmad *et al.* in [1] introduced the following irregular labeling: A vertex *k*-labeling $\phi: V \to \{1, 2, ..., k\}$ is defined to be an *edge irregular k-labeling* of the graph *G* if for every two different edges *e* and *f* there is $w_\phi(e) \neq w_\phi(f)$, where the weight of an edge $e = xy \in E(G)$ is $w_\phi(xy) = \phi(x) + \phi(y)$. The minimum *k* for which the graph *G* has an edge irregular *k*-labeling is called the *edge irregularity strength* of *G*, denoted by *es*(*G*).

The following theorem that is proved in [1], establishes lower bound for the edge irregularity strength of a graph G.

Theorem 1. [1] *Let* $G = (V, E)$ *be a simple graph with maximum degree* $\Delta = \Delta(G)$. *Then*

$$es(G) \geq \max\left\{\left\lceil \frac{|E(G)|+1}{2} \right\rceil, \Delta(G)\right\}.$$

In this paper, Theorem 1 is mapped on *complete* $m-ary$ *trees* to compute and prove the exact values of the edge irregularity strength using algorithmic approach.

2. Labeling of m-ary trees

In *rooted tree* one vertex is designated as the root and every edge is directed away from the root. The *level* of a vertex *v* in a rooted tree is the length of the unique path from the root to this vertex. The level of the root is defined to be zero. The *height* of a rooted tree is the maximum of the levels of vertices. A vertex of a rooted tree is called a *leaf* if it has no children. Vertices that have children are called *internal* vertices. A rooted tree is called a $m-ary$ *tree* if every internal vertex has no more than *m* children. The tree is called a *full* $m-ary$ *tree* if every internal vertex has exactly *m* children [15].

A *full* $m-ary$ *tree* is a *complete* $m-ary$ *tree* where all leaf vertices are at the same level. A *complete* $m-ary$ *tree* with $m=2$ is called *Complete Binary Tree* $T_{2,h}$ [15], and similarly a *complete* $m-ary$ *tree* with $m=3$ is called *Complete Ternary Tree* $T_{3,h}$. Due to symmetrical arrangements of vertices at each level in *complete* $m-ary$ *tree*, many mathematical properties have been devised. In this article some of those properties are used to formulate some new properties and their proofs are given on $T_{2,h}$ and $T_{3,h}$ and later by generalizing those properties on *complete* $m-ary$ *tree*. In *complete* $m-ary$ *tree* the number of vertices at each level is equal to m^{level}. Hence in *complete* $m-ary$ *tree* of height *h*, the number of vertices in *V* can be calculated using the formula $\frac{(m^{h+1}-1)}{(m-1)}$ and the number of edges in *E* is exactly one less than the number of vertices.

Structure of $T_{3,h}$ is shown in Fig. 1 and it can be observed that root vertex has degree *m*, while m^h leaf-vertices has degree 1 and $\frac{(m^h-m)}{(m-1)}$ vertices are of degree $m+1$.

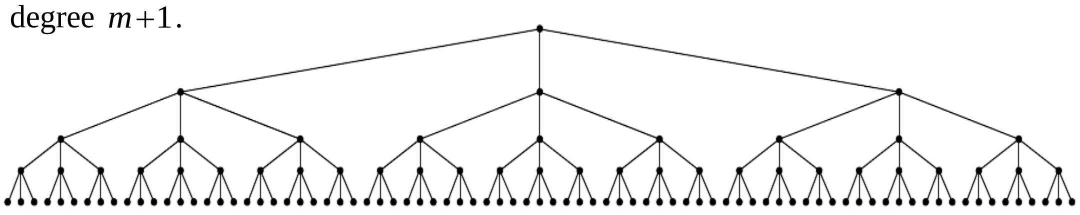

Fig. 1: $T_{m,h}$ = $T_{3,4}$: Complete Ternary Tree of height 4

Let $T_{m,h}$ denote the complete $m-ary$ tree with height *h*. By considering the properties of $m-ary$ tree Theorem 1 can be re-written for $T_{2,h}$ and $T_{3,h}$ as follows:

Theorem 2. *For* $m=2,3$ *and* $h\geq 2$, *let* $T_{m,h}$ *be a complete binary tree* $T_{2,h}$ *and complete ternary tree* $T_{3,h}$, *respectively with V set of vertices and E set of edges. Then*

$$es(T_{m,h})=\left\lceil \frac{|V|}{2} \right\rceil$$

Proof. The number of edges in any tree $T_{m,h}$ are $|V|-1$ and the maximum degree of $T_{2,h}$ and $T_{3,h}$ are 3 and 4, respectively. According to Theorem 1 $es(T_{m,h}) \geq \left\lceil \frac{|V|}{2} \right\rceil$. For computing upper bound an experiment is performed.

