AKCE J. Graphs. Combin., 2, No. 2 (2005), pp. 123-136

Domination Equivalence in Graphs

Jean R.S. Blair
Department of Electrical Engineering and Computer Science
United States Military Academy,West Point, NY, 10996, USA
E-mail: *Jean.Blair@usma.edu*

Wayne Goddard, Sandra M. Hedetniemi, Stephen T. Hedetniemi
Department of Computer Science
Clemson University, Clemson, SC, 29634, USA
E-mail: *Goddard@cs.clemson.edu* , *SHedet@cs.clemson.edu* , *Hedet@cs.clemson.edu*

and

Steven B. Horton
Department of Mathematical Sciences
United States Military Academy,West Point, NY, 10996, USA
E-mail: *Steve.Horton@usma.edu*

Communicated by: T.W. Haynes
Received 13 November 2005; accepted 30 December 2005

Abstract

A DE-pair (ODE-pair) in a graph consists of two disjoint subsets of vertices with the same closed neighborhood (open neighborhood). We consider the question of determining the smallest and largest subsets over all such pairs. We provide sharp bounds on these for general graphs and for trees, and show that the associated parameters are computable for trees but intractable in general.

Keywords: DE-pair, ODE-pair

2000 Mathematics Subject Classification: 05C69

1. Introduction

The question of when a graph has two disjoint subsets of vertices with the same neighborhoods has recently received attention ([3, 4]). In this paper we consider the question of determining the smallest and largest subsets over all such pairs.

Consider a graph. A DE-***pair*** (domination-equivalent pair) consists of two disjoint sets R and B of vertices whose closed neighborhoods are the same; that is, $N[R] = N[B]$. An ODE-***pair*** (open-domination-equivalent pair) consists of two disjoint sets R and B whose open neighborhoods are the same (meaning $N(R) = N(B)$). One may think of an ODE-pair as a partial coloring of the vertices with red and blue, such that every vertex that has a red neighbor also has a blue neighbor, and vice versa. There is a similar interpretation for DE-pairs.

A trivial DE- or ODE-pair is obtained from two empty sets. Indeed, if a graph has an isolated vertex v , then another trivial ODE-pair is $\{\emptyset, \{v\}\}$. We will always ignore the case where one or both the sets of a pair are empty. Thus, one may ask whether a nontrivial pair exists or not. This can be viewed as a special instance of the "Equal Union" problem, explored in [4, 5, 11] inter alia.

Ore [8] observed that, in a connected graph, the complement of any minimal dominating set is also a dominating set. So a nontrivial DE-pair always exists in a connected graph; indeed, both sets in the pair have the whole vertex set as their closed neighborhood. We say that a pair is ***full*** if the neighborhood is the whole graph. It follows that a graph has a nontrivial DE-pair iff it contains an edge; and it has a full DE-pair iff it is isolate-free.

1.1 Existence of ODE-pairs

By elementary considerations, it follows that an odd-order path has an ODE-pair (color the first vertex red, the third blue, the fifth red, and so on), but an even-order path does not. This dependence on the parity is a general result. It turns out that the question of the existence of ODE-pairs is linked to linear algebra.

Theorem 1.1 (Jacobs & Jamison [4]). *An* ODE*-pair exists in a graph iff either the adjacency matrix of the graph is nonsingular, or the graph has two cycle covers, one of which has an odd number of even cycles and the other an even number of even cycles. Equivalently, an* ODE*-pair exists iff the adjacency matrix is not sign-nonsingular.*

McCuaig [6] and Robertson, Seymour and Thomas [10] have shown that whether a matrix is sign-nonsingular can be determined in polynomial time. (For definitions we refer the reader to those papers.) This implies that there is a polynomial-time algorithm for determining whether a graph has an ODE-pair!

Corollary 1.2. *A tree has an* ODE*-pair iff it does not have a perfect matching (which can be determined in linear time).*

In particular, any odd-order tree has an ODE-pair.

On the other hand, the question of whether a graph has a full ODE-pair is equivalent to asking whether it has two disjoint total dominating sets, and this is known to be NP-hard [3].

1.2 Our results

In this paper we examine the smallest and largest sets of vertices in a DE- or ODE-pair. We provide (mostly) sharp bounds for trees and general graphs. We also show that, as expected, these parameters can be calculated for trees, but are NP-hard in general.

