New Formulations for the Minimum Branch Vertices Problem

Temel Öncan

Abstract—**In this work, we address a variant of the well-known spanning tree problem which consists of finding a spanning tree such that the number of branch vertices, namely vertices with degrees of at least three, is minimum (MBV). First, we introduce novel Mixed Integer Linear Programming (MILP) formulations for the MBV by improving the formulations from the literature. Then, we perform extensive computational experiments on standard test sets, to analyze their performance. According to our computational experiments, we have observed that the Linear Programming relaxations of the new MILP formulations yield drastically improved lower bounds than the ones output by earlier MBV formulations.**

Index Terms—**spanning trees, branch vertices, mixed integer linear programming**

I. Introduction

MANY combinatorial optimization problems deal with constructing *spanning trees* which are optimal with respect to some conditions. For a discussion of various real-life applications of the spanning trees arising in transportation, telecommunication, energy distribution, irrigation, data storage and cluster analysis we refer to Ahuja, Magnanti and Orlin [1], Capone, Corti, Gianoli, and Sanso [3] and, Ozeki and Yamashita [10].

Let $G=(V,E)$ be a connected undirected graph with a set of vertices $V=\{1,...,n\}$ and a set of edges $E=\{e_1,...,e_m\}$ where n and m stand for the number of vertices and edges, respectively. A spanning tree is a subset of E such that no cycle is constructed and the set of selected edges spans all nodes in V. In this paper, we address the spanning tree problem with a minimum number of branch vertices, i.e. vertices with degrees of at least three, (MBV), which involves finding a spanning tree of G such that the number of branch vertices is minimum. The MBV was first introduced by Gargano, Hell, Stacho and Vaccaro [7] in their seminal work whereby they showed it to be NP-complete. A closely related problem to the MBV is the spanning tree problem with minimum degree sum of branch vertices (MDS) which has been first launched by Cerulli, Gentili and Iosa [5]. The authors have suggested Single Commodity Flow (SCF) formulations, and they have introduced three construction heuristics, for both the MBV and MDS. Later on, Sundar, Singh and Rossi [13] suggested efficient construction heuristics and hybrid Ant Colony Optimization (ACO) algorithms for both the MBV and MDS. Recenty, Cerrone, Cerulli and Raiconi [4] have devised a memetic algorithm for three degree-dependent spanning tree problems including the MBV. In another study, four mathematical programming formulations for the MBV and different relaxations of them, have been analyzed by Carrabs, Cerruli, Gaudioso and Gentili [2]. The authors suggested Lagrangean relaxation approaches and reported results with a subgradient method and ad-hoc finite ascent algorithm.

Real-life applications of the MBV arises prominently in Wavelength Division Multiplexing (WDM) technology, used in optical networks (Stern and Bala [13]). The WDM technology enables the fiber optic communication used for web browsing, video conferences, video on demand services, etc. (Gargano, Hell, Stacho and Vaccaro [7]). Fiber optic communication encodes information into light waves and beams through an optical fiber. Since the wavelength of light determines its characteristic, each wavelength carries its own independent data-traffic. In WDM systems, several different wavelengths are combined and simultaneously transmitted over a single fiber. A multiplexer is used to join signals at the transmitter (i.e. source vertex) whereas a de-multiplexer, at the receiver (i.e. destination vertex) splits them apart. Furthermore, WDM technology also permits multi-casting in an optical network by using sophisticated switches which make copies of optical signals by splitting light, and hence, they transmit information from a single source vertex to multiple destination vertices. The multi-casting technique enables the distribution of copied data packets to multiple users via sophisticated light-splitting switches, located on the branch vertices of the optical network. Consequently, it is of great interest to design optical networks which enable multi-casting with a minimum number of light-splitting switches and with a minimum sum of branch vertex degrees, taking into account their costs.

Manuscript received June 11, 2014. This work was supported by the Turkish Scientific and Technological Research Council Research Grants No: 107M462 and 109M139, and Galatasaray University Scientific Research Project Grant No: 14.402.003.

