Far East Journal of Mathematical Sciences (FJMS)
Volume 42, Number 2, 2010, Pages 161-173
Published Online: September 9, 2010
This paper is available online at http://pphmj.com/journals/fjms.htm

AN APPLICATION OF MATHEMATICAL PROGRAMMING IN VERTEX-MAGIC TOTAL LABELING PROBLEMS

PARHAM AZIMI

Islamic Azad University (Qazvin Branch)
No. 21, Shahid Fallahi St. (Zaferanieh)
Vali-e-Asr Ave.
Tehran, Iran
e-mail: p.azimi@setarehiran.com

Abstract

A labeling (or valuation) of a graph $G = (V, E)$ is a map that carries graph elements to the special numbers (usually to the positive or non-negative integers). There are some famous labeling methods such as graceful labeling, harmonious labeling, magic-type total labeling, antimagic-type labeling and the other miscellaneous labelings. Among magic-type labelings, a well known one is vertex-magic total labeling. For a graph $G = (V, E)$, an injective mapping f from $V \cup E$ to the set $\{1, 2, ..., |V| + |E|\}$ is a vertex-magic total labeling if there is a constant k, called the magic constant, such that for every vertex v, $f(v) + \Sigma f(vu) = k$, where the sum is over all vertices u adjacent to v.

In this paper, at first, an integer programming model has been developed for formulating a vertex-magic total labeling of a given graph and then a branch and bound algorithm has been developed to solve the correspondent problem for finding the vertex-magic total labeling of the

2010 Mathematics Subject Classification: 05Cxx.

Keywords and phrases: graph labeling, vertex-magic total labeling, mathematical programming.

Received July 23, 2009; Revised December 2, 2009

graph. Finally, the algorithm has been extensively tested on a set of randomly generated graphs in different classes.

1. Introduction

All graph labeling methods trace their origin to the one which was introduced by Rosa [1] in 1967 or the one given by Graham and Sloane [2] in 1980. A labeling (or valuation) of a graph G is a map that carries graph elements to some special numbers (usually to the positive or non-negative integers). If we review all studies from that time to the current time, we can find that major studies are focused on proving a special labeling for different classes of graphs. For instance, we can refer to the one that was collected by Gallian [3] which covers all studies conducted in the field of graph labeling. But since the graph labeling problems have some applications in industries [4], therefore some researches focused on how to label a graph by using programming languages [5], i.e., constraint programming or mathematical programming [6, 7]. In these studies, the researchers intended to develop an algorithm to find a feasible solution of a special labeling for a given graph. The aim of this paper is as the same as such works. However, in this section, we try to overview the origin of graph labeling problems as well.

Rosa [1] called a function f, a β-valuation of a given graph $G = (V, E)$, if f is an injective from the V to the set $\{1, 2, ..., |V|\}$ such that we assign label $|f(u) - f(v)|$ to the edge uv, then the resulting edge labels are distinct. The most common choices of domain are the set of all vertices and edges (such a labeling is called *total labeling*), the vertex-set alone (vertex-labeling), or the edge-set alone (edge-labeling). Other domains are also possible. Golomb [8] called such a labeling as *graceful labeling*.

Gallian [3] categorized all labelings into 5 groups and called them: Graceful labeling, Harmonious labeling, Magic-type labeling, Antimagic-type labeling and Miscellaneous labeling. Since the paper domain is magic-type labeling, so we neglect other labelings, however, we may find the definitions and the recent researches in [3].

Magic labeling motivated by the notion of magic squares in number theory but magic labeling was introduced by Sedlàček [9] in 1963. Based on Sedlàček's work,

Stewart [10, 11] studied various ways to label the edges of a given graph in the mid 1960s. He defined "semi-magic" labeling of a connected graph if we can label the edges with integers such that for each vertex v, the sum of the labels of all edges incident to v is the same for all v. A semi-magic labeling is called "*magic labeling*" if the edges are labeled with distinct positive integers. Stewart called a magic labeling "*supermagic*" if the set of edge labels consists of consecutive positive integers. Stewart [9] proved the following: K_n is magic for $n = 2$ and all $n \geq 5$; $K_{n,n}$ is magic for all $n \geq 3$; Fans F_n are magic iff n is odd and $n \geq 3$; Wheels W_n are magic for all $n \geq 3$; and W_n with one spoke deleted is magic for $n = 4$ and for all $n \geq 6$. He also proved that K_n is supermagic iff $n > 5$ and $n \neq 0 \pmod 4$. We may find the other researches in this field in [3].

