Vertex-Magic Total Labeling Algorithms on Unicycle Graphs and Some Graphs Related to Wheels

Denny Riana Silaban, Budi Utami, Alfa Isti Ananda, Dhian Widya, and Siti Aminah
Department of Mathematics, University of Indonesia, Depok, Indonesia
e-mail: denny-rs@ui.ac.id, budi.utami62@ui.ac.id, alfa.isti@ui.ac.id, aminahr@ui.ac.id

Received 17 January 2012, accepted for publication 9 February 2012

Abstract
Let G be a graph with vertex set V and edge set E, where |V| and |E| be the number of vertices and edges of G. A bijection \( \lambda : V \cup E \rightarrow \{1, 2, ..., |V| + |E|\} \) is called a vertex-magic total labeling if there is a constant \( k \) so that the weight of vertex \( x, w_v(x) = \lambda(x) + \sum_{y \in N(x)} \lambda(xy) = k \), \( \forall x \in V \) where \( N(x) \) is the set of vertices adjacent to \( x \). This paper gives algorithms to generate all vertex-magic total labelings on some classes of unicycle graphs (suns and tadpoles) and some classes of graph related to wheels (friendships, fans, generalized Jahangirs). Using those algorithms, we enumerate all non-isomorphic vertex-magic total labelings on those classes of graphs for some values of |V|.

Keywords: Fan, Friendship, Generalized Jahangir, Sun, Tadpole, Unicycle, Vertex magic total labeling, Wheel.

1. Introduction
All graphs considered in this paper are finite, simple, and undirected. The graph \( G \) has vertex set \( V \) and edge set \( E \), and let \( n = |V| \) and \( e = |E| \). The degree of a vertex is a number of edges incident to it.

MacDougall et al. (2002) introduced the notion of a vertex-magic total labeling. The vertex-magic total labeling (VMTL) of a graph \( G \) is a one-to-one mapping from \( V \cup E \) onto the integers \( 1, 2, ..., n + e \) such that there is a constant \( k \) so that for every vertex \( x \),

\[
w_v(x) = \lambda(x) + \sum_{y \in N(x)} \lambda(xy) = k,
\]

where \( N(x) \) is the set of all vertices \( y \) that are adjacent to \( x \). The constant \( k \) is called the magic constant for \( \lambda \) and \( w_v(x) \) is called the weight of \( x \) under labeling \( \lambda \).

The possible values of the magic constant \( k \) of a VMTL can be found using basic counting. By assigning either \( n \) smallest labels or \( n \) largest labels to the vertices, we can obtain the inequality

\[
\frac{13n^2 + 11n + 2}{2(n+1)} \leq k \leq \frac{17n^2 + 15n + 2}{2(n+1)} \quad (1)
\]

which gives the lower and upper bound on the magic constant \( k \) for any graph with \( n \) vertices. This bound is obtained without taking the structure of the graph into account. MacDougall et al. (2002) gave another approach to find the bound on \( k \) which considers the structure of the graph, by applying either \( n \) largest or smallest labels to vertices depending on the degree of the vertices.

Baker and Sawada (2008) gave algorithms that generate all non-isomorphic VMTLs for cycles and wheels. In this paper we use the idea as in Baker and Sawada (2008) to design algorithms that generate all non-isomorphic VMTLs for unicycle graphs (suns and tadpoles) and graphs related to wheel (fans, friendships, and generalized Jahangirs).

2. Background
A path \( P_n \) is a graph with \( n \) vertices and a set of edges \{\( v_1v_2, v_2v_3, ..., v_{n-1}v_n \}\}. By adding an edge \( v_nv_1 \) to path \( P_n \) we will get a cycle \( C_n \).

A sun \( C_n \bigcirc K_1 \) is constructed from a cycle \( C_n \) by adding a leaf to each vertex in a cycle. We call the vertices in the cycle and the leaves as inner and outer vertices.

A tadpole \( T_{m,n} \) is constructed by joining an end vertex of a path \( P_n \) to a vertex of a cycle \( C_m \). A wheel
\( W_n \) consists of a cycle \( C_n \) together with an additional vertex (hub) which is adjacent to each vertex on \( C_n \). We call the edges connecting vertices in the cycle as \( \text{rims} \) and the edges connecting the hub to other vertices as \( \text{spokes} \) of the wheel. If we delete one rim of \( W_n \), we will get a \( \text{fan} \) \( f_n \).

A \( \text{friendship} \) \( F_n \) can be obtained from \( W_n \) with \( n \) by deleting \( \frac{n}{2} \) rims alternately. A friendship \( F_n \) also can be constructed by joining \( n \) copies of \( C_3 \) with a common vertex.

