Modelling and Solving the Perfect Edge Domination Problem

Vinicius L. do Forte ∗1, Min Chih Lin †2, Abilio Lucena ‡3, Nelson Maculan §3, Veronica A. Moyano ¶2, and Jayme L. Szwarcfiter ∥3,4

1Universidade Federal Rural do Rio de Janeiro
   Departamento de Matemática
   Seropédica, Brasil
   vlforte@ufrrj.br

2Universidad de Buenos Aires
   Instituto de Cálculo and Departamento de Computación
   Buenos Aires, Argentina
   oscarlin@dc.uba.ar, vmoyano@ic.fcen.uba.ar

3Programa de Engenharia de Sistemas e Computação
   Universidade Federal do Rio de Janeiro
   Rio de Janeiro, Brazil
   {abiliolucena, maculan, jayme}@cos.ufrj.br

4Instituto de Matemática e Estatística, Universidade do Estado do
   Rio de Janeiro
   Rio de Janeiro, Brasil
   June 2018

∗Vinicius leal do Forte was partially funded by CNPq.
†Min Chih Lin was partially funded by UBACyT Grant 20020120100058, and PICT ANPCyT Grants 2010-1970 and 2013-2205.
‡Abilio Lucena was partially funded by CNPq grant 307026/2013-2.
§Nelson Maculan was partially funded by CNPq.
¶Veronica A. Moyano was partially funded by UBACyT Grant 20020120100058, and PICT ANPCyT Grants 2010-1970 and 2013-2205
∥Jayme L. Szwarcfiter was partially funded by CNPq.
Abstract

A formulation is proposed for the Perfect Edge Domination Problem and some exact algorithms based on it are designed and tested. So far, perfect edge domination has been investigated mostly in computational complexity terms. Indeed, we could find no previous explicit mathematical formulation or exact algorithm for the problem. Furthermore, testing our algorithms also represented a challenge. Standard randomly generated graphs tend to contain a single perfect edge dominating solution, i.e., the trivial one, containing all edges in the graph. Accordingly, some quite elaborated procedures had to be devised to have access to more challenging instances. A total of 736 graphs were thus generated, all of them containing feasible solutions other than the trivial ones. Every graph giving rise to a weighted and a non-weighted instance, all instances solved to proven optimality by two of the algorithms tested.

Keywords: Perfect Edge Domination, Exact Algorithms, Instance Generation, Computational Results.

1 Introduction

Let $G = (V, E)$ be a simple undirected graph with a set of vertices $V$ and a set of edges $E$. Denote by $\delta(i) \subseteq E$ the set of edges incident to $i \in V$ and respectively by $N[e] = \delta(i) \cup \delta(j)$ and $N(e) = N[e] \setminus \{e\}$ the closed and the open edge neighborhoods of $e = \{i, j\} \in E$. A set $D \subseteq E$ is said to dominate all the edges of $N[D]$ and is called an Edge Dominating Set (EDS) if $D \cap N[e] \neq \emptyset$ holds for every $e \in E$, i.e., if $D$ dominates all the edges of $G$. The Minimum Edge Dominating Set Problem (MEDSP) is to find an EDS of $G$ with cardinality as small as possible.

Edge domination has also been investigated in the literature under more restricted forms than the one described above. Indeed, perfect and efficient edge domination have probably attracted even more interest than the general problem. Given an EDS $D$, perfect domination applies if $|N(e) \cap D| = 1$ holds for every $e \in E \setminus D$, i.e., if $e \in E \setminus D$ is dominated by a single edge of $D$. Domination is efficient if, in addition, every edge of $D$ is dominated just by itself. The Perfect Edge Domination Problem (PEDP) is to find a Perfect EDS (PEDS) of $G$ with cardinality as small as possible. The Efficient Edge Domination Problem (EEDP) is defined along similar lines.

MEDSP is known to be NP-complete for some time. Moreover, it still remains NP-complete for graphs of maximum degree at most 3 that are either bipartite or planar [27]. In [14] the hardness has been further proved for planar bipartite graphs, planar cubic graphs, line graphs and total graphs. Additionally, MEDSP was also shown to be NP-complete for regular bipartite graphs [11]. On the other hand, polynomial-time algorithms are known for trees [27], block graphs [10], series-parallel graphs [23], bipartite-permutation graphs [24], and co-triangulated graphs [24]. As for exact algorithms for general graphs, references [2, 25] investigate Integer Programming (IP) based approaches for
finding Maximum Maximal Matchings (M MM s), where a MMM is a restricted type of EDS (see [2, 25] for details).

The problem of deciding whether a given graph admits an efficient EDS (E EDS) is known to be NP-complete [12]. Furthermore, it remains so even for planar bipartite graphs of maximum degree 3 [4], k-regular graphs with k \geq 3 [8] and, as indicated in [9], for subcubic (C_3, \ldots, C_k, H_1, \ldots, H_k)-free bipartite graphs for any fixed k (where H_i is the graph obtained by subdividing i – 1 times the middle edge of an H graph). On the other hand, polynomial time algorithms do exist for some graph classes, such as chordal graphs [21], generalized series-parallel graphs [21] (for both weighted and non weighted variants), claw-free graphs [9], weighted claw-free graphs [18], long claw-free graphs [13], graphs with bounded clique-width [9], hole-free graphs [4], convex graphs [16], dually-chordal graphs [5], P_7-free graphs [6], P_8-free graphs [7], bipartite permutation graphs [22], AT-free graphs [5], interval-filament graphs [5], and weakly chordal graphs [5]. Additionally, various exact algorithms do exist for general graphs [20, 26]. In particular, the one proposed in [20] also allows the counting of the number of EEDSs. So far, no exact IP based algorithm has yet been proposed for EEDP.

Less is known about PEDs. It is NP-complete to determine if a given graph contains a PED of a given size [21] and that result also holds true for claw-free graphs of degree at most 3, bipartite graphs [21], k-regular graphs with k \geq 3, and bounded-degree graphs of large girth or bounded-degree F-free graphs, except when F is a set of disjoint paths (see [17], for all of these particular cases). Polynomial time algorithms are known for chordal graphs [21], circular-arc graphs [19], P_5-free graphs [17], cubic claw-free graphs and bounded-degree F-free graphs [17], where F is a set of disjoint paths. No exact algorithm, combinatorial or IP based, appears to exist for PEDP. Indeed, we could not even find in the literature any explicit mathematical formulation for the problem.

Suggesting a formulation for PEDP and designing and testing some accompanying exact algorithm for it are two of the contributions of this paper. An additional one, as we shall see, is to generate challenging graphs G = (V, E) that, apart from the trivial solution, E, also contain additional feasible solutions for the problem.

Closing this brief overview on edge domination, it should be mentioned that it finds practical applications in the design and analysis of communication networks, network routing and coding theory [27, 12, 26].

Finally, it should also be stressed that we distinguish between the weighted and the non weighted variants of the problem. The former is called here Weighted PEDP (WPEDP) while the latter is simply called PEDP.