In the literature of magic labeling, there are two different categories: edge-magic total labeling and vertex-magic total labeling. For edge-magic total labeling or as an abbreviation EMTL, one can find the details in [3] and [7].

But MacDougall et al. [12] introduced the notion of vertex-magic total labeling in 1999. For a graph $G = (V, E)$, an injective mapping f from $V \cup E$ to the set $\{1, 2, ..., |V| + |E|\}$ is a vertex-magic total labeling (VMTL) if there is a constant k, called the *magic constant*, such that for every vertex v, $f(v) + \Sigma f(vu) = k$, where the sum is over all vertices u adjacent to v, however, some authors use the term "vertex-magic" for this concept. They proved that the following graphs have vertex-magic total labeling (VMTL): C_n; P_n $(n > 2)$; $K_{m,m}$ $(m > 1)$; $K_{m,m} - e$ $(m > 2)$ and K_n for n odd and they also proved that when $n > m + 1$, $K_{m,n}$ does not have VMTL. They conjectured that $K_{m,m+1}$ has a VMTL for all m and that K_n has VMTL for all $n \geq 3$. The latter conjecture was proved by Lin and Miller [13]. Finally, Gray et al. [14] have shown that all complete graphs are VMTL. In [15], McQuillan introduced a technique for constructing VMTL of 2-regular graphs. Specially, if m is a positive integer and J is any subset of $I = \{1, 2, ..., k\}$, then $(\bigcup_{i \in J} mC_{ni}) \cup (\bigcup_{i \in I-J} mC_{ni})$ has VMTL. The summary of VMTL of different graphs has been shown in the following table:

Table 1. The latest results of VMTL

Graph	Conditions
C_n	[12]
P_n	$n > 2$ [12]
$K_{m,m} - e$	$m > 2$ [12]
$K_{m,n}$	Iff $\lvert m - n \rvert \ll 1$ [12, 16, 17]
K_n	For n odd [12] For $n \equiv 2 \pmod 4$, $n > 2$ [13]
nK_3	Iff $n \neq 2$ [18-20]
mK_n	$m \geq 1$, $n \geq 4$ [21]
Peterson $P(n, k)$	[22]
Prisms $C_n \times P_2$	[23]
W_n	Iff $n \leq 11$ [12, 17]
F_n	Iff $n \leq 10$ [12, 17]

Meissner and Zwierzyński [24] used VMTL of graphs as a way to compare the efficiency of parallel execution of a program versus sequential processing as an application of VMTL. We can find more details about the latest results in Gallian [3].

Now, we are ready to define the integer programming model and develop an algorithm for solving it.

2. Integer Programming (0-1) Model of Vertex-magic Total Labeling Problem

Denote the vertices of the graph $G = (V, E)$ by $v_1, v_2, ..., v_n$, which has n vertices and m edges. Now, we define the decision variables of the model as follows:

x_i The label of vertex v_i, $i = 1, 2, ..., n$.

x_{ij} The label of an edge (v_i, v_j), $i, j = 1, 2, ..., n$, $i \neq j$.

k The magic constant.

w_{it} A 0-1 variable, where $w_{it} = 1$ implies that $x_i = t$, $i = 1, 2, ..., n$, $t = 1, 2, ..., m + n$.

r_{ijt} A 0-1 variable, where $r_{ijt} = 1$ implies that $x_{ij} = t$, $i, j = 1, 2, ..., n$, $i \neq j$, $t = 1, 2, ..., m + n$.

The following 0-1 model has a feasible solution iff $G = (V, E)$ has vertex-magic total labeling:

Problem VMTL

(1) $x_i + \sum_{(v_i, v_j) \in E(G)} x_{ij} = k, \quad \forall i = 1, 2, ..., n.$

(2) $x_i = \sum_{t=1}^{m+n} t w_{it}, \quad \forall i = 1, 2, ..., n.$

(3) $x_{ij} = \sum_{t=1}^{m+n} t r_{ijt}, \quad \forall (v_i, v_j) \in E(G).$

(4) $\sum_{t=1}^{m+n} w_{it} = 1, \quad \forall i = 1, 2, ..., n.$

(5) $\sum_{t=1}^{m+n} r_{ijt} = 1, \quad \forall (v_i, v_j) \in E(G).$

(6) $\sum_{i=1}^{n} w_{it} + \sum_{(v_i, v_j) \in E(G)} r_{ijt} = 1, \quad \forall t = 1, 2, ..., m + n.$