A \( \text{generalized Jahangir} \) \( J_{n,m} \) is a graph on \( nm + 1 \) vertices which consists of a cycle \( C_{nm} \) and another additional vertex that is adjacent to \( m \) vertices of \( C_{nm} \) at distance \( m \) to each other on \( C_{nm} \) (Gallian, 2009). A generalized Jahangir can be obtained from \( W_{nm} \) by deleting all spokes except spokes at distance \( m \) with other spokes.

Baker and Sawada (2008) removed isomorphic labeling for cycle and wheel by avoiding \( \text{rotational symmetry} \) and \( \text{reflective symmetry} \). From observation, for any graph, identifying isomorphic labeling can be done by checking either the rotational symmetry or the reflective symmetry. Both symmetries may arise involving the whole graph or just the subgraph (partial symmetry). For example, for \( \text{tadpole} \) and \( \text{friendship} \), only reflective symmetry holds. While for \( \text{fan} \) and generalized \( \text{Jahangir} \), beside the symmetry that involves the whole graph, there are partial rotational and reflective symmetry also. Figure 1 gives an example of rotational symmetry and partial reflective symmetry on \( f_2 \).

![Figure 1](image)

**Figure 1.** Rotational symmetry (a) and partial reflective symmetry (b) on \( f_2 \).

### 3. Known Results

Slamin \textit{et al.} (2006) proved that for \( n \geq 3 \) and \( t \geq 1 \), the \( t \) copy of sun, \( (C_n \odot R_t) \), is VMTL with \( k = 6nt + 1 \). Furthermore, Rahim and Slamin (pp.) proved that the union of \( t \) sun \( C_{nj} \odot R_1 \) with \( n_j \geq 3 \) for all \( j = 1, 2, \ldots, t \) and \( t \geq 1 \), are VMTLs with \( k = 6 \sum_{j=1}^{t} n_j + 1 \).

MacDougall, Miller, and Wallis (2002) showed that friendship \( F_n \) is VMTL if and only if \( n \leq 3 \) and fan \( f_n \) is VMTL if and only if \( n \leq 10 \).

According to Gallian (2009), Rahim \textit{et al.} prove that generalized \( \text{Jahangir} \) \( J_{t+1} \) is VMTL if \( n = 3 \) and \( t = 1 \) and is not VMTL if \( n \geq 7t + 11 \) and \( t \geq 1 \). As far as we know, there is no result for VMTL of tadpoles has been reported. A further result for VMTL can be found in Gallian (2009).

### 4. The VMTL Algorithms and Results

The general approach of our algorithms is to apply vertex and edge labels by working iteratively around the graphs. In general, the inputs of the algorithm are parameters that give the size of a graph and the magic constant \( k \) we want to compute.

In the VMTL generation algorithm, the cases of rotational and reflective symmetry are removed by assigning some restrictive conditions to vertex or edge labels. The conditions to remove (partial) rotational and reflective symmetry for each graph depend on the structure of the graph. By removing rotational and reflective symmetry we will obtain the non-isomorphic VMTLs for such graphs.

Baker and Sawada (2008) defined a \textit{determined label} as a label which contributes its value to the weight of a vertex for which every other contributing label is known. Assuming the magic constant is given, there is only one possible value for the determined label. They also gave three conditions for the determined label \( \lambda(x) \) which allow us to terminate the recursion tree at any node and backtrack. These conditions are:

1. \( \lambda(x) < 1 \),
2. \( \lambda(x) > n + e \), and
3. \( \lambda(x) \) has already been used in this labeling.

Furthermore, they explained that the aims of the algorithms are to obtain determined labels as quickly as possible. If the given label being applied in the algorithm is determined and the partial labeling is infeasible, then the entire computation subtree rooted at that partial labeling can immediately be excluded. Even in the worst-case scenario, where none of the determined labels eliminates a partial labeling, the use of a determined label reduces the branching factor at a position in the computation tree.

To make the computations faster, Baker and Sawada (2008) parallelize the algorithm to be run on multiple processors. However, our concern is not on the speed of the computation. All results presented in this paper are run in a single processor computer, without parallelizing.

#### 4.1 Sun algorithm

A sun \( C_n \odot R_1 \) has \( 2n \) vertices and edges. We call the inner and outer vertices as \( v_i \) and \( u_i \) respectively, the edges connecting inner vertices as \( e_i \) and the edges connecting inner and outer vertices as \( s_i, i = 1, 2, \ldots, n \). By substituting the number of vertices to (1), we found the possible values of \( k \) for sun are \( 5n + \frac{5}{2} \leq k \leq 7n + \frac{7}{2} \) and by considering the structure of sun as done by MacDougall, Miller, and
Walls (2002), the possible values of \( k \) are \( 5n + 2 \leq k \leq 6n + 1 \). So the possible magic numbers for sun \( C_6 \) or \( K_1 \) are:

\[
5n + \frac{1}{2} \leq k \leq 7n + \frac{1}{2} \land 5n + 2 \leq k \leq 6n + 1.
\]

or

\[
5n + 2 \leq k \leq 6n + 1
\]

The inputs of the algorithm are \( n \) (the number of vertices in \( C_n \)) and the magic number \( k \).