This paper contains four additional sections. Section 2 introduces our PEDSP formulation and describes some exact algorithms for it. Section 3 then follows with some quite elaborated procedures to generate graphs that contain PEDSs other than trivial ones. Section 4 describes the computational experiments we carried out for our algorithms. Finally, Section 5 closes the paper with some conclusions and suggestions for future work.
2 Problem Formulation

Our WPEDP formulation associates variables $x \in \mathbb{R}_+^{|E|}$ with the edges of $G = (V, E)$ and enforces that $x_e = 1$ holds if $e \in E$ is selected as a dominating edge, with $x_e = 0$ applying otherwise. It is given by

$$
\min \left\{ \sum_{e \in E} c_e x_e : x \in \mathcal{R}_o \cap \mathbb{Z}^{|E|} \right\},
$$

where $\mathcal{R}_o$ is the polyhedral region defined by the intersection of inequalities

$$
\sum_{f \in N[e]} x_f \geq 1, \forall e \in E
$$

$$
\sum_{e \in N(a)} x_e + (1 - |N(a)|)x_a \leq 1, a \in E, |N(a)| \geq 2
$$

$$
0 \leq x_e \leq 1, \ e \in E.
$$

The formulation takes edge weights $\{c_e \in \mathbb{R} : e \in E\}$ and specializes to PEDP when $\{c_e = 1 : e \in E\}$ applies.

Inequalities (2) ensure that at least one edge is selected for every neighborhood $N[e], e \in E$, thus guaranteeing that (1) returns an EDS of $G$. In turn, as required for perfect edge domination, inequalities (3) enforce that any non dominating edge, i.e., any $a \in E$ with $x_a = 0$, must be dominated by no more than one of its neighbor vertices. These inequalities thus become redundant when $x_a = 1$ applies. However, when $x_a = 0$ holds, they impose the required dominance condition, in combination with (2), written for $a$. As previously remarked, formulation (1) appears to be the only one so far proposed for PEDP.

A Linear Programming (LP) relaxation for the formulation is defined as

$$
\min \left\{ \sum_{e \in E} c_e x_e : x \in \mathcal{R}_o \right\}
$$

and may be reinforced with inequalities

$$
\sum_{e \in F} x_e + (1 - |F|)x_a \leq 1, a \in E, F \subseteq N(a), |F| \geq 2,
$$

that contain (3) and are valid for PEDP. It should be pointed out that inequalities (6) subsume

$$
x_b + (1 - x_a) + x_c \leq 2, a = \{i, j\} \in E, b \in \delta(i) \setminus \{a\}, c \in \delta(j) \setminus \{a\}
$$

and

$$
\sum_{e \in F \setminus \{a\}} x_e + (2 - |F|)x_a \leq 1, i \in V, F \subseteq \delta(i), a \in F, |F| \geq 3,
$$

and that we have two basic reasons to highlight that fact. The first is that we came across inequalities (7) and (8) prior to reaching (6). The second and
most important one is that, out of our computational experiments, the best performing BC algorithm we obtained is one that reinforces LP relaxation (5) with inequalities in (7) and (8), and not with the additional inequalities in (6). Although stronger LP relaxation bounds are attained by restricting oneself to using only the inequalities in (6), the resulting algorithm did not pay off in CPU time and RAM memory terms. Quoting specific figures for that, the BC algorithm based on the latter bounds took, on average, 71% more CPU time than its counterpart, in spite of exploring, on average, 10% less enumeration tree nodes. Furthermore, under the CPU time limit imposed, it failed to solve 6 instances of our test set. That compares with only one for its counterpart. For that reason and also due to space limitation constraints, computational results in Section 4 are restricted to those obtained by IP solver CPLEX, version 12.6 [15], used as a stand alone algorithm, and the BC algorithm based on the reinforced formulation

\[ \min \{ \sum_{e \in E} \epsilon_e x_e : x \in \mathcal{R} \cap \mathbb{Z}^{\left| E \right|} \}, \quad (9) \]

where \( \mathcal{R} \) is defined as the intersection of (2)-(4), (7), and (8).

### 2.0.1 Reinforcing the LP Relaxation with Cutting Planes

Out of the inequalities that define \( \mathcal{R} \), the very first LP relaxation for our BC algorithm only uses (2)-(4) and (7). The number of inequalities in either (2) or (3) is precisely \( |E| \). On the other hand, \( O(|E| \cdot |V|^2) \) inequalities (7) do exist for general graphs. These, however, would reduce to as low as \( O(|E|) \), as it applies to the 3-regular graphs we will consider in Section 4. Finally, the exponentially many inequalities in (8) are separated on the fly by the algorithm and are only appended to LP relaxations when violated.

Assume that \( \overline{x} \) is an optimal solution to the LP relaxation problem in hand and that \( \mathcal{G} = (\overline{V}, \mathcal{E}) \) is its corresponding support graph, i.e., the subgraph of \( G \) defined by the edges \( e \in E \) with \( \overline{x}_e > 0 \). We use notation \( \overline{\delta}(i) \) to identify support graph edges incident to \( i \in \overline{V} \). Additionally, inequalities (8) are rewritten as

\[ \sum_{e \in F \backslash \{a\}} (x_e - x_a) \leq (1 - x_a), \quad i \in V, \quad F \subseteq \delta(i), \quad a \in F, \quad |F| \geq 3 \quad (10) \]

and for every \( i \in \overline{V} \) with \( |\overline{\delta}(i)| \geq 3 \) and for every \( a \in \overline{\delta}(i) \), a particular set \( F \) is defined. Namely, a set formed by \( a \) and all edges \( e \in \overline{\delta}(i) \backslash \{a\} \) with non negative \( (\overline{x}_e - \overline{x}_a) \) values. If \( |F| \geq 3 \) and \( \sum_{e \in F \backslash \{a\}} (\overline{x}_e - \overline{x}_a) > (1 - \overline{x}_a) \) holds, inequality (10) corresponding to the triplet \( i, a, F \) is violated and is used to reinforce the relaxation. Conversely, if \( |F| < 3 \) applies, one then checks, in decreasing order of \( (\overline{x}_e - \overline{x}_a) \) values, if there exists a subset \( F_e \subseteq \overline{\delta}(i) \backslash F \) with \( |F| + |F_e| = 3 \) and \( \sum_{e \in (F \cup F_e) \backslash \{a\}} (\overline{x}_e - \overline{x}_a) > (1 - \overline{x}_a) \). In case it does, \( \sum_{e \in F \cup F_e \backslash \{a\}} (x_e - x_a) \leq (1 - x_a) \) is then violated by \( \overline{x} \) and the inequality is used to reinforce the relaxation.
3 Test Instances

We initially tested our algorithms over general graphs $G = (V, E)$, randomly generated so as to enforce that a given pre-defined percentage density is attained. Graph connectivity was ensured by initializing the scheme with a randomly generated Hamiltonian path for $G$. Additional edges would then follow, until the desired graph density was reached. However, as pointed out before, after experimenting with literally hundreds of these graphs, they all contained no PEDS apart from the trivial one, $E$. That outcome prompted us to look for alternative generation schemes that would escape that pattern and produce more challenging instances.

PEDP is known to be NP-hard for some specific graph classes. Among them, bipartite graphs [21] and $K_3$-free 3-regular graphs [17], where $K_3$ stands for a complete graph on three vertices. We have thus devised generation schemes that produce particular graphs belonging to these two classes. Additionally, we also consider graphs from an additional class we call Efficient Edge Domination (EED) graphs. The reasoning behind that denomination is that these graphs, as we shall see, always contain a feasible EEDP solution. As such they also contain a feasible solution for PEDP, given that efficient edge domination implies perfect edge domination. Finally, as it will be indicated, that solution is a non trivial PEDP one.

EED graphs and the generation scheme we implemented for them will be described first, followed by our particular types of connected bipartite and $K_3$-free 3-regular graphs. Every graph we generate gives rise to a PEDP and a WPEDP instance, the latter instances with edge weights randomly drawn from the uniform distribution in the range $[1,1000]$.

3.1 EED graphs

In order to generate graphs that contain non trivial PEDSs, we rely on the alternative and frequently used description of an efficient EDS as a Dominating Induced Matching (DIM) (see [1], for instance).