T. Öncan is with the Department of Industrial Engineering, Galatasaray Üniversitesi, Ortaköy,İstanbul, 34357, TÜRKİYE (phone: +90.212 227 44 80 fax: +90 212 259 55 57 e-mail: ytoncan@gsu.edu.tr).

The motivation of this study is to suggest strong and novel Mixed Integer Linear Programming (MILP) formulations for the MBV which is known to be NP-Complete. We believe our attempts in devising novel MILP formulations will be helpful to develop efficient exact solution procedures for the MBV, as well.

The remainder of this work is organized as follows. In the next section we present some of the existing MILP formulations for the MBV from the literature. In Section III, we further elucidate on our novel MILP formulations. Then, in Section IV, we report the results of our extensive computational experiments. Finally, we conclude with Section V.

II. MILP Formulations for the MBV

In this section we present three existing MILP formulations for the MBV suggested by Cerrone, Cerulli and Raiconi [4].

The proposed formulations are defined on a directed graph *G'=(V,A)* obtained from *G=(V,E)* where *A={(i,j):i,j* $\in$ *V; i* $\neq$ *j}* is the arc set. In *G'=(V,A),* each undirected edge $e_p \in E$, which corresponds to edge *{i,j}*, is replaced by two directed arcs *(i,j)* and *(j,i)* where $i \neq j$. Furthermore, we assume vertex $r \in V$ represents the root vertex. $\delta(i)$ denotes the degree of vertex *i* and |*V*| stands for the cardinality of the vertex set *V*.

For all formulations, we define the following decision variables. Binary variables x_{ij} equal to *1* if there exists an arc *(i,j)* from vertex *i* to vertex *j*. Binary variables y_i equal to *1* if vertex *i* is a branch vertex. The proposed MILP formulations assume that a feasible solution consists of a subgraph of *G'=(V,A)* that spans all vertices in *V*, i.e. connected and acyclic, and has exactly one incoming arc to each vertex except the root vertex. Namely, a feasible solution is represented with an arborescence originating at the root vertex *r* such that there exists exactly one directed path from the root vertex *r* to every other non-root vertex $i \in V \backslash r$. Now we will present three MBV formulations from the literature.

A. Desrochers - Laporte Formulation

The first formulation which we will present in this section employs the subtour elimination constraints by Desrochers and Laporte [6] which were originally proposed for the Asymmetric Travelling Salesman Problem (ATSP) by lifting the constraints by Miller, Tucker and Zemlin [9]. We refer to Gutin and Punnen [8] for an overview of the ATSP formulations. In the Desrochers-Laporte (DL) formulation for the MBV, in addition to binary variables x_{ij} and y_i, continuous decision variables u_i are also used to calculate the visit order of vertex *i* from the root vertex *r*. The DL formulation for the MBV is the following.

$$DL: \min Z = \sum_{i=1}^{n} y_i \qquad (1)$$

s.t

$$\sum_{\substack{j \neq r \\ (i,j)\in A}} x_{ij} = 1 \qquad j \in V \backslash r \qquad (2)$$

$$\sum_{i \in V} \sum_{\substack{j \neq r \\ (i,j)\in A}} x_{ij} = |V| - 1 \qquad (3)$$

$$\sum_{\substack{i \neq r \\ (i,j)\in A}}^{n} x_{ji} + \sum_{\substack{i \neq r \\ (i,j)\in A}}^{n} x_{ij} \leq \delta(j) y_j + 2 \qquad j \in V \backslash r \qquad (4)$$

$$(|V|-2)x_{ji} + |V|x_{ij} + u_i - u_j \leq (|V|-1) \quad (i,j)\in A \quad (5)$$