The algorithm consists of 2 functions: initializeSun which sets the labels of a vertex and 3 edges \((v_i, e_1, e_n, s_1)\) and extendSun which recursively labels the remaining vertices and edges (Algorithm 1a-b).

To remove rotational and reflective symmetry we assign \( \lambda(v_i) < \lambda(v_j) \), \( i = 2, 3, \ldots, n \) and \( \lambda(e_i) < \lambda(e_2) \) respectively. Since \( v_i \) must receive the smallest vertex label, it cannot be larger than \( n + 1 \) or there will be insufficient labels to label the remaining vertices.

**Results for Sun**

Table 1 gives the total number of VMTLS on the suns with \( n = 3-7 \) broken down by the magic constant. Although we have use two approaches to find the bound on the magic constant \( k \), our simulation shows that there is no VMTL with the smallest magic constant for suns with \( n = 3-7 \). Further research needs to be done to prove that there is no VMTL for sun with the smallest magic constant.

**Algorithm 1a. function initializeSun()**

```plaintext
1. for each available label \( i \) where \( i \leq 3n+1 \) do
2. \( \lambda(v_i) := i \)
3. avail[i] := false
4. for each available label \( p \) do
5. \( \lambda(e_p) := p \)
6. avail[p] := false
7. for each available label \( j \) where \( j \leq \lambda(e_p) \) do
8. \( \lambda(e_p) := j \)
9. avail[j] := false
10. if \( 0 < \lambda(v_i) \leq 4n \) and avail[\( \lambda(v_i) \)] then
11. avail[\( \lambda(v_i) \)] := true
12. avail[i] := true
```

**Algorithm 1b. function extendSun(t)**

```plaintext
1. if \( t = n \) then
2. for each available label \( i \) where \( i > \lambda(v_1) \) do
3. \( \lambda(v_i) := i \)
4. avail[i] := false
5. \( \lambda(v_2) := k - \lambda(v_1) - \lambda(e_2) - \lambda(e_2) \)
6. if \( 0 < \lambda(v_2) \leq 4n \) and avail[\( \lambda(v_2) \)] then
7. avail[\( \lambda(v_2) \)] := false
8. \( \lambda(v_3) := k - \lambda(v_2) \)
9. if \( 0 < \lambda(v_3) \leq 4n \) and avail[\( \lambda(v_3) \)] then
10. Print()
11. avail[\( \lambda(v_3) \)] := true
12. avail[i] := true
13. else
14. for each available label \( i \) where \( i > \lambda(v_1) \) do
15. \( \lambda(v_i) := i \)
16. avail[i] := false
17. for each available label \( p \) do
18. \( \lambda(e_p) := p \)
19. avail[p] := false
20. \( \lambda(s_1) := k - \lambda(v_1) - \lambda(e_2) - \lambda(e_2) \)
21. if \( 0 < \lambda(s_1) \leq 4n \) and avail[\( \lambda(s_1) \)] then
22. avail[\( \lambda(s_1) \)] := false
23. \( \lambda(u_1) := k - \lambda(s_1) \)
24. if \( 0 < \lambda(u_1) \leq 4n \) and avail[\( \lambda(u_1) \)] then
25. extendSun(\( t+1 \))
26. avail[\( \lambda(u_1) \)] := true
27. avail[i] := true
28. avail[\( \lambda(s_1) \)] := true
29. avail[p] := true
30. avail[i] := true
```

**Table 1.** The number of non-isomorphic VMTLs for sun \( C_3 \odot K_1 \) through \( C_7 \odot K_1 \) broken down by the magic constant \( k \).

<table>
<thead>
<tr>
<th>( C_n \odot K_1 )</th>
<th>( C_6 \odot K_1 )</th>
<th>( C_7 \odot K_1 )</th>
<th>( C_8 \odot K_1 )</th>
<th>( C_9 \odot K_1 )</th>
<th>( C_{10} \odot K_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k )</td>
<td># VMTLS</td>
<td>( k )</td>
<td># VMTLS</td>
<td>( k )</td>
<td># VMTLS</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>23</td>
<td>1</td>
<td>28</td>
<td>12</td>
</tr>
<tr>
<td>19</td>
<td>5</td>
<td>24</td>
<td>6</td>
<td>29</td>
<td>93</td>
</tr>
<tr>
<td>20</td>
<td>25</td>
<td>118</td>
<td>30</td>
<td>190</td>
<td>35</td>
</tr>
<tr>
<td>21</td>
<td>31</td>
<td>213</td>
<td>36</td>
<td>7966</td>
<td>41</td>
</tr>
<tr>
<td>22</td>
<td>37</td>
<td>64778</td>
<td>42</td>
<td>328510</td>
<td>43</td>
</tr>
</tbody>
</table>