Computer Based Experiment

Experiment is conducted to prove the claim empirically, for this purpose a small code is designed and executed to compute the results. Steps of experiment are performed in systematic order [16] like definition, planning, operation, analysis and interpretation to follow the rules of quantitative research. In each operation two major mathematical factors are covered that are type of $m-ary$ tree and its height. For the value of *m*, $T_{2,h}$ and $T_{3,h}$ are considered as simplest structures of $m-ary$ tree whereas height for both trees is increased gradually from 2 to 10 in different operations. Brute-force is applied in recursive calls as algorithm design strategy. To deal with subsets of $T_{2,h}$ and $T_{3,h}$, pre-order traversal is applied to assign the labels of vertices. Weights of edges are changed in increasing and decreasing order alternatively in separate operations. During result analysis at the end of each operation though primary focus was to identify the smallest value of *k*, but also arrangement of labels was observed that *k* reside at what location and is there any pattern exist?

Results of Experiment

After the execution of tiring brute-force experiment, it provided impressive results about the best arrangement for $es(T_{2,h})$ and $es(T_{3,h})$. Following facts are the interpretations for the results of the experiment:

- Minimum value of *k* is exactly same as given in Theorem 2.
- Value of *k* always found at right most child or m^{th} child of root vertex.
- In $T_{2,h}$ for left sub-tree of the root edge weights in ascending sequence whereas for right or m^{th} sub-tree, descending sequence led towards correct results.
- Similarly in $T_{3,h}$ for left and mid sub trees of the root edge weights in ascending sequence whereas for the right most or m^{th} sub-tree, descending sequence led towards correct results.

Results of experiment can be seen as pictorial representation in the Figure 2.

For $m = 2,3$ *and* $h \geq 2$, results of the experiment clearly states $es(T_{m,h}) \leq \left\lceil \frac{|V|}{2} \right\rceil$. By combining the results of experiment and Theorem 2 it is concluded that $es(T_{m,h}) = \left\lceil \frac{|V|}{2} \right\rceil$, for $m = 2,3$ *and* $h \geq 2$ that completes the proof. □

Experiment proved Theorem 2 and it is a good contribution in mathematical domain of graph labeling. But designed code for experiment cannot be applied on bigger trees nor in computer applications due to complexity of brute-force strategy. To complete this research, efficient algorithms are designed by viewing the interesting patterns explored in the experiment. Efficient algorithm for $T_{2,h}$ and ultimately for $m-ary$ trees labeling for any value of $T_{m,h}$ algorithms cost only $O(V)$.

Algorithm for Complete Binary Tree (CBT) Labeling

Input: A positive integer *h* that will be considered as the height of $T_{2,h}$.

Output: Label of vertices $TArray\ [V] \rightarrow \{1,1,2,2,....,k\}$.

Algorithm 1 CBT-Labeling(*h*)

```
1: V ← $2^{h+1} - 1$
2: $TArray[V]$ ← 1
3: Edge-Weight ← 2
4: CBT-Left-Labeling(2)
5: CBT-Right-Labeling(3)
```

TArray[*V*] is a linear array that holds the labels of vertices as outcome of this algorithm. Whereas Edge-Weight is variable that holds the instantaneous values of edges in an ascending fashion for left sub-tree and descending fashion for right sub-tree in Algorithm 2 and Algorithms 3, respectively. For sake of simplicity and efficiency array implementation is used to store the labels of vertices. Therefore, to probe the values of Left-Child, Right-Child and Parent in the array inline functions are used. Left(i) return index location $2*i$, Right(i) return index location $2*i+1$ and Parent(i) return index location $\left\lfloor \frac{V}{2} \right\rfloor$.

Algorithm 2 CBT-Left-Labeling(i)

```
1: if $i \neq NULL$
2:     $TArray[i]$ ← Edge-Weight-TArray[Parent(i)]
3:     Edge-Weight ← Edge-Weight+1
4:     CBT-Left-Labeling(Left(i))
5:     CBT-Left-Labeling(Right(i))
```

Algorithm 3 CBT-Right-Labeling(i)

```
1: if  $i \neq NULL$
2:        if i=3
3:            $TArray[i] \leftarrow \left\lceil \frac{V}{2} \right\rceil$
4:            Edge-Weight $\leftarrow V$
5:          CBT-Right-Labeling(Left(i))
6:          CBT-Right-Labeling(Right(i))
7:      else
8:          $TArray[i] \leftarrow$ Edge-Weight-TArray[Parent(i)]
9:           Edge-Weight $\leftarrow$ Edge-Weight-1
10:          CBT-Right-Labeling(Left(i))
11:          CBT-Right-Labeling(Right(i))
```

Collective outcome of above algorithms, is shown in Figure 2, for $T_{2,4}$ tree where label of vertices is computed by algorithms with unique edge weights.