$$1 \leq u_j \leq (|V|-1) \qquad j \in V \backslash r \qquad (6)$$

$$u_r = 0 \qquad (7)$$

$$x_{ij} \in \{0,1\} \qquad (i,j)\in A \qquad (8)$$

$$y_j \in \{0,1\} \qquad j \in V \qquad (9)$$

Here, objective function (1) calculates the number of branch vertices. Constraints (2) state the each vertex, except the root vertex, must have exactly one incoming arc. Constraints (3) guarantee that exactly |*V*|*-1* arcs are chosen to construct a spanning tree. Namely, constraints (2) and constraints (3) stand for the assignment constraints. Constraints (4) ensure that a vertex *j* is a branch vertex when the total number of outgoing and incoming arcs are larger than three. Constraints (5) are the subtour elimination constraints originally proposed for the ATSP by Desrochers and Laporte [6]. These constraints impose that when $x_{ij}=1$ then $u_i \leq u_j+1$ holds and when $x_{ji}=1$ then $u_i -1 \leq u_j$ is satisfied. Namely, constraints (5) restrict that $u_j=u_i+1$ holds when there exists an outgoing arc from vertex *i* to vertex *j*. Constraints (6) and constraints (7) impose lower and upper bounds on visit order variables u_i. Finally, constraints (8) and constraints (9) stand for the integrality restriction on the decision variables x_{ij} and y_j, respectively.

B. Single-Commodity Flow Formulation

The single-commodity flow (SCF) formulation employs the flow variables v_{ij} which stands for the amount of flow passing through arc *(i,j)*.

$$\text{SCF}: (1)\text{-}(4), (8), (9)$$

$$\sum_{(r,i)\in A} v_{ri} - \sum_{(i,r)\in A} v_{ir} = |V| - 1 \qquad (10)$$

$$\sum_{(j,i)\in A} v_{ji} - \sum_{(i,j)\in A} v_{ij} = -1 \qquad j \in V \backslash r \qquad (11)$$

$$x_{ij} \leq v_{ij} \leq (|V|-1) x_{ij} \qquad (i,j)\in A \qquad (12)$$

Constraints (10) make sure |*V*|*-1* units of commodities leave the root vertex. Constraints (11) enforce that exactly one unit of commodity arrives into each vertex except the root. Constraints (12) impose lower and upper bounds on the flow quantity for each arc in the spanning tree.

C. Multi-Commodity Flow Formulation

The multi-commodity flow (MCF) formulation uses the flow variables w_{ij}^k equal to *1* if and only a commodity of type *k* flows through arc *(i,j)*.

$$\text{MCF: (1) - (4), (8), (9)}$$

$$\sum_{(j,i)\in A} w_{ji}^k - \sum_{(i,j)\in A} w_{ij}^k = 0 \qquad j,k \in V \setminus r;\ j \neq k \qquad (13)$$

$$\sum_{(r,j)\in A} w_{rj}^k - \sum_{(j,r)\in A} w_{jr}^k = 1 \qquad k \in V \setminus r \qquad (14)$$

$$\sum_{(k,j)\in A} w_{kj}^k - \sum_{(j,k)\in A} w_{jk}^k = -1 \qquad k \in V \setminus r \qquad (15)$$

$$0 \le w_{ij}^k \le x_{ij} \qquad k \in V;(i,j) \in V \qquad (16)$$

Constraints (13) guarantee that for each vertex *j*, such that *j≠k*, when a commodity of type *k* enters into vertex *j* then it must leave vertex *j*. Constraints (14) make sure that there must be exactly one commodity of type *k* which must leave the root vertex. Constraints (15) ensure that there is exactly one incoming commodity of type *k* which is entered into vertex *k*. Constraints (16) are for the lower and upper restrictions on the flow variables w_{ij}^k.