\( \Sigma: 6 \) \( \Sigma : 126 \) \( \Sigma : 508 \) \( \Sigma : 79025 \) \( \Sigma : 3272430 \)
4.2 Tadpole algorithm

A tadpole $T_{m,n}$ consists of $C_m$ and $P_n$. We call the vertices and edges in $T_{m,n}$ as $v_i$ and $e_i = v_iv_{i+1}$, $i = 1, 2, \ldots, m$ for vertices in $C_m$ and $i = m+1,m+2,\ldots, m+n$ for vertices in $P_n$. 

Algorithm 2.a. function initializeTadpole()
1. for each available label $i$ do
2. $\lambda(v_i) := i$
3. avail[i] := false
4. for each available label $j$ do
5. $\lambda(e_j) := j$
6. avail[j] := false
7. for each available label $p$ where $p > \lambda(e_1)$ do
8. $\lambda(e_p) := p$
9. avail[p] := false
10. $\lambda(e_{p+1}) := k - \lambda(e_1) - \lambda(e_p)$
11. if $0 < \lambda(e_{p+1}) \leq (m+n)$ and avail[\lambda(e_{p+1})] then
12. avail[\lambda(e_{p+1})] := false
13. extendTadpole()
14. avail[\lambda(e_{p+1})] := true
15. avail[p] := true
16. avail[i] := true
17. avail[j] := false

Algorithm 2.b. function extendTadpole()
1. if $t = n$ then
2. $\lambda(v_{mn}) := k - \lambda(e_{p+1})$
3. if $0 < \lambda(v_{mn}) \leq 2(m+n)$ and avail[\lambda(v_{mn})] then
4. avail[\lambda(v_{mn})] := false
5. finalizeTadpole()
6. avail[\lambda(v_{mn})] := true
7. else
8. for each available label $i$ then
9. $\lambda(v_{mn}) := i$
10. avail[i] := false
11. $\lambda(e_{p+1}) := k - \lambda(e_1) - \lambda(e_p)$
12. if $0 < \lambda(e_{p+1}) \leq (m+n)$ and avail[\lambda(e_{p+1})] then
13. avail[\lambda(e_{p+1})] := false
14. extendTadpole(t+1)
15. avail[\lambda(e_{p+1})] := true
16. avail[i] := true

By substituting the number of vertices on tadpole, $mn$, to (1) and by considering the structure of tadpole as done by MacDougall, Miller, and Wallis (2002), the possible values of $k$ for tadpole $T_{m,n}$ to be VMTL is

$$\left\{\frac{5(m+n)+3}{2} \leq k \leq \frac{7(m+n)+3}{2}\right\} \land \left\{10 \leq k \leq \frac{(m+n)(7(m+n)-3)-14}{2(m+n)-1}\right\}$$

The inputs of the algorithm are $n$, $m$, and the magic constant $k$. The algorithm consists of 3 functions: initializeTadpole which sets the labels of a vertex and 3 edges $(v_1, e_1, e_p, e_{p+1})$, extendTadpole which recursively labels the remaining vertices and edges in $C_m$, and finalizeTadpole which recursively labels the remaining vertices and edges in $P_n$ (Algorithm 2a-c).

The rotational symmetry is not possible in a tadpole and the reflective symmetry is possible only in the cycle subgraph. To remove reflective symmetry we assign $\lambda(e_1) < \lambda(e_n)$.

Algorithm 2.c. function finalizeTadpole()
1. if $t = m$ then
2. $\lambda(v_m) := k - \lambda(e_m) - \lambda(e_n)$
3. if $0 < \lambda(v_m) \leq 2(m+n)$ and avail[\lambda(v_m)] then
4. Print()
5. else
6. for each available label $i$ then
7. $\lambda(v_m) := i$
8. avail[i] := false
9. $\lambda(e_m) := k - \lambda(e_1) - \lambda(e_n)$
10. if $0 < \lambda(e_m) \leq 2(m+n)$ and avail[\lambda(e_m)] then
11. avail[\lambda(e_m)] := false
12. finalizeTadpole(t+1)
13. avail[\lambda(e_m)] := true
14. avail[i] := true

Table 2. The number of non-isomorphic VMTLs for tadpole $T(3,1)$-$T(3,5)$, $T(4,1)$-$T(4,5)$, $T(5,1)$-$T(5,5)$, $T(6,1)$-$T(6,5)$, $T(7,1)$-$T(5,5)$ broken down by the magic constant ($k$).