An induced matching of a graph $G = (V, E)$ is a matching $D \subseteq E$ such that no two edges of $D$ are joined by an edge of $E \setminus D$. Additionally, $D$ is a DIM if every edge of $E \setminus D$ shares exactly one vertex with an edge of $D$. Accordingly, a DIM and an EEDS define a same graph theoretical structure and the problem of determining whether or not $G$ contains an EEDS is frequently cast in the literature in DIM terms. As indicate in [1], $G$ contains a DIM if there exists a partition of $V$ into vertex subsets $V_1$ and $V_2$ such that $V_1$ is an independent set and $V_2$ induces a matching of $G$. Furthermore, provided such a partition exists, a DIM is defined by the matching induced by $V_2$.

In addition to the conditions above, we also impose, for simplicity, that $n_2 = |V_2|$ is even, so that the resulting DIM is a perfect matching for the vertices of $V_2$. An EED graph, $G = (V, E)$, is thus defined as follows: (a) vertex set $V$ that partitions into two non empty subsets $V_1$ and $V_2$, (b) $n_2 = |V_2|$ is even, (c) edge set $E$ partitions into two disjoint subsets $E_1$ and $E_2$, (d) $E_1$ edges,
numbering $m_1 = |E_1|$, all have an end vertex in $V_1$ and the other in $V_2$, and (e) $E_2$ edges, numbering $m_2 = |E_2|$, all have both end vertices in $V_2$. Finally, we will only consider connected EED graphs since, otherwise, one could simply decompose them into connected components and solve a separate PEDP for every component.

### 3.1.1 A Generation Procedure for Connected EED Graphs

A description follows of the procedure we use to generate connected EED graphs. It takes $n = |V|$ as an input and then randomly selects positive values for $n_1$ and $n_2$, with $n_2$ even. In doing so the number of edges in $E_2$, i.e., $m_2 = \frac{|V|}{2}$, is automatically fixed and the edges of $E_2$ are generated by randomly selecting the pairs of vertices of $V_2$ to be matched (see Figures 1(a) and 1(b) for an example in which $n = 10$, $n_1 = 6$, $n_2 = 4$ and $m = 12$). Next, the number of edges of $G$, i.e., $m = |E|$ with $m = m_1 + m_2$, is fixed by randomly selecting a positive integer $m_1$ in the range $[2 \cdot \min\{n_1, n_2\} + |n_1 - n_2| - 1, n_1 \cdot n_2]$. At that stage, the generation of $E_1$ edges is initiated with an initial focus on enforcing connectivity for $G$. Accordingly, $2 \cdot \min\{n_1, n_2\}$ edges are generated in association with an (inclusion-wise) maximal simple alternating path, every edge of it with an end vertex in $V_1$ and the other in $V_2$. If $|n_1 - n_2| \geq 2$ holds, some leftover vertices in either $V_1$ or $V_2$, whatever applies, would not belong to the alternating path. In that case, additional $E_1$ edges are generated, one edge for every leftover vertex, all edges with a same end point in the opposite vertex set. See Figure 1(a) for the required 8 simple path edges for our example plus an additional edge, highlighted in dotted lines, associated with the single existing leftover vertex. Notice that the $E_1$ edges generated so far suffice to ensure that $G$ is connected. Finally the procedure terminates by generating the remaining edges of $E_1$, so that $|E_1| = m_1$ is attained. See Figure 1(b) for the three additional $E_1$ edges required by the example, all of them highlighted in dotted lines.

A total of 363 EED graphs were generated, with $|E|$ ranging from 30 to 300 and densities ranging from either 50% to 10% or 50% to 1%, depending on the values of $n_1$, $n_2$ and $m$.

![Figure 1: EED graph generation.](image-url)
3.2 Bipartite graphs

The (connectred) bipartite graphs we consider conform with those found within the EED graphs described above. More specifically, they are defined by the $E_1$ edges found within an EED graph. Accordingly, we partially follow the EED graph generation scheme to obtain them. The only difference between the two is that no edges with both end vertices in $V_2$ are now generated.

A total of 363 bipartite graphs were generated, with $|E|$ ranging from 30 to 300 and densities ranging from either 50% to 10% or 50% to 1%, depending on the values of $n_1$, $n_2$ and $m$.

3.3 Connected $K_3$-free 3-regular Graphs

We denote CR graphs the connected $K_3$-free 3-regular graphs $G = (V, E)$ considered in this subsection. Likewise any 3-regular graph, a CR graph must contain an even number of vertices, $n = |V|$. However, contrary to general 3-regular graphs, they decompose into three particular components. Among these, the most important is a certain type of Binary Tree (BT), to be called an OBT.

The other two are: (a) an additional edge incident to two edge degree 2 OBT vertices and (b) a simple cycle restricted to containing only all OBT leaves. Cycles conforming to (b) are required to satisfy an additional constraint, to be described later on, and will be called leaf-restricted cycles. A description of these three components follows and will serve as a description for a CR graph.

Any OBT must contain all vertices of $G$. Additionally, as it will be explained next, the OBT topology is tied to the topology of the minimal Complete BT (CBT) that contains it. A $p \geq 0$ levels CBT corresponding to a BT with $2^k$ vertices at every level $0 \leq k \leq p$. Accordingly, the standard graphical representation of a BT, in terms of father vertices and their left an right sons, thus applies to a CBT and will also be enforced here for OBTs.

For reasons that will become evident later on, OBTs are also required to have $p \geq 4$ levels, all of them complete up to level $p - 1$. That condition, in conjunction with $n$ even, implies that OBT level $p$ is incomplete and contains an odd number of vertices, $t$, with $1 \leq t \leq 2^p - 1$. Finally, these $t$ vertices are required to correspond exactly to the $t$ leftmost $p$-level vertices at a $p$ levels CBT. Accordingly, all definitions put together, an OBT, as we shall see, is uniquely defined by its corresponding number of vertices, $n$. For simplicity, assume that the vertices of $G$ are indexed so that: (a) vertices 1 and $n$ are respectively the OBT root and the right most vertex found at OBT level $p$, (b) vertex indices increase with OBT level and within a level, from left to right. A 30 vertices OBT is depicted in Figure 2, together with and additional edge connecting its two edge degree 2 vertices and a leaf-restricted cycle, both components highlighted in dotted lines.

From the definitions above, the following remarks would apply to an OBT: (a) $n$ is an OBT leaf and is the rightmost vertex found at level $p$, (b) $n$ has no brother and is the left son of $f$, a vertex located at level $p - 1$, (c) $f$ and the OBT root vertex, 1, are the only two edge degree 2 vertices found in the OBT.
and (d) remaining OBT vertices either have edge degree 1, as it applies to OBT leaves, or else, edge degree 3.

Figure 2: CR graph = OBT + single edge + leaf-restricted cycle.

Edge \{1, f\} ensures that vertices 1 and \(f\) have edge degrees 3, in \(G\), as required by a 3-regular graph such as \(G\). It also explains why an OBT is required to have \(p \geq 4\) levels. Notice that vertices 1 and \(f\) respectively belong to OBT levels 0 and \((p - 1) \geq 3\). They are therefore more than 3 levels apart, what guarantees that no \(K_3\) subgraph is thus induced by \{1, \(f\}\).

Let us now turn to leaf-constrained cycles. They ensure that OBT leaves have edge degrees 3, in \(G\). Furthermore, they must not contain edges between leaves of a same father, what prevents \(K_3\) subgraphs being formed. At this point, it only remains to be shown that a leaf-restricted cycle always exists for an OBT. We will show that through an example.