(7) $x_i, x_{ij}, k \geq 0, \quad \forall i = 1, 2, ..., n, \forall (v_i, v_j) \in E(G).$

(8) $w_{it}, r_{ijt} = \{0, 1\}, \quad \forall i = 1, 2, ..., n, \forall (v_i, v_j) \in E(G), \forall t = 1, 2, ..., m + n.$

Since the aim is to find a feasible solution of the problem VMTL and no optimization is required, problem VMTL has no objective function. As we may see, since all labels for vertices and edges were defined based on some positive integers, so this is a pure 0-1 programming model. In the above model, the first constraints are related to the definition of a vertex-magic total labeling problem. The second and third constraints define vertex and edge labels based on positive integers from 1 to $m + n$. The fourth, fifth and sixth constraints cause the vertex and edge labels to be distinct positive integers from 1 to $m + n$.

The total number of constraints (1)-(6) of problem VMTL is $4n + 3m$ and the total number of variables is $(n + m)(n + m + 1) + 1$.

3. A Branching Method for Solving Graph Labeling Problems

It should be mentioned that the suggested algorithm is somehow similar to the one that proposed by Eshghi and Azimi [7] with some modifications due to the definition of VMTL of a graph. The modifications occurred in "Branching Strategy" and "Separation Rule", especially when there is a tie up. According to the computational results, this version is much more powerful than the one in [7]. Since the problem VMTL is a pure 0-1 programming model, a branch and bound algorithm with some modifications has been developed to solve the model in suitable time. At first, the algorithm starts with the relaxation of constraints (8). The resulting submodel is a linear programming one and it is very easy to be solved. The termination rule in the algorithm is defined as follows:

3.1. The termination rule

The algorithm terminates whenever the corresponding solution from the relaxed subproblem satisfies the integrality constraints (8) or all nodes have been fathomed. If a node has not any feasible solution or its optimum solution does not lie between the upper and the lower bounds, then it should be fathomed. It should be mentioned that in a branch and bound algorithm, defining efficient upper and lower bounds can increase the efficiency of the algorithm, since it terminates faster. Therefore, the bounds are defined as follows:

3.2. The upper bound

Since the model (VMLT) has not any objective function, it was defined a dummy objective function to robust the branch and bound algorithm. So we redefine the VMLT as follows:

Problem VMTL1

Max. $Z = K$

$$(1)\ x_i + \sum_{(v_i, v_j) \in E(G)} x_{ij} = k, \quad \forall i = 1, 2, ..., n.$$

$$(2)\ x_i = \sum_{t=1}^{m+n} tw_{it}, \quad \forall i = 1, 2, ..., n.$$

$$(3)\ x_{ij} = \sum_{t=1}^{m+n} tr_{ijt}, \quad \forall (v_i, v_j) \in E(G).$$

(4) $\sum_{t=1}^{m+n} w_{it} = 1, \quad \forall i = 1, 2, ..., n.$

(5) $\sum_{t=1}^{m+n} r_{ijt} = 1, \quad \forall (v_i, v_j) \in E(G).$

(6) $\sum_{i=1}^{n} w_{it} + \sum_{(v_i, v_j) \in E(G)} r_{ijt} = 1, \quad \forall t = 1, 2, ..., m + n.$

(7) $x_i, x_{ij}, k \geq 0, \quad \forall i = 1, 2, ..., n, \forall (v_i, v_j) \in E(G).$

(8) $0 \leq w_{it}, r_{ijt} \leq 1, \quad \forall i = 1, 2, ..., n, \forall (v_i, v_j) \in E(G), \forall t = 1, 2, ..., m + n.$

When problem VMTL1 is being relaxed from constraints (8) and solved, the optimum solution of VMTL1 is an upper bound for k, i.e.,

$$\text{Upper bound for } k = Z^*_{\text{VMTL1}} \quad \text{or} \quad k \leq Z^*_{\text{VMTL1}}.$$

It should be mentioned that a given graph may not have only one vertex-magic labeling. So the vertices, edges and the magic constant are not unique for a given graph. Therefore, we can label a graph by different distinct positive integers. Finally, knowing the maximum value of k is the most important issue for the upper bound. According to Wallis [25], if a regular graph has a VMTL, then we can create a new VMTL from it. Given the VMTL λ for a given graph, define the map $\acute{\lambda}$ on $E \bigcup V$ by:

$$\acute{\lambda}(v) = m + n + 1 - \lambda(v),$$
$$\acute{\lambda}(uv) = m + n + 1 - \lambda(uv),$$

so, $\acute{\lambda}(v)$ and $\acute{\lambda}(uv)$ are the new VMTL for each vertex and edge.