<table>
<thead>
<tr>
<th>$T(3,1)$</th>
<th>$T(3,2)$</th>
<th>$T(3,3)$</th>
<th>$T(3,4)$</th>
<th>$T(3,5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$</td>
<td>VM TLs</td>
<td>$k$</td>
<td>VM TLs</td>
<td>$k$</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>14</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>16</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>20</td>
<td>6</td>
<td>22</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>24</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>25</td>
<td>10</td>
<td>28</td>
<td>22</td>
<td>29</td>
</tr>
<tr>
<td>$\Sigma : 1$</td>
<td>$\Sigma : 4$</td>
<td>$\Sigma : 26$</td>
<td>$\Sigma : 234$</td>
<td>$\Sigma : 724$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$T(4,1)$</th>
<th>$T(4,2)$</th>
<th>$T(4,3)$</th>
<th>$T(4,4)$</th>
<th>$T(4,5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$</td>
<td>VM TLs</td>
<td>$k$</td>
<td>VM TLs</td>
<td>$k$</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>17</td>
<td>6</td>
<td>19</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>18</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>19</td>
<td>10</td>
<td>21</td>
</tr>
<tr>
<td>17</td>
<td>4</td>
<td>20</td>
<td>4</td>
<td>22</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>21</td>
<td>2</td>
<td>23</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>24</td>
<td>9</td>
<td>27</td>
</tr>
<tr>
<td>25</td>
<td>0</td>
<td>28</td>
<td>18</td>
<td>30</td>
</tr>
<tr>
<td>29</td>
<td>0</td>
<td>31</td>
<td>257</td>
<td></td>
</tr>
<tr>
<td>$\Sigma : 12$</td>
<td>$\Sigma : 24$</td>
<td>$\Sigma : 117$</td>
<td>$\Sigma : 812$</td>
<td>$\Sigma : 4512$</td>
</tr>
</tbody>
</table>
Table 2 shows that for some value of the magic constant, from this simulation we can conclude the tadpole is VMTL for some values of $k$. Table 2 also shows that for some value of $k$, there is no VMTL found. Further research need to be done to prove it.

### 4.3 Friendship algorithm

A friendship $F_n$ consists of $n$ copies of $C_3$ with a common vertex. The number of vertices and edges in $F_n$ are $2n + 1$ and $3n$. We call the hub, vertices, rims, and spoke as $c, v_i (i = 1, 2, ..., 2n), r_i (i = 1, 3, ..., 2n)$ and $s_i (i = 1, 2, ..., 2n)$.

By substituting the number of vertices to (1) and by considering the structure of $F_n$ as done by MacDougall, Miller, and Wallis (2002), the possible values of $k$ for friendship $F_n$ are

$$\left(17n^2 + 9n + 1 \leq k \leq 23n^2 + 12n + 1\right)$$

$$\cap \left(2n + 1)(n + 1) \leq k \leq 17n + 9\right)$$

The inputs of the algorithm is $n$ (the number of $C_3$ in $F_n$) and the magic constant $k$. The algorithm consists of 3 functions: friendship1 which recursively labels $v_i$, $r_i$, $s_i$ for odd $i$, friendship2 which recursively labels $v_i$, $r_i$ for even $i$, and finalizedfriendship which sets the labels of $s_{2n}$, $v_{2n}$, and $c$ (Algorithm 3a-c).

To remove rotational symmetry we assign $\lambda(s_i) < \lambda(s_i)$, $i = 1, 3, ..., 2n - 1$. To remove partially rotational symmetry we assign $\lambda(s_i) < \lambda(s_i) < ... < \lambda(s_{2n}).$ To remove reflective symmetry we assign $\lambda(s_{i1}) < \lambda(s_i), i = 2, 4, ..., 2n.$ To remove partially reflective symmetry we assign $\lambda(s_{i1}) < \lambda(s_i), i = 2, 3, 4, ..., 2n.$ Since $\lambda(s_{i1}) < \lambda(s_i), i = 2, 3, 4, ..., 2n$, the label of $s_i$ cannot be bigger than $3n + i + 1$ or there will be insufficient labels to label the remaining vertices.