III. New Formulations for the MBV

In this section we will present new formulations for the MBV. The new formulations are obtained with the addition of constraints (17)-(20) to the formulations DL, SCF and MCF and the new ones will be named as DL+, SCF+ and MCF+, respectively. According to our computational experiments, which will be presented in the next section, DL+, SCF+ and MCF+ formulations are much stronger than DL, SCF and MCF, respectively.

$$x_{ji} + x_{ik} \le 1 + y_i \qquad i \in V \setminus r \qquad (17)$$

$$x_{ri} + x_{rj} + x_{rk} \le y_r + 2 \qquad i,j,k \in V \setminus r \qquad (18)$$

$$2y_j \le \sum_{\substack{j\neq r \\ (j,i)\in A}} x_{ji} \le (\delta(j) - 2)y_j + 1 \qquad j \in V \setminus r \qquad (19)$$

$$1 + 2y_r \le \sum_{\substack{j\neq r \\ (r,j)\in A}} x_{rj} \le (\delta(r) - 2)y_r + 2 \qquad (20)$$

Now, we will show that constraints (17) -(20) are valid inequalities for the MBV. First of all, notice that constraints (17) and constraints (18) hold by definition. Recall that a branch vertex *i* has degree of at least three. Keeping in mind that all vertices but the root vertex have one incoming arc then each branch vertex has at least two incoming arcs. Moreover, in case the root vertex is a branch vertex then it has at least three outgoing arcs. Therefore, we can state that constraints are valid by definition.

Proposition 1. *Constraints (19) and (20) are valid inequalities for the MBV*

Proof. The left part of the constraints (19) hold by the definition of the x_{ij} and y_j variables. In case, vertex *j* is a branch vertex then there exist at least two outgoing arcs from the vertex *j*. For the other case, there is either one or none outgoing arc from vertex *j*. The right part of the constraints (19) hold also by the definition. Whenever vertex *j* is a branch vertex then there at most *δ(j)-1* outgoing arcs. Furthermore, when vertex *j* is not a branch vertex then there is at most one outgoing arc from the vertex *j*.

Similarly, for the left part of constraints (20), when the root vertex is a branch vertex then, there must be at least three outgoing arcs. However, when the root vertex is not a branch vertex then there must be at least one outgoing arc. Considering the right part of constraints (20), when the root vertex is a branch vertex then there are at most *δ(r)* outgoing arcs. On the other hand, when the root vertex is not a branch vertex then constraints (20) state that there are at most two outgoing arcs from the root vertex. ■

Proposition 2. *Constraints (19) dominate constraints (4)*

Proof. Considering constraints (2), we can rewrite constraints (4) as follows:

$$\sum_{\substack{i\neq r \\ (i,j)\in A}} x_{ji} \le \delta(j)y_j + 1 \qquad j \in V \setminus r \qquad (21)$$

On the other hand, the right hand side of constraints (19) can be rewritten as

$$\sum_{\substack{i\neq r \\ (i,j)\in A}} x_{ji} \le \delta(j)y_j + 1 - 2y_j \qquad j \in V \setminus r \qquad (22)$$

Note that since $0 \le y_j \le 1$ holds for $j \in V$, the right hand side of constraints (22) are tighter than the one of constraints (21), which completes the proof. ■

In summary, we suggest four new formulations: DL+, SCF+ and MCF+ which are obtained with the addition of constraints (17)-(20) to (1)-(9); (1)-(4),(8),(9),(10)-(12) and (1)-(4),(8),(9),(13)-(16), respectively.

IV. Computational Experiments

In this section, we present our computational experiments to expose the strength of the constraints (17)-(20) when used with DL, SCF and MCF formulations for the MBV. That is to say, we introduce an experimental comparative analysis of DL, SCF, MCF, DL+, SCF+ and MCF+ formulations. In the next discussion the test instances will be presented. Finally, the computational results will be reported.

A. Test Bed

To perform our experiments, we consider the test instances generated by Carrabs, Cerulli, Gaudioso and Gentili [2]. Broadly speaking we have two classes of test instances, the first class consists of *400* instances with *n* between *20* and *500*, the second class includes *125* instances with *n* between *500* and *1000*. Consequently, we have *525* test instances in total. All the test instances generated by Carrabs, Cerulli, Gaudioso and Gentili [2] consist of sparse graphs where the number of edges are fixed according to the following equation

$$m = \lfloor (n-1) + i \times 1.5 \times \lfloor \sqrt{n} \rfloor \rfloor \quad (23)$$

with *i=1,2,3,4,5*.