3.3. The lower bound

As Wallis [25] explained in his works, if we label the edges with minimum possible labels, then a lower bound can be found as follows:

$$k \geq \frac{\binom{m+n+1}{2} + \binom{m+1}{2}}{n}.$$

These upper and lower bounds can be used in a branch and bound tree for fathoming the current node, i.e., if the Z^* denotes the optimum solution of the current node and it does not meet the upper bound or the lower bound, then it can be fathomed and we can continue from the next open node. According to the experimental results,

the mentioned upper bound and lower bound are very effective, i.e., they are very close to each other and this means that the branch and bound tree will be minimal enough to reach to the feasible point.

3.4. Branching strategy

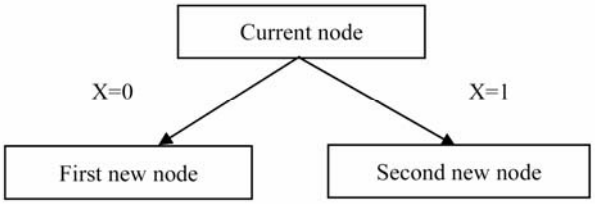
This strategy defines which node should be selected among the list of open nodes. This strategy has great influence on branch and bound efficiency. Let N denote the total number of variables of the corresponding problem VMTL1 in the current node which are not feasible according to constraints (8) in model VMTL.

Furthermore, suppose that L denotes the total number of integer variables corresponding to the vertex labels in the current node which generate different values for vertex labels. In fact, N is a degree of infeasibility of the current node and L shows that how many integer values of the set $\{1, 2, ..., m + n\}$ are generated by the current node. If N is small and L is close to $m + n$, then the current node is very good to be selected. So in the suggested algorithm, the "Jumptracking Strategy" is chosen as "Branching Strategy", i.e., a node from the active list (open nodes) with the maximum value of L is chosen to branch on. If there is a tie up, then a node with minimum value of N is selected. Again if there is a tie up, then a node which has the smallest sum of fractional part is chosen and finally if there is a tie up again, then a node is selected randomly.

3.5. Separation rule

Separation rule means the selection of a variable in the selected node to separate on. Assume to the "Jumptracking Strategy", node j is selected. According to the experimental research among more than 250 samples of different types of graphs, it was shown that the potential effect of distinct vertex labels is more powerful than that of distinct edge labels in a particular graph. Therefore, in "Separation Rule" of the Branching Method, a variable corresponding to a vertex has the priority to the other variables. Furthermore, when the "Branching Method" continues, many branches on the same variable will be generated in different parts of the branching tree. If a variable is chosen many times, then probably separation on this variable will not lead us to a feasible solution. Thus in "Separation Rule" of the algorithm, at first, variable in the selected node is chosen according to the priority. If there is a tie up, then a variable which has been selected less than the others is chosen, again if there is a tie up, then a variable with minimum fractional part is chosen and finally if there is a tie up again, then a variable is selected randomly.

Suppose variable x has been chosen from the current node, now we can separate the current node into two new nodes. Each new node is a subproblem of VMTL1 with two new constraints:



Finally, the branch and bound algorithm for VMTL problem is as follows:

Step 0. Initializing

Suppose that a graph G with n vertices and m edges is given and we want to know whether or not the graph G has VMTL and if it has VMTL, then we want to know how to label the vertices and edges. Now, consider the problem VMTL1. A node in a branching tree is active if its corresponding problem has not been either solved or subdivided yet. Let A denote the current list of active nodes. Compute the upper bound the lower bound.

Step 1. Branching

If A is empty, then stop, G has VMTL. Otherwise, select a node j from the active list A, according to "Jumptracking Strategy" which was described in 3.4. If the objective function of the current solution does not lie between the upper bound and the lower bound, then fathom the current node and select the next node. If the current solution satisfies constraints (8) in VMTL problem, then graph G has VMTL and terminate the algorithm, otherwise, go Step 2.

Step 2. Selecting

Separate the current node into two subproblems according to "Separation Rule" described before. In each new node, solve its corresponding problem. Add new subproblems to A and go to Step 1.