#### Algorithm 3. a. function friendship1 ($t$)

1. if $t = 1$ then
2. for each available label $i$ do
3. $\lambda(s_1) := i$
4. avail [$i$] := false
5. for each available label $j$ do
6. $\lambda(v_1) := j$
7. avail [$j$] := false
8. $\lambda(v_i) := k - \lambda(s_i) - \lambda(r_i)$
9. if $0 < \lambda(v_i) \leq 5n+1$ and avail [$\lambda(v_i)$] then
10. avail [$\lambda(v_i)$] := false
11. friendship2($2$)
12. $\lambda(v_{i1}) := true$
13. avail [$i$] := true
14. avail [$i$] := true
15. else
16. for each available label $\geq \lambda(s_{i2})$ do
17. $\lambda(s_1) := i$
18. avail [$i$] := false
19. for each available label $j$ do
20. $\lambda(r_1) := j$
21. avail [$j$] := false
22. $\lambda(v_i) := k - \lambda(s_i) - \lambda(r_i)$
23. if $0 < \lambda(v_i) \leq 5n+1$ and avail [$\lambda(v_i)$] then
24. avail [$\lambda(v_i)$] := false

Results for tadpole

Table 2 gives the total number of VMTLs on the tadpole $T_{m,n}, m = 1-7, n = 1-5$ broken down by the magic constant. From this simulation we can conclude that tadpole $T_{m,n}$ is VMTL for $m = 1-7, n = 1-5$. Table 2 also shows that for some value of $k$, there is no VMTL found. Further research need to be done to prove it.
friendship2(r+1)
26. avail [l(vi)]:= true
27. avail [l]:= true

Algorithm 3b. function friendship2(i)
1. if i = 2n then
2. friendship3
3. else
4. for each available label [i > l(s_i)] do
5. l(s_i):= i
6. avail [l]:= false
7. l(v_i):= k - l(s_i) - l(v_i)
8. if 0 < l(v_i) ≤ 5n+1 and avail [l(v_i)] then
9. avail [l(v_i)]:= false
10. friendship1(r+1)
11. avail [l(v_i)]:= true
12. avail [l]:= true

Algorithm 3c. function finalizefriendship
1. for each available label [i > l(s_i)] do
2. l(s_i):= i
3. avail [l]:= false
4. l(v_i):= k - l(s_i) - l(v_i)
5. if 0 < l(v_i) ≤ 5n+1 and avail [l(v_i)] then
6. avail [l(v_i)]:= false
7. λ(c):= k - Σ (l(s_i))
8. if 0 < λ(c) ≤ 5n+1 and avail [l(c)] then
9. print
10. avail [l(v_i)]:= true
11. avail [l]:= true

Results for friendship

Table 3 gives the total number of VMTLs on the friendships F_n for n = 3 – 7 broken down by the magic constant k. As mentioned in the known result, MacDougall, Miller, and Wallis (2002) shows that friendship F_n is VMTL if and only if n ≤ 3. Our simulation enumerates all VMTL friendships. There are only 52 non-isomorphic VMTLs for friendships.

Table 3. The number of non-isomorphic VMTLs for friendship F_n broken down by the magic constant (k).

<table>
<thead>
<tr>
<th>n</th>
<th>#VMTLS</th>
<th>n</th>
<th>#VMTLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>1</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Σ: 7</td>
<td>Σ: 45</td>
</tr>
</tbody>
</table>

4.4 Fan algorithm

A fan f_i can be obtained from W_n by deleting a rim. The number of vertices and edges in f_i are n + 1 and 2n - 1. We call the hub, vertices, rims, and spoke as c, v_i (i = 1, 2, ..., n), r_i (i = 1, 2, ..., n - 1), and s_i (i = 1, 2, ..., n). By substituting the number of vertices to (1) and by considering the structure of f_i as done by MacDougall, Miller, and Wallis (2002), the possible values of k are

\[
\frac{13b^2 + n}{2(n+1)} \leq k \leq \frac{17b^2 + 3n - 2}{2(n+1)}
\]

Algorithm 4a. function fan(f)
1. if i = n then
2. finalizefan
3. else
4. for each available label i do \(l(v_i) = i\)
5. avail [l]:= false
6. for each available label j do
7. \(l(r_i) = j\)
8. avail [l]:= false
9. if i = 1
10. \(l(v_i) = k - l(s_i) - l(r_i)\)
11. else
12. \(l(v_i) = k - l(s_i) - l(v_i) - l(r_i)\)
13. if 0 < \(l(v_i) \leq 3n\) and avail [l(v_i)] then
14. avail [l(v_i)]:= false
15. fan (+1)
16. avail [l(v_i)]:= true
17. avail [l]:= true

Algorithm 4b. function finalizefan
1. for each available label [i > l(s_i)] do
2. l(s_i):= i
3. avail [l]:= false
4. l(v_i):= k - l(s_i) - l(v_i)
5. if 0 < l(v_i) ≤ 3n and avail [l(v_i)] then
6. avail [l(v_i)]:= false
7. λ(c):= k - Σ (l(s_i))
8. if n = 3 then
9. if \(l(v_i) < λ(c) \leq 3n\) and avail [l(c)] then
10. print
11. avail [l(v_i)]:= true
12. else
13. if 0 < λ(c) ≤ 3n and avail [l(c)] then
14. print
15. avail [l(v_i)]:= true
16. avail [l]:= true