B. Computational Results

We now present the details of our computational experiments. All computations were performed on a Dell Server PE2900 with two 3.16 GHz Quad Core Processors and 16 GB RAM operating within Microsoft Windows Server 2003 environment. Cplex 11 with default options is used to solve the MILP and Linear Programming (LP) problems.

To better expose the strength of the proposed formulations we report some results on the empirical quality of the LP relaxation bounds of the DL, SCF, MCF, DL+, SCF+ and MCF+ formulations which are obtained by replacing constraints (8) and (9) with

$$0 \le x_{ij} \le 1 \qquad (i,j) \in A \qquad (24)$$

$$0 \le y_j \le 1 \qquad j \in V \qquad (25)$$

respectively. The LP relaxations of DL, SCF, MCF, DL+, SCF+ and MCF+ formulations will be denoted as LP-DL, LP-SCF, LP-MCF, LP-DL+, LP-SCF+ and LP-MCF+, respectively. Furthermore we have also conducted computational experiments with the Integer Programming (IP) relaxation of the DL+ and SCF+, by only replacing constraint (8) with constraints (24), and keeping all the remaining constraints the same. In the sequel, the IP relaxations of the DL+ and SCF+ will be referred to as IPR-DL+ and IPR-SCF+, respectively. Furthermore, note that, for the sake of a fair comparison of formulations, we have selected vertex *1* as the source vertex in all instances.

We have employed the following formulae to measure the relative deviations from the optimum or best known solutions

$$100 \times \left(\frac{z_{IP} - z_{LP}}{z_{IP}} \right) \qquad (26)$$

where z_{IP} is the optimum or best known solution value and z_{LP} is the lower bound obtained by solving the LP relaxation of the models. The optimum solutions of Class I instances are obtained by solving DL+ formulation via Cplex 11 MILP solver with default options. On the other hand, for Class II instances, no optimum solution values are reported in the literature. Hence, we have again tried to employ DL+ formulations via Cplex 11 MILP solver with a 3 hours CPU time limit. However, we have been able to reach the optimality only in only 99 out of 125 cases within the 3hours CPU time limit. For the rest of the Class II instances, i.e. 26 instances, we consider the best feasible solutions output by Cplex 11 MILP solver.

TABLE I
LP RELAXATION OF SCF AND DL ON CLASS I INSTANCES

	LP-SCF		LP-DL	
$\lvert V\rvert$	**%Dev**	**CPU**	**%Dev**	**CPU**
20	35.12	0.00	30.23	0.00
40	74.00	0.01	58.99	0.00
60	78.12	0.01	61.95	0.01
80	76.16	0.01	59.06	0.01
100	73.69	0.01	58.10	0.01
120	71.65	0.02	57.90	0.01
140	71.47	0.02	57.36	0.01
160	71.74	0.03	57.26	0.02
180	69.99	0.03	56.57	0.02
200	69.64	0.03	56.25	0.02
250	68.36	0.04	55.25	0.02
300	68.09	0.05	55.52	0.03
350	68.00	0.05	55.40	0.04
400	67.33	0.05	54.64	0.02
450	67.59	0.07	55.23	0.03
500	66.89	0.07	54.99	0.04
Aver.	**68.62**	**0.03**	**55.29**	**0.02**

In Table I (Table II) we report the results obtained with the LP relaxation of the SCF and DL (SCF+ and DL+) formulations on Class I instances, while in Table III (Table IV) we give the results output by LP relaxation of the SCF and DL (SCF+ and DL+) formulations on Class II instances. Table V includes the results obtained with the LP relaxation of the MCF formulation on Class I instances. In Table VI and Table VII, we present the results obtained with the MILP relaxation of the DL+ and SCF+ formulations on Class I and Class II instances, respectively.