Under a conveniently chosen layout, Figure 3 displays the leaves for 16, 18, 20 and 22 vertices OBTs. In particular, 16 is the least possible number of vertices for an OBT. Examples of accompanying leaf-restricted cycles are also depicted in that figure, highlighted in dotted lines. Leaves are displayed side-by-side in increasing order of their indices, as if they all belonged to a same level. The 16 vertices OBT contains two \textit{type 1} leaves and 3 sets of \textit{type 2} leaves. A type 1 leaf defined as a son of \(f\) while a type 2 set is formed by exactly the two sons of a father other than \(f\). Corresponding type 1 and type 2 figures for the 18 vertices OBT are respectively 2 and 4. On the other hand, the 20 vertices OBT, that conforms with the 16 vertices OBT, contains one type 1 leaf and four type 2 sets. Corresponding figures for the 22 vertices OBT, that conforms with the 18 vertices OBT, are respectively 1 and 5. As one may then infer from the cycle patterns we present, leaf-restricted cycles would always exist for OBTs with \(n \geq 24\).

3.3.1 A CR Graph Generator

Our CR graph generator relies on leaf-restricted cycles similar to those displayed in Figure 3. It take \(n\) as an input, \(n \geq 16\), and firstly generates an \(n\) vertices OBT. In doing so, edge \{1, \(f\}\} is automatically defined. Additionally, depending
on the number of type 1 leaves involved, one out of two leaf-restricted cycle building procedures is activated. If a single type 1 leaf exists, the cycle would involve the following edges: $\{\{f + 2k, f + 2k + 1\} : k = 1, \ldots, l_2\}$, where $l_2$ is the number of type 2 brother pairs, $\{f + 1, f + 3\}$, $\{\{f + 2k, f + 2k + 3\} : k = 1, \ldots, l_2 - 1\}$, and $\{f + 1, f + 2l_2\}$. Otherwise, if two type 1 leaves exist, one would then have: $\{\{f + 2k - 1, f + 2k\} : k = 1, \ldots, l_2 + 1\}$, $\{f + 2, f + 4\}$, $\{\{f + 2k + 1, f + 2k + 4\} : k = 1, \ldots, l_2 - 1\}$, and $\{f + 1, f + 2l_2 + 1\}$. To introduce a degree of randomness into the procedure, a permutation of type 2 sets is randomly selected and the ordering it implies is then followed by the generation procedure. Namely, the positioning of type 1 leaves would be taken unaltered. However, type 2 sets would be taken in their permutation implied positions, and not in their true ones.

Ten CR graphs were generated, for $n \in \{20, 40, \ldots, 200\}$ and $|E|$ ranging from 30 to 300. Instances are identified by their corresponding parameters $\{|V|, d, |E|\}$, where $d$ stands for percentage graph density: $\{20, 15.79, 30\}$, $\{40, 7.69, 60\}$, $\{60, 5.08, 90\}$, $\{80, 3.8, 120\}$, $\{100, 3.03, 150\}$, $\{120, 2.52, 180\}$, $\{140, 2.16, 210\}$, $\{160, 1.89, 240\}$, $\{180, 1.68, 270\}$, and $\{200, 1.51, 300\}$.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure3.png}
\caption{Leaves and leaf-restricted cycles for \{16, 18, 20, 22\}-vertices OBTs.}
\end{figure}
4 Computational Experiments

Computational experiments are reported in this section for the BC algorithm described in Section 2, denoted BCA, and also for CPLEX IP solver, version 12.6 [15], operating as a stand alone algorithm and denoted CPLEX-BC, here. BCA was implemented in C, uses the LP module of CPLEX, and also relies on that solver for the management of the BC trees. However, it makes no use of the pre-processing, cutting plane, and heuristic modules CPLEX offers. BCA separates inequalities (8) at every node of the BC tree, under the procedure described in Subsection 2.0.1. An Intel Xeon X5675 based machine with 48 Gb of RAM memory and running at 3.07 GHz was used in the experiments. A CPU time limit of 7,200 seconds was imposed on every run and to simplify eventual future comparisons, the experiments were performed on a single CPU thread.

Results are presented by graph class and problem variant, i.e., PEDP or WPEDP. Furthermore, due to space limitation, we only show detailed results for some particular representative graphs for either variant. These are packed in two groups of five graphs, each. Each group highlighting one of the following BC performance parameters: (a) CPU time and (b) percentage LP relaxation gap, given by (optimal solution value - LP relaxation value)/(optimal solution value). In particular, for every different graph class, results are presented for the five PEDP and the five WPEDP graphs for which CPLEX-BC performed the best (resp. the worst). These results are accompanied by BCA results for the same set of instances. Additionally, some complementary information is also included. Namely, we replicate these experiments for the same graphs, but now tested for the alternative problem variant, PEDSP or WPEDP, whatever applies. In doing so one is able to compare, at least for extreme cases, the differences involved in solving the two problem variants over a same set of graphs.

For any of the tables that follow, a graph \( G = (V, E) \) is identified by its corresponding \(|E|\), density, \(d\), and \(|V|\). Algorithm statistics come next and indicate: optimal solution value, \(OPT\); number of BC nodes, \(NN\); CPU time in seconds, \(t(s)\); LP relaxation value at the root node of the enumeration tree, \(LB_0\); LP relaxation gap at the root node of the enumeration tree, \(GAP\), and number of cuts separated by the BC algorithm, \(NUC\). Following that, as complementary information, statistics are also presented for these same graphs, but solved for the alternative problem variant.

4.1 Results for EED graphs

Under the 7,200 seconds CPU time limit imposed, both algorithms obtained optimal PEDP and WPEDP solutions for all 363 EED graphs. The average CPU time taken by CPLEX-BC to solve a PEDP instance was 0.52 seconds while its average LP relaxation gap was 0%. Corresponding figures for BCA are respectively 0.823 and 0%. For WPEDP instances, the corresponding CPLEX-BC figures are 0.91 and 15.8% while those for BCA are 5.01 and 37.6%. From the average results obtained, CPLEX-BC outperforms BCA for EED graphs.

Tables 1, 2 and 3 show extreme CPU time and LP relaxation gaps for EED
graph instances. The first table applies to PEDP instances while the other two are for WPEDP.

Naturally integral LP relaxation solutions were obtained for all PEDP instances, either by CPLEX-BC or BCA. Accordingly, no table is used to display these results. Differently from that, fractional LP relaxation solutions were obtained by CPLEX-BC for all but 8 WPEDP instances. The corresponding figure for BCA is 9.