2. The Four Parameters

One can look at a DE- or ODE-pair and measure it in several ways. One way is to look at the cardinality of the common neighborhood, and ask for the smallest and largest neighborhoods that may result. Another possibility is to consider the combined cardinality of the pair. Here we consider the sizes of the sets in the pair. We define

$$\begin{aligned} de(G) &\equiv \text{the minimum cardinality of a member of a nontrivial DE-pair,} \\ ode(G) &\equiv \text{the minimum cardinality of a member of a nontrivial ODE-pair,} \\ DE(G) &\equiv \text{the maximum cardinality of a member of a DE-pair, and} \\ ODE(G) &\equiv \text{the maximum cardinality of a member of an ODE-pair.} \end{aligned}$$

2.1 Full DE-pairs

We observed earlier that if the graph is isolate-free, a minimum dominating set and its complement form a DE-pair. It follows that in a nonempty graph G that

$$de(G) \le \gamma(G),$$

and that $DE(G) \ge n - \gamma(G)$. However, in the latter inequality there is actually equality, as we now show.

Lemma 2.1. *Consider an isolate-free graph $G = (V, E)$. If $\{R, B\}$ is a DE-pair which is not full, then there exists a full pair $\{R', B'\}$ such that $R \subset R'$ and $B \subset B'$. That is, any non-full DE-pair can be extended to one which is full.*

Proof. Let $V' = V - (B \cup R)$. Since there is a vertex v that is not dominated by R (or equivalently B), and v is not isolated in G, the subgraph induced by V' contains at least one edge, and hence contains a nontrivial component, say H. As discussed above, H has a full DE-pair, say $\{R_1, B_1\}$. Then $\{R \cup R_1, B \cup B_1\}$ is a DE-pair which extends the original pair. The result follows by induction. □

It follows that:

Theorem 2.2. *In a graph G without isolates, $DE(G) = n - \gamma(G)$.*

Proof. By the above lemma, any DE-pair can be extended to a full pair. Since the smaller set in a full DE-pair is by definition dominating, it follows that $DE(G) \le n - \gamma(G)$. Equality is obtained by taking a minimum dominating set and its complement as a DE-pair. □

2.2 The ranges of de, ode and ODE

The natural limits on the other three parameters are summarized in the following theorem.

Theorem 2.3. *For any graph G of order n:*
(a) $1 \le de(G) \le n/2$ if G is nonempty.
(b) $1 \le ode(G) \le n/2$ if G has an ODE-pair.
(c) $1 \le ODE(G) \le n - 2$ if G has an ODE-pair and is isolate-free.
And these bounds are sharp, except for the upper bound on $de(G)$.

Proof. (a) For equality in the lower bound, just have two adjacent vertices with the same closed neighborhoods. For the upper bound, note that the smaller set in a pair can contain at most half the vertices. There are exactly three examples of equality in the upper bound: K_2, P_4 and C_4. (See discussion below.)

(b) The lower bound is achieved in any graph with two nonadjacent vertices with identical neighborhoods. The upper bound proof is as in (a). An example of equality in the upper bound is a barbell obtained by taking two cycles with length congruent to 1 modulo 4, and joining them by one edge: every ODE-pair must color half the vertices blue and half the vertices red. (For the proof of this, see Lemma 2.4 below.) An example of such a barbell is given in Figure 1.

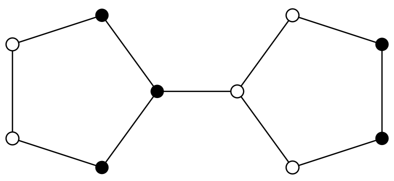

Figure 1: A barbell with its unique ODE-pair

(c) For equality in the lower bound, consider a path with even order and duplicate one end-vertex. The unique ODE-pair results from coloring the two leaves that have a common neighbor. For the upper bound, suppose there was an ODE-pair with $n-1$ blue vertices. Then the single red vertex has a blue neighbor, but not a red neighbor, a contradiction. For an example with $ODE(G) = n-2$ consider the star; color red one leaf and color blue all other leaves. □

We do not know the best upper bound on $de(G)$. We conjecture that it is the same as the bound given for trees (Theorem 2.5 below).

Conjecture 1. *For any graph G, $de(G) \le \lceil n/3 \rceil$.*

It can be shown that $de(G) = n/2$ is not achievable for $n > 4$. For, such a value clearly requires G to be connected, and for $\gamma(G)$ to be $n/2$, which means that G is C_4 or a corona $H \circ K_1$ [9]. So if $|H| \ge 3$, then there exists a vertex v with degree in H at least 2. Let w be a neighbor of v in H. Color red w and all leaf-vertices of the corona except for the two adjacent to w and v; and color blue the leaf-vertex adjacent to w and all vertices of H except v and w. This shows that $de(H \circ K_1) \le n/2 - 1$.