TABLE II
LP RELAXATION OF THE SCF+ AND DL + ON CLASS I INSTANCES

	LP-SCF+		LP-DL+	
$\lvert V\rvert$	**%Dev**	**CPU**	**%Dev**	**CPU**
20	10.02	0.00	10.02	0.00
40	28.96	0.01	29.10	0.01
60	17.17	0.02	18.15	0.01
80	12.94	0.03	13.25	0.01
100	9.45	0.03	9.77	0.01
120	8.91	0.04	9.24	0.01
140	7.76	0.06	7.91	0.02
160	6.72	0.07	6.99	0.03
180	6.68	0.08	6.84	0.03
200	5.40	0.10	5.66	0.03
250	4.72	0.11	4.85	0.03
300	3.71	0.15	3.77	0.05
350	5.33	0.21	5.46	0.06
400	3.88	0.21	3.95	0.05
450	4.63	0.27	4.72	0.06
500	4.01	0.29	4.12	0.07
Aver.	**8.77**	**0.11**	**8.99**	**0.03**

TABLE III
LP RELAXATION OF THE SCF AND DL ON CLASS II INSTANCES

	LP-SCF		LP-DL	
$\|V\|$	%Dev	CPU	%Dev	CPU
600	63.99	0.08	52.11	0.05
700	64.03	0.09	52.00	0.07
800	63.81	0.10	51.78	0.04
900	63.68	0.10	51.58	0.04
1000	63.76	0.12	51.66	0.04
Aver.	**63.86**	**0.10**	**51.83**	**0.05**

In all tables, the first columns stand for the size of the instances and the last rows give average values of the corresponding columns. The rows indicate the average results obtained with 25 test instances with the same size. In Table I and Table II, the second and third (fourth and fifth) columns stand for the percent deviations from the optimum solution and the CPU time in seconds obtained with the LP relaxation of the SCF (DL) formulation, respectively. However, In Table III and Table IV, the second and third (fourth and fifth) columns are for the percent deviations from the best known solutions and the CPU time in seconds obtained with the LP relaxation of the SCF+ (DL+) formulation, respectively.

TABLE IV
LP RELAXATION OF THE SCF+ AND DL+ ON CLASS II INSTANCES

	LP-SCF+		LP-DL+	
$\|V\|$	%Dev	CPU	%Dev	CPU
600	17.55	0.06	1.54	0.18
700	17.93	0.07	1.54	0.21
800	18.60	0.10	1.53	0.21
900	19.88	0.10	1.54	0.23
1000	19.84	0.10	1.60	0.26
Aver.	**18.76**	**0.09**	**1.55**	**0.22**

When we analyze Table I and Table II, we can conclude that there is no considerable difference between the LP relaxation of the SCF and MCF formulations. The average percent deviations (CPU times in seconds) reported for the SCF and DL formulations are 38.69 % and 32.13 % (0.7 and 0.03) for the SCF and DL formulations, respectively. Actually, we know that the subtour elimination constraints of the SCF and DL formulations are incomparable in terms of their solution quality. For an analytical comparison of several ATSP subtour elimination constraints we refer to Öncan, Altınel and Laporte [11]. Although, the SCF and DL are theoretically incomparable, when we consider the solutions reported in Table III and Table IV we can observe that the DL model outperforms SCF model in terms of accuracy at the expense of a slight increase in CPU time requirement. Note that, the average percent deviations (CPU times in seconds) reported in Table III and Table IV are 41.31 % and 26.69 % (0.09 secs. and 0.13secs.) for SCF and DL, respectively.