From the average results obtained and also from an analysis of the extreme cases shown in the tables, PEDP instances appear to be much easier to solve than their WPEDP counterparts, at least as far as EED graphs are concerned.

| PEDP | WPEDP |
|------|-------|
| OPT  | NN t | LBk | NUC | GAP | OPT  | NN t | LBk | NUC | GAP |
| 200  | 33.68%| 42  | 5   | 1.027 | 5 | 0 | 0% | 0  | 2.491 | 3544.63 | 0 | 0.04% |
| 220  | 39.22%| 34  | 8   | 1.021 | 8 | 0 | 0% | 0  | 1.219 | 4415.57 | 0 | 1.02% |
| 240  | 34.14%| 38  | 6   | 1.099 | 6 | 0 | 0% | 0  | 2.124 | 3470.42 | 0 | 1.52% |
| 70   | 40.04%| 19  | 4   | 1.002 | 4 | 0 | 0% | 0  | 0.605 | 1610.91 | 0 | 7.15% |
| 260  | 19.61%| 52  | 8   | 1.001 | 8 | 0 | 0% | 0  | 2.612 | 4261.14 | 0 | 1.02% |

Five highest CPU times for CPLEX-BC

| BCA results for the same instances above |
|-----------------------------------------|
| 200  | 33.68%| 42  | 5   | 2.672 | 5 | 0 | 0% | 0  | 2.612 | 4261.14 | 0 | 1.02% |
| 220  | 39.22%| 34  | 8   | 1.167 | 8 | 0 | 0% | 0  | 0.407 | 1751.73 | 0 | 44.25% |
| 240  | 34.14%| 38  | 6   | 2.267 | 6 | 0 | 0% | 0  | 1.219 | 3470.42 | 0 | 1.52% |
| 70   | 40.04%| 19  | 4   | 0.369 | 4 | 0 | 0% | 0  | 0.605 | 1610.91 | 0 | 7.15% |
| 260  | 19.61%| 52  | 8   | 0.977 | 8 | 0 | 0% | 0  | 2.612 | 4261.14 | 0 | 1.02% |

Five lowest CPU times for CPLEX-BC

| BCA results for the same instances above |
|-----------------------------------------|
| 70   | 45.75%| 18  | 3   | 0.019 | 3 | 0 | 0% | 0  | 0.638 | 1193.17 | 0 | 10.82% |
| 120  | 24.19%| 32  | 4   | 0.021 | 4 | 0 | 0% | 0  | 0.743 | 1690.67 | 0 | 1.07% |
| 140  | 24.29%| 39  | 8   | 0.027 | 8 | 0 | 0% | 0  | 0.324 | 2303.84 | 0 | 1.59% |
| 150  | 34.48%| 30  | 6   | 0.027 | 6 | 0 | 0% | 0  | 0.957 | 2663.24 | 0 | 1.07% |
| 70   | 51.47%| 17  | 4   | 0.031 | 4 | 0 | 0% | 0  | 0.786 | 1751.73 | 0 | 19.53% |

Table 1: Extreme PEDP cases and WPEDP complementary information.
Table 2: Extreme WPEDP cases and PEDP complementary information.

| (E) | dens. (%) | | | WPEDP | | | PEDP | | |
|-----|-----------|-----|-----|------|-----|-----|------|-----|-----|
| | | OPT | N.N | $t(s)$ | LB | NUC | GAP | OPT | N.N | $t(s)$ | LB | NUC | GAP |
| Five highest | | 280 | 0.99% | 238 | 2974 | 0 | 0.586 | 204.85 | 0 | 93.11% | 4 | 0 | 0.937 | 4.00 | 0 | 0% |
| CPU times | for CPLEX-BC | | | | | | | | | | | | | |
| | | 140 | 2.95% | 98 | 1182 | 0 | 0.482 | 157.71 | 0 | 86.66% | 2 | 0 | 0.492 | 2.00 | 0 | 0% |
| | | | | 280 | 3.99% | 119 | 2558 | 0 | 0.854 | 342.20 | 0 | 86.62% | 4 | 0 | 0.045 | 4.00 | 0 | 0% |
| | | | | | | | | | | | | | | |
| | | 260 | 1.99% | 162 | 1759 | 0 | 0.593 | 254.47 | 0 | 85.53% | 3 | 0 | 0.772 | 3.00 | 0 | 0% |
| | | | | 300 | 4% | 123 | 3736 | 0 | 0.281 | 566.61 | 0 | 84.83% | 5 | 0 | 0.346 | 5.00 | 0 | 0% |
| Five lowest | | 220 | 0.99% | 211 | 4841 | 0 | 0.023 | 207.91 | 0 | 93.11% | 10 | 0 | 0.655 | 10.00 | 0 | 0% |
| gaps | for the same | | | | | | | | | | | | | |
| instances above | | 140 | 2.95% | 98 | 842 | 0 | 0.033 | 799.35 | 0 | 90.77% | 2 | 0 | 0.429 | 2.00 | 0 | 0% |
| | | | | 300 | 3.99% | 119 | 4996 | 0 | 0.063 | 4614.37 | 0 | 86.62% | 2 | 0 | 0.045 | 4.00 | 0 | 0% |
| | | | | | | | | | | | | | | |
| | | 170 | 1% | 215 | 6291 | 0 | 0.096 | 1725.17 | 0 | 93.11% | 11 | 0 | 1.211 | 11.00 | 0 | 0% |

Table 3: Extreme WPEDP cases and PEDP complementary information.

| (E) | dens. (%) | | | WPEDP | | | PEDP | | |
|-----|-----------|-----|-----|------|-----|-----|------|-----|-----|
| | | OPT | N.N | $t(s)$ | LB | NUC | GAP | OPT | N.N | $t(s)$ | LB | NUC | GAP |
| Five highest | | 280 | 0.99% | 238 | 2974 | 0 | 0.586 | 204.85 | 0 | 93.11% | 4 | 0 | 0.937 | 4.00 | 0 | 0% |
| CPU times | for CPLEX-BC | | | | | | | | | | | | | |
| | | 140 | 2.95% | 98 | 1182 | 0 | 0.482 | 157.71 | 0 | 86.66% | 2 | 0 | 0.492 | 2.00 | 0 | 0% |
| | | | | 280 | 3.99% | 119 | 2558 | 0 | 0.854 | 342.20 | 0 | 86.62% | 4 | 0 | 0.045 | 4.00 | 0 | 0% |
| | | | | | | | | | | | | | | |
| | | 260 | 1.99% | 162 | 1759 | 0 | 0.593 | 254.47 | 0 | 85.53% | 3 | 0 | 0.772 | 3.00 | 0 | 0% |
| | | | | 300 | 4% | 123 | 3736 | 0 | 0.281 | 566.61 | 0 | 84.83% | 5 | 0 | 0.346 | 5.00 | 0 | 0% |
| BCA results | | 220 | 0.99% | 211 | 4841 | 0 | 0.709 | 2521.84 | 0 | 93.11% | 10 | 0 | 0.331 | 10.00 | 0 | 0% |
| for the same | | 140 | 2.95% | 98 | 842 | 0 | 0.543 | 317.57 | 0 | 86.66% | 2 | 0 | 0.238 | 2.00 | 0 | 0% |
| instances above | | | | 280 | 3.99% | 119 | 4996 | 0 | 2.385 | 3393.05 | 0 | 83.9% | 2 | 0 | 0.711 | 4.00 | 0 | 0% |
| | | | | | | | | | | | | | | |
| | | 170 | 1% | 215 | 6291 | 0 | 0.687 | 3126.17 | 0 | 85.53% | 11 | 0 | 0.967 | 11.00 | 0 | 0% |

Table 4: Results for bipartite graphs

Under the 7, 200 seconds CPU time limit imposed, optimal PEDP and WPEDP solutions were obtained for all 363 EED graphs by both algorithms. The average CPU time taken by CPLEX-BC to solve a PEDP instance was 1.06 seconds while
the average LP relaxation gap it attained was 40.7%. Corresponding figures for BCA are respectively 56.3 and 43.3%. For WPEDP instances, the corresponding CPLEX-BC figures are 1.68 and 43.3%, while those for BCA are 25.3 and 64.3%. From average bipartite graph results, CPLEX-BC clearly outperforms BCA for these graphs.