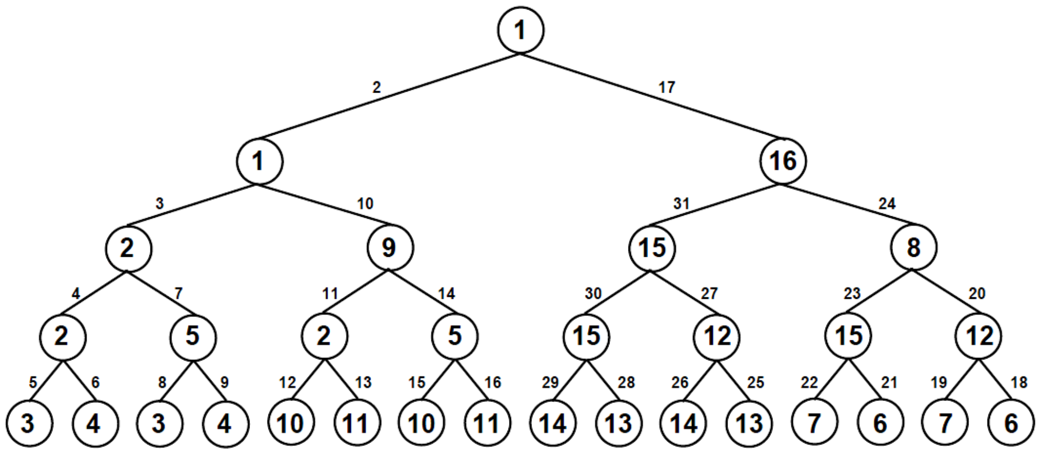

Fig. 2: An edge irregular labeling of complete binary tree $T_{2,4}$.

General Algorithm for *m*-ary Tree Labeling

Input: A positive integer *m* as type of tree and a positive integer $h > 1$ as height of $m - ary$ tree.

Output: Label of vertices $TArray\ [V] \rightarrow \{1, 1, 2, 2, ..., k\}$.

Algorithm 4 *m*-ary-Labeling(h)

```
1:  $V \leftarrow \frac{(m^{h+1} - 1)}{(m-1)}$
2:  Assign 1 as label to root
3:  $TArray[1][1] \leftarrow 0$
```

```
4:  $d \leftarrow \left\lfloor \frac{\left\lceil \frac{|V|}{2} \right\rceil}{(m-1)} \right\rfloor$
5:  **for** each child of root j ← 2 to $m$
6:      $TArray[1][j] \leftarrow TArray[1][j-1] + d$
7:      $TArray[2][j] \leftarrow TArray[1][j] + 1$
8:  $TArray[1][1] \leftarrow 1$
9:  $TArray[2][1] \leftarrow 2$
10: Apply recursive call on 1 to $m$-1 sub-trees in 2nd level
11:     Edge-Weights will increase by 1 starting from 3
12:     Label of vertices will be $TArray[j] \leftarrow$ Edge-Weight-$TArray$[Parent(j)]
13:     Avoid Edge-Weights that are already used in level 1
14: Apply recursive call on $m^{th}$ sub-tree in 2nd level
15:     Edge-Weights will decrease by 1 starting from $V$
16:     Label of vertices will be $TArray[j] \leftarrow$ Edge-Weight-$TArray$[Parent(j)]
```

Figure 3 illustrates the results of *m*-ary-Labeling(h) algorithm pictorially for $T_{8,2}$.

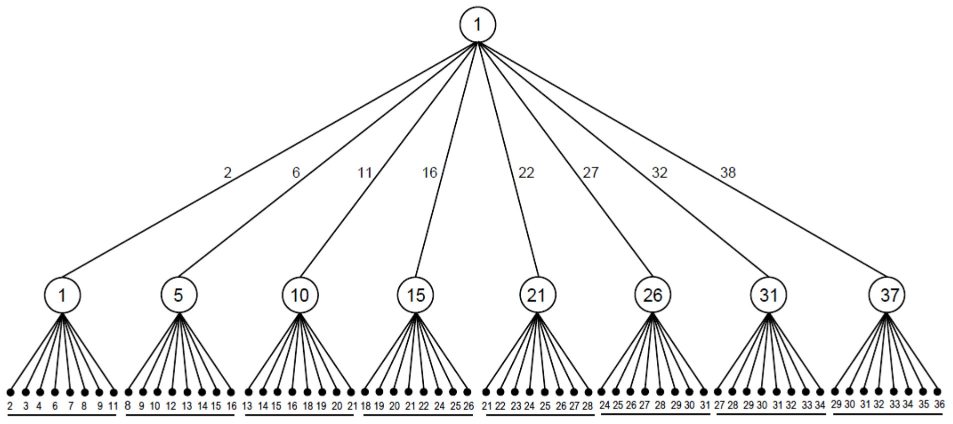

Fig. 3: An edge irregular labeling of $T_{8,2}$.