The rotational symmetry is possible only in f_3. To remove it we assign \(l(v_i) < λ(c)\). To remove reflective symmetry we assign \(l(s_i) < l(s_i)\), i = 2, 3, 4, ..., n. Since \(λ(s_i) < λ(s_i)\), i = 2, 3, 4, ..., n. The label of s_1 cannot be larger than 2n.
Results for fan

Table 4 gives the total number of VMTLs on the fan $f_n$ with $n = 3 - 7$ broken down by magic constant. MacDougall, Miller, and Wallis (2002) shows that friendship $F_n$ is VMTL if and only if $n \leq 10$. Because of time constraint, we cannot enumerate all non-isomorphic VMTLs for fan. Table 4 also shows that there is no VMTL for $f_7$ with $k = 15, 18, 20$.

### 4.5 Generalized Jahangir algorithm

A generalized Jahangir $J_{n,m}$ can be obtained from $W_{nm}$ by deleting all spokes except spokes at distant $m$ with other spokes. The number of vertices and edges in $J_{n,m}$ are $nm + 1$ and $n(m + 1)$. The way we name the hub $c$, vertices $v_i$, rims $r_m$, and spokes $s_k$ of $J_{n,m}$ is given in Figure 2.

![Figure 2. The name of hub, vertices, rims, and spokes of $J_{n,m}$](image)

By substituting the number of vertices to (1) and by considering the structure of $J_{n,m}$ as done by MacDougall, Miller, and Wallis (2002), the possible values of $k$ for $J_{n,m}$ to be VMTL are

$$k \leq \frac{7n^2m^2 + 8nm^2 + 9nm + 2n^2 + 6n + 2}{2(nm + 1)}$$

Algorithm 5a. function jahangir1

1. for each available label $i$ do
2. $\lambda(s_i) := i$
3. avail $[i] := \text{false}$
4. for each available label $j$ do
5. $\lambda(r_m) := j$
6. avail $[j] := \text{false}$
7. for each available label $\lambda(v_i)$ do
8. $\lambda(r) := i$
9. avail $[i] := \text{false}$
10. $\lambda(v_i) := k - \lambda(s_i) - \lambda(r) - \lambda(r_m)$
11. if $0 < \lambda(v_i) \leq 2nm + n + 1 \text{ and avail } [\lambda(v_i)]$ then
12. avail $[\lambda(v_i)] := \text{false}$
13. jahangir2(2)
14. avail $[\lambda(v_i)] := \text{true}$
15. avail $[i] := \text{true}$
16. avail $[j] := \text{true}$
17. avail $[i] := \text{true}$

Algorithm 5b. function jahangir2(t)

1. if $t = nm$
2. jahangir4(t)
3. else
4. for each available label $i$ do
5. $\lambda(r) := i$
6. avail $[i] := \text{false}$
7. $\lambda(v_i) := k - \lambda(r) - \lambda(r_m)$
8. if $0 < \lambda(v_i) \leq 2nm + n + 1 \text{ and avail } [\lambda(v_i)]$ then
9. avail $[\lambda(v_i)] := \text{false}$
10. jahangir3(+1)
11. avail $[\lambda(v_i)] := \text{true}$
12. avail $[i] := \text{true}$

Algorithm 5c. function jahangir3(t)

1. if $t = nm - m + 1$
2. jahangir2(t)
3. elseif $(r \mod m) \neq 1$
4. jahangir2(t)
5. else
6. for each available label $\lambda(v_i)$ do
7. $\lambda(s_i) := i$
8. avail $[i] := \text{false}$
9. for each available label $j$ do
10. $\lambda(r) := j$
11. avail $[j] := \text{false}$
12. $\lambda(v_i) := k - \lambda(s_i) - \lambda(r) - \lambda(r_m)$
13. if $0 < \lambda(v_i) \leq 2nm + n + 1 \text{ and avail } [\lambda(v_i)]$ then
14. avail $[\lambda(v_i)] := \text{false}$
15. jahangir2(+1)
16. avail $[\lambda(v_i)] := \text{true}$
17. avail $[j] := \text{true}$
18. avail $[i] := \text{true}$

Algorithm 5d. function finalizejahangir(t)