We also need to show that the barbell constructed in the proof of Theorem 2.3(b) has the claimed property. To this end, we define a ***lariat*** as an odd cycle where all but one vertex—the large-degree vertex—has degree exactly 2.

Lemma 2.4. *Consider a graph with an* ODE*-pair and let C be a lariat. If any vertex of C is colored, then every vertex of C is colored. Furthermore, if $|C| \equiv 1 \pmod 4$, then the large-degree vertex has the same color as both its neighbors in C.*

Proof. Let the vertices of C be $x_0, x_1, \ldots, x_{n-1}, x_0$, where x_0 is the large-degree vertex. Then x_i being colored implies $x_{i\pm 2}$ has the opposite color (arithmetic modulo n), except when the common neighbor is x_0 (that is, x_1 being colored does not directly imply x_{n-1} being colored, and vice versa). □

2.3 Trees

The lower bounds for all four parameters are the same for trees as they are for general graphs, as is the upper bound for $ODE(G)$. The upper bounds for $de(G)$ and $ode(G)$, however, can be strengthened when G is a tree.

Theorem 2.5. *If T is a tree of order n, then $de(T) \le \lceil n/3 \rceil$, and this bound is sharp.*

Proof. The result is true if the diameter is at most two, since then $\gamma(T) = 1$, and if the diameter is three, since then $\gamma(T) = 2$ and $n \ge 4$. The proof in general is by induction on the order n.

So let T be a tree of diameter at least 4. Consider a diametral path $a, b, c, d, e, \ldots$. If b has degree three or more, then let T' be the component of $T - bc$ containing c. Apply the inductive hypothesis to color T' with an ODE-pair using at most $\lceil |T'|/3 \rceil$ red vertices. If c does not receive a color, then we are done. Otherwise, color b with red and color its leaf-neighbors (which are at least two) with blue.

On the other hand, if b has degree two, then let T' be the component of $T - cd$ containing d. Again apply the inductive hypothesis to T'. If d does not receive a color, then we are done. Otherwise, color a with the same color as d and b the opposite color.

The result is sharp for a path. □

Turning to the parameter *ode*, we will need the following observation (which can be deduced from Corollary 1.2, but we need the constructive version).

Observation 2.6. *Let T be a tree of order at least 4 and suppose a vertex v of degree 2 has a leaf neighbor w and other neighbor x. Then T has an* ODE*-pair iff $T' = T - \{v, w\}$ has one.*

Proof. Suppose T has an ODE-pair. Since w has only one neighbor, vertex v cannot be colored. It follows that the coloring of T restricted to T' is an ODE-pair in T'. On the other hand, suppose T' has an ODE-pair. Then this is an ODE-pair in T if x is not colored; and it becomes an ODE-pair in T by coloring w the opposite color of x otherwise. □

Theorem 2.7. *If T is a tree of order n and has an* ODE*-pair, then $ode(T) \leq (n+1)/4$, and this bound is sharp.*

Proof. We argue that one can always find an ODE-pair where the number of colored vertices is at most $(n+1)/2$. From this the desired bound follows.

The bound on the number of colored vertices is proven by induction on the order. The base case is the path on 3 vertices: this has an ODE-pair with two colored vertices.

So assume $n \geq 4$, and let T be a tree of order n with an ODE-pair. If in T there are two leaves with a common neighbor, then color one leaf red and one blue and so $ode(T) = 1$. Otherwise, there is a vertex v of degree 2 with a leaf neighbor w and other neighbor x. Let tree T' be T with vertices v and w removed. By Observation 2.6, T' has an ODE-pair. Hence, by the inductive hypothesis, T' has an ODE-pair with at most $(n-1)/2$ colored vertices. We might need to color w to extend to an ODE-pair of T, but in any case T has an ODE-pair with at most $(n+1)/2$ colored vertices.

An example of equality in the theorem is an odd-order path. □

Another useful fact is that for any graph the subgraph induced by the colored vertices does not have a vertex of degree 1. From this it follows that:

Lemma 2.8. *In an* ODE*-pair, no vertex adjacent to a leaf is colored. Furthermore, in a tree, the colored vertices form an independent set.*

3. Algorithms for Trees

The algorithms use a standard (dynamic programming) postorder traversal, motivated by the methodology of Borie et al. [1] or Wimer [12]. Consider a tree T rooted at r. For a vertex v we denote by T_v the subtree consisting of v and all its descendants. We let $C(v)$ denote the children of v.