4. Computational Results

In this section, the computational results are summarized. The branch and bound algorithm coded with Visual Basic Language and the corresponding relaxation problems in Step 1 run by OSL V.3.0 software. All computations were run on a

Pentium IV, CPU 5.3 GHz, 500 GB H.D. with 2 GB of RAM. All graphs have been generated randomly by "Naughty V. 2.2" software. In the following table, the computational results of the algorithm have been shown among different classes of graphs. In this table, the first column defines the graph type, the second column is the number of edges, the third column is the number of vertices in the graph, the fourth column is the number of variables in problem VMTL1, the fifth column is the number of constraints in problem VMTL1, the sixth column specifies whether the graph has VMTL or not and the last column is the CPU average time of computations among all samples of that class.

Table 2. The results of the algorithm for different classes of graphs

Graph type		m	n	Number of variables	Number of Constraints	VMTL?	Average time in seconds
Cycles	C15	15	15	931	105	Yes	23.26
	C20	20	20	1,641	140	Yes	68.46
	C25	25	25	2,551	175	Yes	452.45
	C30	30	30	3,661	210	Yes	886.02
	C35	35	35	4,971	245	Yes	1853.79
	C40	40	40	6,481	280	Yes	4205.33
	C45	45	45	8,191	315	Yes	9541.12
Snakes	P15	14	15	871	102	Yes	13.1
	P20	19	20	1,561	137	Yes	54.23
	P25	24	25	2,451	172	Yes	385.33
	P30	29	30	3,541	207	Yes	745.00
	P35	34	35	4,831	242	Yes	1643.89
	P40	39	40	6,321	277	Yes	3965.21
	P45	44	45	8,011	312	Yes	9002.27
Complete graphs	K10	45	10	3,081	175	Yes	645.21
	K15	105	15	14,521	375	No	16544.73
	K20	190	20	44,311	650	Yes	28747.45

Complete bipartite graphs	K5,5	25	10	1261	115	Yes	30.98
	K5,10	50	15	4291	210	No	1698.43
	K10,10	100	20	14521	380	Yes	15887.74
Wheels	W10	20	11	993	104	Yes	29.31
	W15	30	16	2163	154	No	311.08
	W20	40	21	3783	204	No	855.67
Generalized Peterson graphs	P(5,2)	15	10	651	85	Yes	14.25
	P(8,4)	20	16	1333	124	Yes	76.20
	P(10,5)	25	20	2071	155	Yes	1034.53
Helms	H10	30	21	2653	174	No	557.32
	H15	45	31	5853	259	No	3742.80
	H20	60	41	10303	344	No	4907.44
Product graphs	K4xP10	46	20	4423	218	Yes	17855.32

For comparison purposes, we could not find such an approach for solving VMTL in the literature, but for showing the efficiency of the proposed algorithm, we refer to the results presented by Redl [5]. He has developed two different approaches for the graceful labeling problem which is almost similar to VMTL in terms of the objective function and some similarities in variables and constraints definition. In the first approach, he developed an integer programming model and in the second one, he applied a constraint programming technique. In the implementation of these two methods, three classes of graphs were tested at most: generalized Peterson graphs, product graphs of the form $K_4 \times P_n$ and double cones. According to his results, the best one is the constraint programming approach and the largest graph tested was P(10,5). The CPU time in seconds reported to solve P(10,5) was 10481,40 while total constraints were 50 and total variables were 45 in his approach. As we may see, this time is 1034.53 seconds in the proposed algorithm where total variables are 2071 and total constraints are 155 which shows that the proposed algorithm is around 10 times faster with more constraints and variables.

For the second comparison, we may see the results reported by Eshghi and Azimi [6], where they used a branch and bound algorithm for the graceful labeling problem. They reported 2499.68 seconds for solving P(10,5), where their model had 810 variables.

Therefore, in terms of speed and accuracy, the proposed algorithm is strong enough to be compared with the other works even in large-scale problems.

5. Conclusions

Despite the large number of papers in graph labeling, there are a few researches about developing an algorithm for finding the graph labels due to specified labeling. In this paper, at first, we have developed a 0-1 programming model for vertex-magic total labeling of a given graph, then a branch and bound algorithm was developed to solve the model, i.e., the result of the algorithm specifies the labels of vertices and edges based on vertex-magic total labeling. Finally, the algorithm which is the exact one has been tested among a large number of randomly generated graphs in different classes. The computational results show that the algorithm is powerful enough to solve the vertex-magic total labeling in large size graphs. However, this paper can help the researchers who are interested in the graph labeling problems but also it provides a useful tool for those who are interested in applications.