TABLE V
LP RELAXATION OF THE MCF ON CLASS I INSTANCES

	LP-MCF		LP-MCF+	
$\|V\|$	%Dev	CPU	%Dev	CPU
20	29.49	0.21	9.36	0.22
40	58.94	0.83	28.63	0.87
60	60.67	1.84	16.10	1.95
80	58.73	3.76	12.22	4.51
100	57.62	5.87	9.14	36.66
120	57.54	8.18	8.63	11.15
140	57.10	12.04	7.65	15.78
160	56.95	19.32	6.39	25.45
180	56.33	25.13	6.40	34.93
200	56.06	34.24	5.23	49.10
250	55.16	57.56	4.59	68.85
300	55.41	93.32	3.64	131.32
350	55.22	134.81	5.12	301.02
400	54.55	186.12	3.92	263.12
450	55.09	280.72	4.52	390.67
500	54.93	350.02	3.90	454.28
Aver.	**54.99**	**75.87**	**8.47**	**111.87**

When we compare Table I with Table II and, Table III with Table IV, we can clearly observe the dramatic improvements in accuracy obtained with the valid inequalities (17)-(20) presented in Section III. The average percent deviation obtained with the SCF (the DL) on Class I instances impressively reduces from 68.62 % to 8.77 % (from 55.29 % to 8.99 %) with a insignificant increase in average CPU time requirement from 0.03 secs. to 0.11 secs. (from 0.02 secs. to 0.03 secs.). On the other hand, for Class II instances, the decrease in the average percent deviation is from 68.86 % to 18.76 % (from 51.83 % to 1.55 %) with a slight change in average CPU time requirement from 0.1 secs. to 0.09 secs. (from 0.05 secs. to 0.22 secs.) for the SCF (DL) formulation.

The improvements in the accuracy of the LP relaxation bounds obtained with the DL formulations are quite promising. Especially for Class II instances average percent deviation of 1.55 % from the best known or optimal solution with an insignificant average CPU time requirement implies that the LP relaxation bounds obtained with the DL formulation can be efficiently employed within a Branch and Bound algorithm as a lower bounding procedure. However, this research area is beyond scope of this study. Considering both the accuracy and the CPU time, we can state that the DL formulation is the winner with an overall average percent deviation of 29.42 % and CPU time requirement of 0.08 secs. compared to the overall average values, i.e. 40.00 % and 0.083 secs., obtained with the SCF formulation.

We have also performed experiments with the MCF formulations. Unfortunately, we could only solve instances with sizes up to *n=500*. For Class II instances we will not report the results with MCF formulation due to extreme memory space requirement of CPLEX LP solver. In Table V, we present the computational experiments performed with the LP relaxation of the

MCF formulations on Class I instances. As it can be observed, the accuracy of the LP relaxation bound considerably improves when constraints (22)-(25) are added to the MCF formulation at the expense of an enormous increase in computational time requirement .

TABLE VI
MILP RELAXATION OF THE (DL+) & (SCF+) ON CLASS I INSTANCES

	IPR-DL+		IPR-SCF+	
$\|V\|$	%Dev	CPU	%Dev	CPU
20	0.76	0.00	0.76	0.01
40	2.72	0.01	2.76	0.02
60	6.04	0.03	6.28	0.05
80	8.96	0.05	9.24	0.18
100	13.08	0.04	13.28	0.16
120	17.16	0.11	17.52	0.27
140	20.60	0.18	20.88	0.85
160	24.80	0.24	25.04	0.87
180	28.72	0.34	29.08	2.17
200	32.28	0.21	32.60	1.46
250	44.32	0.59	44.60	3.59
300	57.04	0.93	57.36	5.97
350	67.92	2.95	68.44	22.16
400	81.52	2.81	81.80	15.87
450	92.92	4.62	93.32	22.88
500	106.16	4.13	106.68	23.49
Aver.	**37.81**	**1.08**	**38.10**	**6.25**

Considering all three formulations: DL+, SCF+ and MCF+, we can say that although MCF+ yields the tightest average lower bound values, DL+ is the best choice when we take into account both accuracy and efficiency. Note that, although the LP relaxation of the DL+ yields the worst average percent deviation from the optimum solution on Class I instances, DL+ yields an outstanding efficiency in computation time.