Tables 4, 5, 6 and 7 show extreme CPU time and LP relaxation gaps for bipartite graph instances. From average bipartite graph results and also from an analysis of the extreme cases shown in the tables, WPEDP instances appear to be easier to solve than their PEDP counterparts. Accordingly, the picture that emerges here is the opposite of that we have for EED graphs.

| PEDP | WPEDP |
|------|-------|
| OPT  | NN    | t(s) | LBgü | NUC | GAP  | OPT  | NN    | t(s) | LBgü | NUC | GAP  |
| 270  | 1%    | 233  | 82   | 1791 | 4.782 | 75.63 | 0    | 7.77% | 36490| 661  | 3.803 | 34228.50 | 0 | 6.2% |
| 260  | 1.99% | 162  | 256  | 427  | 4.617 | 46.98 | 0    | 81.65%| 126532| 133  | 2.832 | 21867.50 | 0 | 82.72%|
| 290  | 2.98% | 140  | 42   | 421  | 3.624 | 34.42 | 0    | 18.04%| 20464| 628  | 4.907 | 13380.10 | 0 | 34.62%|
| 290  | 24.66%| 49   | 290  | 129  | 3.211 | 12.47 | 0    | 95.75%| 153572| 95   | 5.159 | 39099.79 | 0 | 97.45%|
| 300  | 4.91% | 111  | 300  | 35   | 3.137 | 29.40 | 0    | 90.2% | 152468| 126  | 4.075 | 10545.50 | 0 | 93.08%|

| Five highest CPU times for CPLEX-BC | 
|-------------------------------------|
| Time | Instance | OPT  | NN    | t(s) | LBgü | NUC | GAP  |
| 270  | 1%        | 82   | 1791  | 4.782 | 75.63 | 0 | 7.77% |
| 260  | 1.99%     | 256  | 427   | 4.617 | 46.98 | 0 | 81.65%|
| 290  | 2.98%     | 42   | 421   | 3.624 | 34.42 | 0 | 18.04%|
| 290  | 24.66%    | 290  | 129   | 3.211 | 12.47 | 0 | 95.75%|
| 300  | 4.91%     | 300  | 35    | 3.137 | 29.40 | 0 | 90.2% |

| BCA results for the same instances above | 
|------------------------------------------|
| Time | Instance | OPT  | NN    | t(s) | LBgü | NUC | GAP  |
| 270  | 1%        | 82   | 1791  | 4.782 | 75.63 | 0 | 7.77% |
| 260  | 1.99%     | 256  | 427   | 4.617 | 46.98 | 0 | 81.65%|
| 290  | 2.98%     | 42   | 421   | 3.624 | 34.42 | 0 | 18.04%|
| 290  | 24.66%    | 290  | 129   | 3.211 | 12.47 | 0 | 95.75%|
| 300  | 4.91%     | 300  | 35    | 3.137 | 29.40 | 0 | 90.2% |

| Five lowest CPU times for CPLEX-BC | 
|-------------------------------------|
| Time | Instance | OPT  | NN    | t(s) | LBgü | NUC | GAP  |
| 30   | 25%       | 7    | 21    | 0.037 | 4.43 | 0 | 38.66%|
| 30   | 28.57%    | 3    | 0     | 0.047 | 2.85 | 0 | 4.91% |
| 50   | 36.76%    | 4    | 0     | 0.057 | 3.40 | 0 | 15% |
| 300  | 1.99%     | 15   | 0     | 0.058 | 15.00 | 0 | 0% |
| 300  | 4.96%     | 11   | 0     | 0.089 | 11.00 | 0 | 0% |

| BCA results for the same instances above | 
|------------------------------------------|
| Time | Instance | OPT  | NN    | t(s) | LBgü | NUC | GAP  |
| 30   | 25%       | 7    | 21    | 0.037 | 4.43 | 0 | 38.66%|
| 30   | 28.57%    | 3    | 0     | 0.047 | 2.85 | 0 | 4.91% |
| 50   | 36.76%    | 4    | 0     | 0.057 | 3.40 | 0 | 15% |
| 300  | 1.99%     | 15   | 0     | 0.058 | 15.00 | 0 | 0% |
| 300  | 4.96%     | 11   | 0     | 0.089 | 11.00 | 0 | 0% |

Table 4: Extreme PEDP cases and WPEDP complementary information.
### Table 5: Extreme PEDP cases and WPEDP complementary information.

| [E] | dens. (%) | [V] | OPT N N | t(s) | LBg | NUC | GAP |
|-----|-----------|-----|---------|------|-----|-----|-----|
| Five highest gaps for CPLEX-BC |
| 260 33.33% 40 | 260 80 | 2.141 | 8.93 | 0 96.57% |
| 300 38.46% 40 | 300 15 | 2.065 | 10.61 | 0 96.46% |
| 290 29.29% 45 | 290 12 | 1.483 | 10.68 | 0 96.12% |
| 0.00 0.00 |
| 260 33.33% 40 | 260 219 | 509.720 | 8.69 | 953 96.66% |
| 300 38.46% 40 | 300 71 | 250.260 | 10.33 | 822 96.56% |
| 290 29.29% 45 | 290 12 | 6.097 | 3.00 | 0 0% |
| Five lowest gaps for CPLEX-BC |
| 40 51.28% 13 | 40 5 | 0.548 | 5.00 | 0 0% |
| 50 14.25% 27 | 50 4 | 0.666 | 4.00 | 0 0% |
| 40 51.28% 13 | 40 5 | 0.548 | 5.00 | 0 0% |
| 50 14.25% 27 | 50 4 | 0.666 | 4.00 | 0 0% |
| Five lowest gaps for CPLEX-BC |
| 40 51.28% 13 | 40 5 | 0.548 | 5.00 | 0 0% |
| 50 14.25% 27 | 50 4 | 0.666 | 4.00 | 0 0% |
| BCA results for the same instances above |
| 270 3.98% 117 | 270 226 | 64.464 | 22.45 | 470 16.87% |
| 290 14.85% 63 | 290 261 | 118.512 | 3383.94 | 953 97.8% |

### Table 6: Extreme WPEDP cases and PEDP complementary information.

| [E] | dens. (%) | [V] | OPT N N | t(s) | LBg | NUC | GAP |
|-----|-----------|-----|---------|------|-----|-----|-----|
| Five highest CPU times for CPLEX-BC |
| 40 29.41% 17 | 40 5 | 0.036 | 451.97 | 0 21.4% |
| 220 0.99% 211 | 220 36 | 0.074 | 3876.50 | 0 2.03% |
| 120 9.8% 50 | 120 434 | 0.084 | 429.76 | 0 0.98% |
| 90 3% 78 | 90 8620 | 0.095 | 8500.88 | 0 1.15% |
| Five lowest CPU times for CPLEX-BC |
| 40 29.41% 17 | 40 5 | 0.036 | 451.97 | 0 21.4% |
| 180 2.95% 111 | 180 2146 | 0.051 | 2081.77 | 0 2.99% |
| 220 0.99% 211 | 220 3957 | 0.074 | 3876.50 | 0 2.03% |
| 120 9.8% 50 | 120 434 | 0.084 | 429.76 | 0 0.98% |
| BCA results for the same instances above |
| 120 9.8% 50 | 120 434 | 0.084 | 429.76 | 0 0.98% |
| 90 3% 78 | 90 8620 | 0.095 | 8500.88 | 0 1.15% |
Table 7: Extreme WPEDP cases and PEDP complementary information.

### 4.3 Results for 3-regular graphs

Under the 7,200 seconds CPU time limit imposed, optimal PEDP and WPEDP solutions were obtained for all 10 CR graphs by both algorithms. The average CPU time taken by CPLEX-BC to solve a PEDP instance was 11.85 seconds while the average LP relaxation gap it attained was 63%. Corresponding figures for BCA are respectively 18.95 and 64.85%. For WPEDP instances, the corresponding CPLEX-BC figures are 17.68 and 65.93%, while those for BCA are 38.80 and 72.80%. From average CR graph results, CPLEX-BC clearly outperforms BCA for these graphs. Furthermore, CR graph instances appear to be the hardest to solve in our test set.