1. if $t = nm$ then
2. if $nm = 4$ then
3. $\lambda(v_m) := k - \lambda(r_m) - \lambda(r_m)$
4. if $\lambda(c) < \lambda(v_m) \leq 2nm + n + 1 \text{ and avail } [\lambda(v_m)]$ then
5. print
6. else
7. $\lambda(v_m) := k - \lambda(r_m) - \lambda(r_m)$
8. if $0 < \lambda(v_m) \leq 2nm + n + 1 \text{ and avail } [\lambda(v_m)]$ then
9. print
10. else
for each available label \( i \) do \\
\[ \lambda(s_i) = i \] \\
avail \([i]\) := false \\
in if \( nm = 4 \) \\
\[ \lambda(c) := k - \sum_{i=1}^{n} s_i \] \\
16. if \( \lambda(v_i) < \lambda(c) \leq 2nm + n + 1 \) and avail \([\lambda(c)]\) \\
then \\
17. avail \([\lambda(c)]\) := false \\
18. for each available label \( j \) do \\
19. \[ \lambda(r_i) := j \] \\
20. avail \([j]\) := false \\
21. \[ \lambda(v_i) := k - \lambda(s_i) - \lambda(r_i) - \lambda(r_{s_i}) \] \\
22. if \( 0 < \lambda(v_i) \leq 2nm + n + 1 \) and avail \([\lambda(v_i)]\) \\
then \\
23. avail \([\lambda(v_i)]\) := false \\
24. jahangir2(\(t+1\)) \\
25. avail \([\lambda(v_i)]\) := true \\
26. avail \([j]\) := true \\
27. avail \([\lambda(c)]\) := true \\
28. else \\
\[ \lambda(c) := k - \sum_{i=1}^{n} s_i \] \\
29. if \( 0 < \lambda(c) \leq 2nm + n + 1 \) and avail \([\lambda(c)]\) then \\
31. avail \([\lambda(c)]\) := false \\
32. for each available label \( j \) do \\
33. \[ \lambda(r_i) := j \] \\
34. avail \([j]\) := false \\
35. \[ \lambda(v_i) := k - \lambda(s_i) - \lambda(r_i) - \lambda(r_{s_i}) \] \\
36. if \( 0 < \lambda(v_i) \leq 2nm + n + 1 \) and avail \([\lambda(v_i)]\) \\
then \\
37. avail \([\lambda(v_i)]\) := false \\
38. jahangir2(\(r+1\)) \\
39. avail \([\lambda(v_i)]\) := true \\
40. avail \([j]\) := true \\
41. avail \([\lambda(c)]\) := true \\

The inputs of the algorithm are \( n, m \), and the magic constant \( k \). The algorithm consists of 4 functions: jahangir1 which sets the labels of one vertex \( v_1 \) and 3 edges \((s_1, r_1, r_{s_1})\), jahangir2 which recursively labels vertices that are not adjacent to hub \( c \), jahangir3 which recursively labels \( v_2, r_2, s_i \) for \( i = 1 \) (mod \( m \)), and finalJedahangir which sets the labels of \( v_0, v_{i-1}, v_{i+1}, s_0, s_{i-1}, s_{i+1} \), and \( c \) (Algorithm 5a-d).

To remove rotational and reflective symmetry we assign \( \lambda(s_i) < \lambda(s_i) \), \( i = 2, 3, \ldots, nm \) and \( \lambda(r_i) < \lambda(s_i) \) respectively. Especially for \( J_{2,2} \) we must assign \( \lambda(v_2) < \lambda(c) < \lambda(v_4) \).

Results for Generalized Jahangir

Table 5 gives the total number of VMTLs on the generalized Jahangir \( J_{2,2} - J_{2,5}, J_{2,2} - J_{3,3}, J_{4,2} \) broken down by the magic constant. Table 5 shows that there is no VMTL with the largest magic constant for generalized jahangir \( J_{2,2} - J_{2,5}, J_{2,2} - J_{3,3}, J_{4,2} \). Further research needs to be done to prove the non-existence of VMTL for generalized Jahangir with the largest magic constant.

Table 5. The number of non-isomorphic VMTLs for generalized Jahangir \( J_{2,2} - J_{2,5}, J_{2,2} - J_{3,3}, J_{4,2} \) broken down by the magic constant (\( k \)).

5. Conclusion

In this paper, we give algorithms to enumerate all non-isomorphic VMTL on two unicycle graphs and three graphs related to wheel. The idea of the algorithms can be applied to other classes of graphs or adopted to develop algorithms for other type of labeling. In the mean time, we are still working on algorithm for other type of labeling such as edge magic total, vertex anti magic total, harmonious, and graceful.

We also present the number of non-isomorphic VMTLs on each graph for some small size graphs. As the number of non-isomorphic VMTLs for friendship has been enumerated, the number of non-isomorphic labeling on larger size of the remaining graphs is still an open problem. Our result also indicates some open problems to make the
lower or upper bound on the magic constant \( k \) for sun, tadpole, and generalized Jahangir more tight.

**Acknowledgements**

A part of this research was funded by Hibah Kompetensi Dikti No. 239/SP2H/PP/DP2M/III/2010.

**References**


Rahim M. T., and Slamin, Vertex-magic total labeling of the union of suns, pp.