In Table VI and Table VII we report the results obtained with the IP relaxation of the DL+ and SCF+ formulations. Recall that, IPR-DL+ and IPR-SCF+ stand for the IP relaxation of the DL+ and SCF+ formulations, respectively. Among all instances in Class I, in 260 and 354 out of 400 cases we get the optimum solution value with IPR-DL+ and IPR-SCF+, respectively. These values are 86 and 121 out of 125 cases for the Class II instances, with IPR-DL+ and IPR-SCF+, respectively.

Consequently, we can say that DL+ and SCF+ formulations yield quite tight LP relaxation and IP relaxation lower bounds. Unfortunately, the IP relaxation of the MCF+ formulation requires drastic CPU time hence we could not report the results obtained.

TABLE VII
MILP RELAXATION OF THE (DL+) & (SCF+) ON CLASS II INSTANCES

	IPR-DL+		IPR-SCF+	
$\|V\|$	%Dev	CPU	%Dev	CPU
600	0.20	1.98	0.00	15.81
700	0.14	4.62	0.05	42.34
800	0.14	8.84	0.00	83.40
900	0.17	8.24	0.00	92.56
1000	0.17	14.60	0.03	233.68
Aver.	**0.16**	**7.65**	**0.02**	**93.56**

V.CONCLUSION

We have devised novel formulations for the MBV. According to our computational experiments, we may conclude that both the linear programming and mixed integer linear programming relaxations of the proposed formulations yield promising lower bounds. We should remark that the design of the branch and bound algorithms remains as further research. Finally, new formulations for other degree dependent spanning tree problems is also an open research avenue.

REFERENCES

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows: Theory, Algorithms and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[2] F. Carrabs, R. Cerulli, M. Gaudioso, and M. Gentili, “Lower and upper bounds for the spanning tree with minimum branch vertices,” *Computational Optimization and Applications*, vol. 56, pp. 405–438, 2009.

[3] A. Capone, D. Corti, L. Gianoli, and D. Sanso, “An optimization framework for the energy management of carrier ethernet networks with multiple spanning trees,” *Computer Networks*, vol. 56, pp. 3666–3681, 2012.

[4] C. Cerrone, R. Cerulli, and A. Raiconi, “Relations, models, and a memetic approach for three degree-dependent spanning tree problems,” *European Journal of Operational Research*, vol. 232, pp. 442–453, 2014.

[5] R. Cerulli, M. Gentili, and A. Iossa, “Bounded-degree spanning tree problems: models and new algorithms,” *Computational Optimization and Applications*, vol. 42, pp. 353–370, 2009.

[6] M. Desrochers, G. Laporte, “Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints,” *Operations Research Letters*, vol. 10, pp. 27–36, 1991.

[7] L. Gargano, P. Hell, L. Stacho, and U. Vaccaro, “Spanning trees with bounded number of branch vertices, *Lecture Notes in Computer Science*, vol. 2380, Springer Verlag, Berlin 2002.

[8] G. Gutin, A.P. Punnen, “The Traveling Salesman Problem and Its Variations, ” Kluwer, Dordrecht, 2002.

[9] C.E. Miller, A.W. Tucker, and R.A. Zemlin, “Integer programming formulations and traveling salesman problems,” *Journal of Association for Computing Machinery*, vol. 7, pp. 326–329, 1960.

[10] K. Ozeki, T. Yamashita, “Spanning trees: a survey, ” *Graphs and Combinatorics*, vol. 27, pp. 1–26, 2011.

[11] T. Öncan, İ.K. Altınel, and G. Laporte, “A comparative analysis of several asymmetric travelling salesman problem formulations,” *Computers and Operations Research*, vol. 36, pp. 637–654, 2009.

[12] T.E. Stern, K. Bala, “Multi-wave length Optical Networks: A Layered Approach, ” Prentice-Hall, NJ, 1999.

[13] S. Sundar, A. Singh, and A. Rossi, “New heuristics for two bounded-degree spanning tree problems,” *Information Sciences*, vol. 195, pp. 226–240, 2012.