The idea is to consider a "quasi-coloring" of T_v as the restriction of an ODE-pair or DE-pair to that subtree. Then what matters when this quasi-coloring is extended is the color of v and by what colors it is dominated. This gives a finite number of ***properties***. If one knows the optimum quasi-coloring for each property and each child subtree T_c, $c \in C(v)$, one can determine the optimum quasi-coloring for T_v for each property.

In each case, then, the task is to identify the relevant properties. Then one determines a formula for the initial parameter-value associated with each property at a leaf, the formula for processing $\{T_c\}$ into T_v, and the formula for deducing the best coloring from the optimal quasi-colorings at the root.

We know that $DE(T) = n - \gamma(T)$ for a tree T. So we give algorithms for the other three parameters.

3.1 Maximum open domination equivalence

In this section we give a linear-time algorithm for computing $ODE(T)$, where T is an arbitrary tree.

Let c_v[property] be the maximum $|R'|$ in the restriction $\{B', R'\}$ to T_v of an ODE-pair $\{B, R\}$ such that the vertex v has the given property. Lemma 2.8 reduces the properties to be maintained. These are:

- *Red*: the property R means that $v \in R$,
- *Blue*: the property B means that $v \in B$,
- *Needs-red*: the property nr means v is uncolored and blue- but not red-dominated (by its children),
- *Needs-blue*: the property nb means v is uncolored and is red- but not blue-dominated,
- *Has-both*: the property hb means v is uncolored but is both red- and blue-dominated,
- *Has-neither*: the property hn means v is uncolored and is neither red- nor blue-dominated.

We will use the value $c_v = -\infty$ to mean that there is no such ODE-pair.

The algorithm for computing ODE on a tree is a postorder traversal with the initialization, processing, and best-at-root procedures as defined in Figure 2.

It is easy to see that ODE-INITIALIZE runs in $O(n)$ time and ODE-BEST-AT-ROOT in $O(1)$ time. We will show that a call to ODE-PROCESS for vertex v uses $O(\deg(v))$ time. It follows that the total time complexity for the ODE algorithm is $O(n)$.

To see that the time for a call to ODE-PROCESS is $O(\deg(v))$, observe that computation of $c_v[\mathsf{R}]$, $c_v[\mathsf{B}]$, and $c_v[\mathsf{hn}]$ clearly requires only $O(\deg(v))$ time. To compute the value for $c_v[\mathsf{hb}]$ when v has at least two children, define d_w as $\max\{c_w[\mathsf{R}], c_w[\mathsf{B}], c_w[\mathsf{hb}], c_w[\mathsf{hn}]\}$. Then

$$c_v[\mathsf{hb}] = \sum_{w \in C(v)} d_w - \min_{x,y \in C(v),\, x \neq y} (d_x - c_x[\mathsf{R}]) + (d_y - c_y[\mathsf{B}]).$$

```
ODE-Initialize:
for each leaf $v$ do
    $c_v[\mathsf{R},\mathsf{B},\mathsf{nr},\mathsf{nb},\mathsf{hb},\mathsf{hn}] \leftarrow [1,0,-\infty,-\infty,-\infty,0]$

ODE-Process( $v$ ):
$c_v[\mathsf{R}] \leftarrow 1+\sum_{x\in C(v)} \max\{c_x[\mathsf{nr}], c_x[\mathsf{hb}]\}$
$c_v[\mathsf{B}] \leftarrow \sum_{x\in C(v)} \max\{c_x[\mathsf{nb}], c_x[\mathsf{hb}]\}$
$c_v[\mathsf{nr}] \leftarrow \max_{x\in C(v)} \{c_x[\mathsf{B}] + \sum_{y\in C(v)-\{x\}} \max\{c_y[\mathsf{B}], c_y[\mathsf{hb}], c_y[\mathsf{hn}]\}\}$
$c_v[\mathsf{nb}] \leftarrow \max_{x\in C(v)} \{c_x[\mathsf{R}] + \sum_{y\in C(v)-\{x\}} \max\{c_y[\mathsf{R}], c_y[\mathsf{hb}], c_y[\mathsf{hn}]\}\}$
$c_v[\mathsf{hn}] \leftarrow \sum_{x\in C(v)} \max\{c_x[\mathsf{hb}], c_x[\mathsf{hn}]\}$
if $|C(v)|=1$ then $c_v[\mathsf{hb}] \leftarrow -\infty$
    else $c_v[\mathsf{hb}] \leftarrow \max_{x,y\in C(v),\, x\neq y} \{c_x[\mathsf{R}] + c_y[\mathsf{B}] + \sum_{w\in C(v)-\{x,y\}} \max\{c_w[\mathsf{R}], c_w[\mathsf{B}], c_w[\mathsf{hb}], c_w[\mathsf{hn}]\}\}$

ODE-Best-At-Root:
return $\max\{c_r[\mathsf{R}], c_r[\mathsf{B}], c_r[\mathsf{hb}], c_r[\mathsf{hn}]\}$
```