References

[1] A. Rosa, On certain valuation of the vertices of a graph, Proceedings of the International Symposium in Theory of Graphs, Gordon and Breach, New York, Dunod, Paris, 1967, pp. 349-355.

[2] R. L. Graham and N. J. A. Sloane, On additive bases and harmonious graphs, SIAM J. Algebraic Discrete Methods 1(4) (1980), 382-404.

[3] J. A. Gallian, A dynamic survey of graph labeling, Electron. J. Combin. 5 (1998), Dynamic Survey 6, pp. 43 (electronic).

[4] G. S. Bloom and S. W. Golomb, Application of numbered undirected graphs, Proceedings of the IEEE 65(4) (1985), 562-570.

[5] T. A. Redl, Graceful graphs and graceful labelings: two mathematical programming formulations and some other new results, Proceedings of the Thirty-fourth Southeastern International Conference on Combinatorics, Graph Theory and Computing, Congr. Numer. 164 (2003), 17-31.

[6] K. Eshghi and P. Azimi, Applications of mathematical programming in graceful labeling of graphs, J. Appl. Math. 1 (2004), 1-8.

[7] K. Eshghi and P. Azimi, An algorithm for finding a feasible solution of graph labeling problems, Util. Math. 72 (2007), 163-174.

[8] S. W. Golomb, How to Number a Graph, Graph Theory and Computing, R. C. Read, ed., Academic Press, New York, 1972, pp. 23-37.

[9] J. Sedlàček, Problem 27, Theory of Graphs and its Applications, Proc. Symposium Smolenice, June 1963, 163-167.

[10] B. M. Stewart, Magic graphs, Canad. J. Math. 18 (1966), 1031-1059.

[11] B. M. Stewart, Supermagic complete graphs, Canad. J. Math. 19 (1967), 427-438.

[12] J. A. MacDougall, M. Miller, Slamin and W. D. Wallis, Vertex magic total labelings of graphs, Proceedings Australasian Workshop Combin. Algorithm, 1999, pp. 222-229.

[13] Y. Lin and M. Miller, Vertex-magic total labelings of complete graphs, Bull. Inst. Combin. Appl. 33 (2001), 68-76.

[14] I. D. Gray, J. MacDougall and W. D. Wallis, On vertex-magic total labeling of complete graphs, Bull. Inst. Combin. Appl. 38 (2003), 42-44.

[15] D. McQuillan, A technique for constructing magic labelings of 2-regular graphs, preprint.

[16] N. C. K. Philips, R. S. Rees and W. D. Wallis, Personal Communication.

[17] J. A. MacDougall, M. Miller and W. D. Wallis, Vertex-magic total labelings of wheels and related graphs, Util. Math. 62 (2002), 175-183.

[18] R. M. Figueroa-Centeno, R. Ichishima and F. A. Muntaner-Batle, The place of super edge-magic labelings among other classes of labelings, 17th British Combinatorial Conference (Canterbury, 1999), Discrete Math. 231(1-3) (2001), 153-168.

[19] R. Figueroa-Centeno, R. Ichishima and F. Muntaner-Batle, On super edge-magic graphs, Ars Combin. 64 (2002), 81-95

[20] D. McQuillan and J. McQuialln, Magic labelings of triangles, Discrete Math., to appear.

[21] D. McQuialln and K. Smith, Vertex-magic total labeling of odd complete graphs. Discrete Math. 305(1-3) (2005), 240-249.

[22] M. Bača, M. Miller and Slamin, Vertex-magic total labeling of generalized Petersen graphs, 11th Australasian Workshop on Combinatorial Algorithms (Hunter Valley, 2000), Int. J. Comput. Math. 79(12) (2002), 1259-1263.

[23] Slamin and M. Miller, On two conjectures concerning vertex-magic total labelings of generalized Petersen graphs, Bull. Inst. Combin. Appl. 32 (2001), 9-16.

[24] A. Meissner and K. Zwierzyński, Vertex-magic total labeling of a graph by distributed constraint solving in the Mozart system, Parallel Processing and Applied Mathematics, Lecture Notes in Computer Science, 3911, Springer, Berlin, Heidelberg, 2006.

[25] W. D. Wallis, Magic Graphs, Birkhäuser, Inc., Boston, 2001.