Tables 8, 9, 10 and 11 show extreme CPU time and LP relaxation gaps for CR graph instances.
Table 8: Extreme PEDP cases and WPEDP complementary information.

| E | den(s.%) | | V | OPT | N.N | t(s) | LB | NUC | GAP | OPT | N.N | t(s) | LB | NUC | GAP |
|---|----------|---|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 50 | 5.08% | 60 | 90 | 728 | 0.735 | 19.03 | 0 | 78.86% | 45141 | 484 | 1.222 | 8389.91 | 0 | 81.41% |
| 30 | 15.79% | 20 | 30 | 53 | 0.869 | 6.48 | 0 | 78.39% | 31511 | 389 | 0.604 | 5930.93 | 0 | 81.18% |
| 120 | 3.8% | 80 | 60 | 502 | 1.581 | 24.76 | 0 | 58.74% | 29526 | 529 | 1.398 | 10872.90 | 0 | 63.18% |

Table 9: Extreme PEDP cases and WPEDP complementary information.

| E | den(s.%) | | V | OPT | N.N | t(s) | LB | NUC | GAP | OPT | N.N | t(s) | LB | NUC | GAP |
|---|----------|---|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 50 | 5.08% | 60 | 90 | 2427 | 2.797 | 18.00 | 172 | 80% | 45141 | 1277 | 2.102 | 6964.47 | 159 | 84.57% |
| 30 | 15.79% | 20 | 30 | 53 | 0.854 | 6.00 | 31 | 80% | 31511 | 522 | 0.794 | 4550.62 | 105 | 85.56% |
| 120 | 3.8% | 80 | 60 | 1532 | 3.164 | 24.00 | 205 | 60% | 29526 | 2327 | 3.379 | 8397.46 | 217 | 71.56% |

Table 8: Extreme PEDP cases and WPEDP complementary information.

| E | den(s.%) | | V | OPT | N.N | t(s) | LB | NUC | GAP | OPT | N.N | t(s) | LB | NUC | GAP |
|---|----------|---|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 50 | 5.08% | 60 | 90 | 2427 | 2.797 | 18.00 | 172 | 80% | 45141 | 1277 | 2.102 | 6964.47 | 159 | 84.57% |
| 30 | 15.79% | 20 | 30 | 53 | 0.854 | 6.00 | 31 | 80% | 31511 | 522 | 0.794 | 4550.62 | 105 | 85.56% |
| 120 | 3.8% | 80 | 60 | 1532 | 3.164 | 24.00 | 205 | 60% | 29526 | 2327 | 3.379 | 8397.46 | 217 | 71.56% |
Table 10: Extreme WPEDP cases and PEDP complementary information.

| WPEDP | PEDP |
|-------|------|
| **Five highest CPU times for CPLEX-BC** | **Five lowest CPU times for CPLEX-BC** |
| OPT | N | N | t(s) | LBg | NUC | GAP | OPT | N | N | t(s) | LBg | NUC | GAP |
| (s) | (%) | | | | | | (s) | (%) | | | | | |
| 90 | 5.08% | 60 | 45141 | 484 | 1.220 | 8389.91 | 0 | 81.41% | 90 | 728 | 0.730 | 19.03 | 0 | 78.86% |
| 60 | 7.69% | 40 | 31511 | 389 | 0.600 | 5930.93 | 0 | 81.18% | 60 | 269 | 1.050 | 12.71 | 0 | 78.81% |
| 180 | 15.79% | 20 | 15218 | 85 | 0.320 | 2873.61 | 0 | 81.12% | 180 | 29526 | 529 | 1.598 | 10872.90 | 0 | 63.16% |
| 30 | 5.08% | 60 | 45141 | 484 | 2.102 | 6964.47 | 159 | 84.57% | 30 | 502 | 5.181 | 24.76 | 0 | 58.74% |
| 110808 | 825 | 6.470 | 21488.10 | 0 | 80.61% | | | | | | | | |
| 180 | 2.52% | 120 | 28055 | 424 | 2.920 | 16560.00 | 0 | 40.97% | 180 | 60971 | 1.412 | 8.00 | 0 | 20.05 |
| 210 | 2.16% | 140 | 110808 | 5624 | 14.967 | 17072.60 | 371 | 84.59% | 210 | 9615 | 0.735 | 9.03 | 0 | 80.61% |
| 30 | 15.79% | 20 | 15218 | 71 | 0.421 | 2310.42 | 36 | 84.82% | 30 | 53 | 0.854 | 6.00 | 31 | 80% |
| 60 | 7.69% | 40 | 31511 | 522 | 0.794 | 4550.62 | 105 | 85.56% | 60 | 443 | 1.192 | 12.00 | 99 | 80% |
| 90 | 5.08% | 60 | 45141 | 484 | 2.102 | 6964.47 | 159 | 84.57% | 90 | 2427 | 2.707 | 18.00 | 172 | 80% |
| 120 | 3.8% | 80 | 29526 | 2327 | 3.379 | 8397.46 | 217 | 71.56% | 120 | 1532 | 3.164 | 24.00 | 205 | 60% |
| 180 | 2.52% | 120 | 28055 | 2572 | 6.6 | 13085.50 | 289 | 53.36% | 180 | 1907 | 4.987 | 36.00 | 267 | 40% |

Table 11: Extreme WPEDP cases and PEDP complementary information.

| WPEDP | PEDP |
|-------|------|
| **Five highest CPU times for CPLEX-BC** | **Five lowest CPU times for CPLEX-BC** |
| OPT | N | N | t(s) | LBg | NUC | GAP | OPT | N | N | t(s) | LBg | NUC | GAP |
| (s) | (%) | | | | | | (s) | (%) | | | | | |
| 90 | 5.08% | 60 | 45141 | 484 | 1.220 | 8389.91 | 0 | 81.41% | 90 | 728 | 0.730 | 19.03 | 0 | 78.86% |
| 60 | 7.69% | 40 | 31511 | 389 | 0.600 | 5930.93 | 0 | 81.18% | 60 | 269 | 1.050 | 12.71 | 0 | 78.81% |
| 180 | 15.79% | 20 | 15218 | 85 | 0.320 | 2873.61 | 0 | 81.12% | 180 | 29526 | 529 | 1.598 | 10872.90 | 0 | 63.16% |
| 30 | 5.08% | 60 | 45141 | 484 | 2.102 | 6964.47 | 159 | 84.57% | 30 | 502 | 5.181 | 24.76 | 0 | 58.74% |
| 110808 | 825 | 6.470 | 21488.10 | 0 | 80.61% | | | | | | | | |
| 180 | 2.52% | 120 | 28055 | 424 | 2.920 | 16560.00 | 0 | 40.97% | 180 | 60971 | 1.412 | 8.00 | 0 | 20.05 |
| 210 | 2.16% | 140 | 110808 | 5624 | 14.967 | 17072.60 | 371 | 84.59% | 210 | 9615 | 0.735 | 9.03 | 0 | 80.61% |
| 30 | 15.79% | 20 | 15218 | 71 | 0.421 | 2310.42 | 36 | 84.82% | 30 | 53 | 0.854 | 6.00 | 31 | 80% |
| 60 | 7.69% | 40 | 31511 | 522 | 0.794 | 4550.62 | 105 | 85.56% | 60 | 443 | 1.192 | 12.00 | 99 | 80% |
| 90 | 5.08% | 60 | 45141 | 484 | 2.102 | 6964.47 | 159 | 84.57% | 90 | 2427 | 2.707 | 18.00 | 172 | 80% |
| 120 | 3.8% | 80 | 29526 | 2327 | 3.379 | 8397.46 | 217 | 71.56% | 120 | 1532 | 3.164 | 24.00 | 205 | 60% |
| 180 | 2.52% | 120 | 28055 | 2572 | 6.6 | 13085.50 | 289 | 53.36% | 180 | 1907 | 4.987 | 36.00 | 267 | 40% |

5 Conclusions

The paper suggests and investigates what appears to be the very first formulation proposed for the Perfect Edge Domination Problem. So far, the problem
has been studied mostly in computational complexity terms and no exact algorithm, Combinatorial or IP based, was previously suggested for it. Based on our formulation, two Branch-and-Cut algorithms were designed, implemented and tested for the problem. Extensive computational tests are reported for the best performing of the two, together with results obtained by IP solver CPLEX, used as a stand alone Branch-and-Cut algorithm. Test instances used in these experiments originate from quite elaborated procedures aimed at generating challenging instances for the problem.