Figure 2: Computing $ODE(T)$ for a tree

It follows that the optimal x has either the smallest or the second-smallest value of $d_x - c_x[\mathsf{R}]$, and similarly with y. The optimal x and y, and hence $c_v[\mathsf{hb}]$, can thus be calculated in time $O(\deg(v))$. Similarly, $c_v[\mathsf{nr}]$ and $c_v[\mathsf{nb}]$ can be computed in time $O(\deg(v))$.

It is straight-forward to establish correctness of the algorithm. Thus, we have:

Theorem 3.1. *The algorithm described by Figure 2 will determine $ODE(T)$ in $O(n)$ time when T is a tree.*

Now we demonstrate the ODE algorithm by determining disjoint subsets R and B of V such that $\{R,B\}$ is an ODE-pair and $|R|$ is maximized. The values of c_{v_i} for each vertex in the tree given in Figure 3 are shown in Table 1 below.

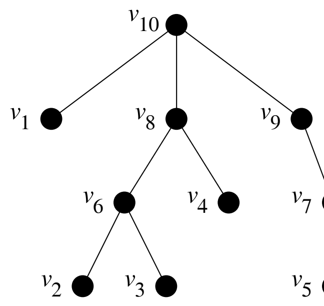

Figure 3: An example tree for MAX-ODE(T)

The c_{v_i} values for $v_1 - v_5$ come directly from ODE-Initialize. The c_{v_i} values for $v_6 - v_{10}$ are computed from their children using ODE-Process. Finally, once the vector $c_{v_{10}}$ is computed, ODE-Best-At-Root is run to determine ODE. Since $c_{v_{10}}[\mathsf{nr}]$ and $c_{v_{10}}[\mathsf{nb}]$ represent colorings that are not ODE-pairs, ODE-Best-At-Root accepts as candidate solutions for $ODE(T)$ only the other four components of $c_{v_{10}}$, which establishes that $ODE(T)$ is 3. Once $ODE(T)$ is

Table 1: c_{v_i}[property]

property	v_1–v_5	v_6	v_7	v_8	v_9	v_{10}
R	1	$-\infty$	$-\infty$	$-\infty$	1	$-\infty$
B	0	$-\infty$	$-\infty$	$-\infty$	1	$-\infty$
nr	$-\infty$	0	0	1	$-\infty$	2
nb	$-\infty$	2	1	2	$-\infty$	3
hb	$-\infty$	1	$-\infty$	$-\infty$	$-\infty$	3
hn	0	0	0	1	0	1

computed, it is easy to backtrack through ODE-Process to find an optimal coloring. One choice is $R = \{v_1, v_3, v_5\}$ and $B = \{v_2, v_9\}$.