3. Conclusions

The problem presented in this paper is labeling the vertices of $m-ary$ tree. According to mathematical definition used in this paper, labels of vertices may be repeating to attain a smallest value of $es(T_{m,h}) = \left\lceil \frac{|V|}{2} \right\rceil$ but weights of edges must be unique. Algorithmic solution proved the exact values of the edge irregularity strength of $T_{m,h}$, even algorithm itself is a new dimension in the field of algorithm design and analysis. Given algorithm is valid for any complete $m-ary$ tree as long as resources of computer supports. Labeled trees with unique edge weights can be used in words problem, coding theory, and tree structures.

Acknowledgements
The research for this article is supported by Jazan University Grant 37/7/000114 and the Slovak Science and Technology Assistance Agency under the contract No. APVV-15-0116.

R E F E R E N C E S

[1]. *A. Ahmad, O. Al-Mushayt and M. Bača*, "On edge irregularity strength of graphs", in Appl. Math. Comput. **vol.** 243, 2014, pp. 607-610
[2]. *A. Ahmad, M. Bača, Y. Bashir and M.K. Siddiqui*, "Total edge irregularity strength of strong product of two paths", in Ars Combin. **vol.** 106, 2012, pp. 449-459
[3]. *A. Ahmad, M. Bača and M.K. Siddiqui*, "On edge irregular total labeling of categorical product of two cycles", in Theory of Comp. Syst. **vol.** 54, 2014, pp. 1-12
[4]. *A. Ahmad, E.T. Baskoro and M. Imran*, "Total vertex irregularity strength of disjoint union of Helm graphs", in Discuss. Math. Graph Theory **vol.** 32, no. 3, 2012, pp. 427-434
[5]. *M. Aigner and E. Triesch*, "Irregular assignments of trees and forests", in SIAM J. Discrete Math. **vol.** 3, 1990, pp. 439-449
[6]. *D. Amar and O. Togni*, "Irregularity strength of trees", in Discrete Math. **vol.** 190, 1998, pp. 15-38
[7]. *F. Ashraf, M. Bača, Z. Kimáková and A. Semaničová-Feňovčíková*, "On vertex and edge *H*-irregularity strengths of graphs", in Discrete Math. Algorithm. Appl. **vol.** 8, no. 4, 2016, 13 pages
[8]. *M. Bača, S. Jendroľ, M. Miller and J. Ryan*, "On irregular total labellings", in Discrete Math. **vol.** 307, 2007, 1378-1388
[9]. *T. Bohman and D. Kravitz*, "On the irregularity strength of trees", in J. Graph Theory **vol.** 45, 2004, 241-254
[10]. *G. Chartrand, M.S. Jacobson, J. Lehel, O.R. Oellermann, S. Ruiz and F. Saba*, "Irregular networks", in Congr. Numer. **vol.** 64, 1988, pp. 187-192
[11]. *T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein*, Introduction to Algorithms, The MIT Press, third edition, 2009.
[12]. *A. Frieze, R.J. Gould, M. Karonski and F. Pfender*, "On graph irregularity strength", in J. Graph Theory **vol.** 41, 2002, pp. 120-137
[13]. *S. Jendrol', J. Miškuf and R. Soták*, "Total edge irregularity strength of complete graphs and complete bipartite graphs", in Discrete Math. **vol.** 310, 2010, pp. 400-407
[14]. *Nurdin, E.T. Baskoro, A.N.M. Salman and N.N. Gaos*, "On the total vertex irregularity strength of trees", in Discrete Math. **vol.** 310, 2010, pp. 3043-3048
[15]. *K.H. Rosen*, Discrete Mathematics and its applications, The McGraw-Hill Companies, Inc., seventh edition, 2012
[16]. *C. Wohlin, P. Runeson, M. Host, M.C. Ohlsson, B. Regnell and A. Wesslen*, Experimentation in Software Engineering: An Introduction, Kluwer Academic Publishers, ISBN: 0-7923-8666-3, 2000