From the computational results obtained, one conclusion that may be drawn is that there exists plenty of room for investigating valid inequalities to reinforce our PEDP formulation. With the single exception of non weighted EED graph test instances, LP relaxation gaps for the remaining instances are, on the average, quite high. Furthermore, the tailor made Branch-and-Cut algorithms we tested lagged well behind CPLEX in terms of CPU time and, to a lesser extent, LP relaxation gaps. The first part of the remark should probably be taken with caution since all CPLEX pre-processing, cutting plane, and heuristic modules were kept switched off for our algorithms. However, it is quite surprising that the tailor-made cutting planes we use are outperformed by the generic ones used by CPLEX. Accordingly, attempting to lift known inequalities for the Set Covering polytope [3] is an alternative one should probably investigate. However, these inequalities have been of limited use, so far, even for the Set Covering Problem itself. The investigation of PEDP heuristics is also another candidate research topic to pursue.

Finally, taking a broader view of the subject, we also plan to investigate formulations and exact algorithms for the Minimum Edge Dominating Set Problem and the Efficient Edge Domination Problem.

References

[1] Enide Andrade, Domingos M. Cardoso, Luis Medina, and Oscar Rojo. On the dominating induced matching problem: Spectral results and sharp bounds. *Discrete Applied Mathematics*, 234:22–31, 2018. Special Issue on the Ninth International Colloquium on Graphs and Optimization (GO IX), 2014.

[2] M. Bodur, T. Ekim, and Z.C. Taskin. Decomposition algorithms for solving the minimum weight maximal matching problem. *Networks*, 62(4):273–287, 2013.

[3] R. Borndörfer. *Aspects of Set Packing, Partitioning, and Covering*. PhD thesis, 1998.

[4] A. Brandstädt, C. Hundt, and R. Nevyries. Efficient edge domination on hole-free graphs in polynomial time. In *Proceedings of the 9th Latin American conference on Theoretical Informatics*, LATIN’10, pages 650–661, Berlin, Heidelberg, 2010. Springer-Verlag.
[5] A. Brandstädt, A. Leitert, and D. Rautenbach. Efficient dominating and edge dominating sets for graphs and hypergraphs. In *Algorithms and Computation - 23rd International Symposium, ISAAC 2012, Taipei, Taiwan, December 19-21, 2012. Proceedings*, pages 267–277, 2012.

[6] A. Brandstädt and R. Mosca. Dominating induced matchings for $P_7$-free graphs in linear time. *CoRR*, abs/1106.2772, 2011.

[7] A. Brandstädt and R. Mosca. Finding dominating induced matchings in $P_8$-free graphs in polynomial time. *Algorithmica*, 77(4):1283–1302, 2017.

[8] D. M. Cardoso, J. O. Cerdeira, C. Delorme, and P. C. Silva. Efficient edge domination in regular graphs. *Discrete Applied Mathematics*, 156(15):3060–3065, 2008.

[9] D. M. Cardoso, N. Korpelainen, and V. V. Lozin. On the complexity of the dominating induced matching problem in hereditary classes of graphs. *Discrete Appl. Math.*, 159(7):521–531, April 2011.

[10] G.J. Chang and S. Hwang. The edge domination problem. *Discussiones Mathematicae Graph Theory*, 15(1):51–57, 1995.

[11] M. Demange and T. Ekim. Minimum maximal matching is np-hard in regular bipartite graphs. In *Theory and Applications of Models of Computation, 5th International Conference, TAMC 2008, Xi’an, China, April 25-29, 2008. Proceedings*, pages 364–374, 2008.

[12] D. L. Grinstead, P. J. Slater, N. A. Sherwani, and N. D. Holmes. Efficient edge domination problems in graphs. *Inf. Process. Lett.*, 48(5):221–228, 1993.

[13] A. Hertz, V.V. Lozin, B. Ries, V. Zamaraev, and D. de Werra. Dominating induced matchings in graphs containing no long claw. *CoRR*, abs/1505.02558, 2015.

[14] D.J. Horton and K. Kilakos. Minimum edge dominating sets. *SIAM J. Discrete Math.*, 6(3):375–387, 1993.

[15] Ibm. *IBM ILOG CPLEX Optimization Studio V12.6.0 documentation*, 2017.

[16] N. Korpelainen. A polynomial-time algorithm for the dominating induced matching problem in the class of convex graphs. *Electronic Notes in Discrete Mathematics*, 32:133–140, 2009.

[17] M.C. Lin, V. Lozin, V.A. Moyano, and J.L. Szwarcfiter. Perfect edge domination: Hard and solvable cases. *Annals of Operations Research*, 2017.

[18] M.C. Lin, M.J. Mizrahi, and J.L. Szwarcfiter. Fast algorithms for some dominating induced matching problems. *Inf. Process. Lett.*, 114(10):524–528, 2014.
[19] M.C. Lin, M.J. Mizrahi, and J.L. Szwarcfiter. Efficient and perfect domination on circular-arc graphs. *Electronic Notes in Discrete Mathematics*, 50:307–312, 2015.

[20] M.C. Lin, M.J. Mizrahi, and J.L. Szwarcfiter. Exact algorithms for minimum weighted dominating induced matching. *Algorithmica*, 77(3):642–660, 2017.

[21] C. L. Lu, M. Ko, and C. Yu Tang. Perfect edge domination and efficient edge domination in graphs. *Discrete Applied Mathematics*, 119(3):227–250, 2002.

[22] C. L. Lu and C. Y. Tang. Solving the weighted efficient edge domination problem on bipartite permutation graphs. *Discrete Applied Mathematics*, 87(1-3):203–211, 1998.

[23] M.B. Richey and R.G. Parker. Minimum-maximal matching in series-parallel graphs. *European Journal of Operational Research*, 33(1):98 – 105, 1988.

[24] A. Srinivasan, K. Madhukar, P. Nagavamsi, C.P. Rangan, and M-S. Chang. Edge domination on bipartite permutation graphs and cotriangulated graphs. *Inf. Process. Lett.*, 56(3):165–171, 1995.

[25] Z.C. Taskin and T. Ekim. Integer programming formulations for the minimum weighted maximal matching problem. *Optimization Letters*, 6(6):1161–1171, 2012.

[26] M. Xiao and H. Nagamochi. Exact algorithms for dominating induced matching based on graph partition. *Discrete Applied Mathematics*, 190-191:147–162, 2015.

[27] M. Yannakakis and F. Gavril. Edge dominating sets in graphs. *SIAM Journal on Applied Mathematics*, 38(3):364–372, 1980.