3.2 Minimum open domination equivalence

The linear-time algorithm for computing $ode(T)$ is similar to the algorithm presented in the previous subsection. Here, however, we must calculate the *minimum* $|R|$ for $\{B, R\}$ an ODE-pair of T, if one exists. In this case we will use the value ∞ to mean that no ODE-pair exists. The algorithm for computing $ode(T)$ on a tree is a postorder traversal with the initialization, processing, and best-at-root procedures as defined in Figure 4. Note that $c_v[\mathsf{hn}]$ is initialized to be ∞. This ensures that a trivial ODE-pair has value ∞.

```
ODE-Initialize:
for each leaf $v$ do
    $c_v[\mathsf{R}, \mathsf{B}, \mathsf{nr}, \mathsf{nb}, \mathsf{hb}, \mathsf{hn}] \leftarrow [1, 0, \infty, \infty, \infty, \infty]$

ODE-Process( $v$ ):
$c_v[\mathsf{R}] \leftarrow 1 + \sum_{x \in C(v)} \min\{c_x[\mathsf{nr}], c_x[\mathsf{hb}]\}$
$c_v[\mathsf{B}] \leftarrow \sum_{x \in C(v)} \min\{c_x[\mathsf{nb}], c_x[\mathsf{hb}]\}$
$c_v[\mathsf{nr}] \leftarrow \min_{x \in C(v)}\{c_x[\mathsf{B}]\}$
$c_v[\mathsf{nb}] \leftarrow \min_{x \in C(v)}\{c_x[\mathsf{R}]\}$
$c_v[\mathsf{hn}] \leftarrow \min_{x \in C(v)}\{c_x[\mathsf{hb}], c_x[\mathsf{hn}]\}$
if $|C(v)| = 1$ then $c_v[\mathsf{hb}] \leftarrow \infty$
    else $c_v[\mathsf{hb}] \leftarrow \min_{x,y \in C(v), x \neq y}\{c_x[\mathsf{R}] + c_y[\mathsf{B}]\}$

ODE-Best-At-Root:
return $\min\{c_r[\mathsf{R}], c_r[\mathsf{B}], c_r[\mathsf{hb}], c_r[\mathsf{hn}]\}$
```

Figure 4: Computing $ode(T)$ for a tree

3.3 Minimum domination equivalence

The linear-time algorithm for computing $de(T)$ uses different properties than those used for open domination equivalence.

Here, since a vertex dominates itself, there are eight properties. The properties Rnn (Red-no-need) and Bnn (Blue-no-need) mean the current vertex is in R or B, respectively, with the other domination from its children. The properties Rnb (Red-need-blue) and Bnr (Blue-need-red) mean the current vertex is in R or B, respectively, but has no domination from its children. The remaining properties represent an uncolored vertex: Wnn (White-no-need) has both red- and blue-domination; Wnr (White-need-red) has only blue-domination; Wnb (White-need-blue) has only red-domination; and Wnw (White-need-white) has no domination from its children.

As before, we will use the value ∞ to mean that no such DE-pair exists. The algorithm for computing de on a tree is a postorder traversal with the initialization, processing, and best-at-root procedures as defined in Figure 5. Again, $c_v[\mathsf{Wnw}] = \infty$ for a leaf to prevent the trivial result.

```
DE-INITIALIZE:
for each leaf $v$ do
    $c_v[\mathsf{Rnn}, \mathsf{Bnn}, \mathsf{Rnb}, \mathsf{Bnr}, \mathsf{Wnn}, \mathsf{Wnr}, \mathsf{Wnb}, \mathsf{Wnw}] \leftarrow [\infty, \infty, 1, 0, \infty, \infty, \infty, \infty]$

DE-PROCESS($v$) :
$c_v[\mathsf{Rnn}] \leftarrow 1 + \min_{x \in C(v)} \{ \min\{c_x[\mathsf{Bnn}], c_x[\mathsf{Bnr}]\}$
        $+ \sum_{y \in C(v)-\{x\}} \min\{c_y[\mathsf{Rnn}], c_y[\mathsf{Bnn}], c_y[\mathsf{Bnr}], c_y[\mathsf{Wnr}], c_y[\mathsf{Wnn}]\}\}$
$c_v[\mathsf{Bnn}] \leftarrow \min_{x \in C(v)} \{ \min\{c_x[\mathsf{Rnn}], c_x[\mathsf{Rnb}]\}$
        $+ \sum_{y \in C(v)-\{x\}} \min\{c_y[\mathsf{Rnn}], c_y[\mathsf{Bnn}], c_y[\mathsf{Rnb}], c_y[\mathsf{Wnb}], c_y[\mathsf{Wnn}]\}\}$
$c_v[\mathsf{Rnb}] \leftarrow 1 + \sum_{x \in C(v)} \min\{c_x[\mathsf{Rnn}], c_x[\mathsf{Wnr}], c_x[\mathsf{Wnn}]\}$
$c_v[\mathsf{Bnr}] \leftarrow \sum_{x \in C(v)} \min\{c_x[\mathsf{Bnn}], c_x[\mathsf{Wnb}], c_x[\mathsf{Wnn}]\}$
$c_v[\mathsf{Wnn}] \leftarrow \min_{x,y \in C(v),\, x \neq y}\{c_x[\mathsf{Rnn}] + c_y[\mathsf{Bnn}]\}$
$c_v[\mathsf{Wnr}] \leftarrow \min_{x \in C(v)}\{c_x[\mathsf{Bnn}]\}$
$c_v[\mathsf{Wnb}] \leftarrow \min_{x \in C(v)}\{c_x[\mathsf{Rnn}]\}$
$c_v[\mathsf{Wnw}] \leftarrow \min_{x \in C(v)}\{c_x[\mathsf{Wnn}], c_x[\mathsf{Wnw}]\}$

DE-BEST-AT-ROOT :
return $\min\{c_r[\mathsf{Rnn}], c_r[\mathsf{Bnn}], c_r[\mathsf{Wnn}], c_r[\mathsf{Wnw}]\}$
```

Figure 5: Computing $de(T)$ for a tree

4. NP-Completeness Results

The three subsections in this section contain NP-completeness results for three variants of domination equivalence: ODE, ode, and de. Since DE is determined by $\gamma(G)$, it is NP-complete even when restricted to bipartite graphs.

4.1 Maximum open domination equivalence

We prove that the following problem is NP-complete for bipartite graphs.

Maximum Open Domination Equivalence (MAX-ODE)
Instance: A graph $G = (V, E)$ and an integer k.
Question: Is there an ODE-pair $\{R, B\}$ for G such that $|R| \geq k$?

We will need the intractability of 3SAT [2]:

3-satisfiability (3SAT)
Instance: A collection C of 3-element clauses on a set U of variables.
Question: Is there a truth assignment for U that satisfies all the clauses in C?

Theorem 4.1. MAX-ODE *is NP-complete.*

Proof. It is easy to see that MAX-ODE $\in NP$, since a nondeterministic algorithm need only guess disjoint subsets R and B of V and check that $N(R) = N(B)$ and $\max\{|R|, |B|\} \geq k$.

The reduction is from 3SAT. Let (U, C) be an instance of 3SAT. We construct a graph G and integer k such that $ODE(G) \geq k$ if and only if C is satisfiable.

For each variable $u_i \in U$ there is a ***truth-setting component*** T_i that is a P_3-path $u_i - m_i - \overline{u_i}$. We will call m_i a ***middle vertex*** and the vertices u_i and $\overline{u_i}$ ***literal vertices***. No other edge will be incident with m_i; thus, if either literal vertex is colored in an ode-pair, the other must receive the other color.

For each clause $c_j \in C$, there is a ***satisfaction-testing component*** S_j that is a P_2-path $c_j - l_j$. We will call each l_j a ***leaf vertex*** and each c_j a ***clause vertex***. The leaf vertex l_j will be a leaf in the graph; thus by Lemma 2.8 no clause vertex is colored.

The graph G is completed by adding ***communicating edges*** between each clause vertex and the literal vertices that correspond to the three literals in the clause. That is, for clause $c_j = (x_j, y_j, z_j)$, where each member of c_j is either a variable or its complement, the communicating edges are c_jx_j, c_jy_j, and c_jz_j.

The instance of MAX-ODE is completed by setting $k = n+m$. An example of the construction is given in Figure 6. It is easy to see how the construction can be accomplished in time polynomial in the size of the 3SAT instance. All that remains is to show that C is satisfiable if and only if G has an open dominating equivalent set R of size k or more.

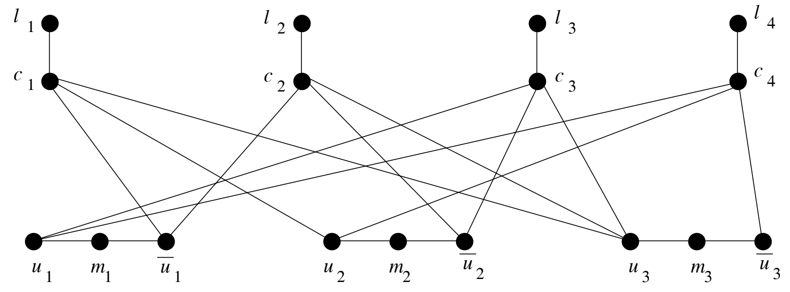

Figure 6: Graph for 3SAT instance $C = \{(\overline{u_1}, u_2, u_3), (\overline{u_1}, \overline{u_2}, u_3), (u_1, \overline{u_2}, u_3), (u_1, u_2, \overline{u_3})\}$.

First, consider a satisfying truth assignment t. Then an ode-pair $\{R, B\}$ is as follows. For each S_j, put the leaf $l_j \in R$. For each variable u_i for which $t(u_i) = \text{TRUE}$, put the literal vertex $u_i \in B$ and the literal vertex $\overline{u_i} \in R$, and for each variable u_i for which $t(u_i) = \text{FALSE}$, put the literal vertex $\overline{u_i} \in B$ and the literal vertex $u_i \in R$. Clearly $|R| = n + m$.

We claim that

$$N(R) = N(B) = Z = \{\, c_j : c_j \in C \,\} \cup \{\, m_i : u_i \in U \,\}.$$

Since all leaf vertices are in R, and one literal vertex from each truth-setting component is in R, we have $N(R) = Z$. It is also easy to see that $\{\, m_i : u_i \in U \,\} \subseteq N(B)$. Furthermore, since t

ensures that each clause has at least one TRUE literal in it and the vertices corresponding to TRUE literals are in B, $\{c_j : c_j \in C\} \subseteq N(B)$. Since only the clause vertices and middle vertices are adjacent to a colored vertex, it follows that $N(B) = N(R) = Z$, and $\{R, B\}$ is an ODE-pair for G.

Conversely, consider an ODE-pair $\{R, B\}$ with $|R| \geq n + m$. Let x_i be a literal vertex from T_i. Note that the neighbors of x_i are the middle vertex m_i and some clause vertices. Thus, since no clause vertex is colored, the vertex m_i is also not colored, as x_i cannot have only one colored neighbor. Thus, the only vertices that can be colored are the literal and leaf vertices.

As observed earlier, at most one literal vertex from each T_i can be in R. Thus, for $|R| \geq k = n + m$, all leaf vertices are in R. This means that each clause vertex is adjacent to a red vertex, and hence must be adjacent to at least one blue literal vertex. But no more than one literal vertex in each T_i can be blue. Thus, we can define $t(x_i) = \text{TRUE}$ for each literal vertex $x_i \in B$, giving a satisfying truth assignment. □

Corollary 4.2. MAX-ODE *is NP-complete even when restricted to bipartite graphs.*

4.2 Minimum open domination equivalence

In this section we prove that the following problem is NP-complete for bipartite graphs.

Minimum Open Domination Equivalence (MIN-ODE)
Instance: A graph $G = (V, E)$ and an integer $k \geq 1$.
Question: Is there an ODE-pair $\{R, B\}$ such that $1 \leq |R| \leq k$?

The proof is similar to that of Theorem 4.1. However, because we want to minimize $|R|$, we no longer use leaf vertices to prevent vertices from being colored. Instead, we use a cycle widget. So, we define a ***v-k-preclusion cycle*** to be a lariat C_{2k+3} containing v, where v is the only vertex in the cycle of degree greater than 2. By Lemma 2.4, if any vertex in a v-k-preclusion cycle is colored, then at least $k+1$ vertices in the v-k-preclusion cycle are in R.

We will need the intractability of NAE-3SAT [2]:

not-all-equal 3-satisfiability (NAE-3SAT)
Instance: A collection C of 3-element clauses over a set U of variables.
Question: Is there a truth assignment for U such that each clause in C has at least one TRUE and one FALSE literal?

We are now ready to prove the main result of this section.

Theorem 4.3. MIN-ODE *is NP-complete.*

Proof. It is easy to see that MIN-ODE $\in NP$, since a nondeterministic algorithm need only guess disjoint subsets R and B of V and check that $N(R) = N(B)$ and $k \geq \min\{|R|, |B|\} \geq 1$.

The reduction is from NAE-3SAT. Let (U, C) be an instance of NAE-3SAT. We construct a graph G and integer k such that $ode(G) \leq k$ if and only if there is a truth assignment such that each clause has at least one TRUE and one FALSE literal.

Define $k = n = |U|$. Then for each clause c_j, the satisfaction-testing component S_j is a clause vertex c_j in a c_j-k-preclusion cycle. For each variable $u_i \in U$, the truth-setting component T_i contains literal vertices u_i and $\overline{u_i}$, and an m_i-k-preclusion cycle where vertex m_i is adjacent to both u_i and $\overline{u_i}$. The communicating edges connect each c_j with the three corresponding literal vertices as before.

Last, we connect together the T_j as follows. For each literal vertex u_i, add a w_i-k-preclusion cycle with edges u_iw_i, w_iu_{i+1}, and $w_i\overline{u_{i+1}}$ connecting w_i to T_i and T_{i+1} (arithmetic modulo n).

An example of the construction is given in Figure 7. We conclude by showing that there is a solution to the instance of NAE-3SAT if and only if $ode(G) \leq k$.

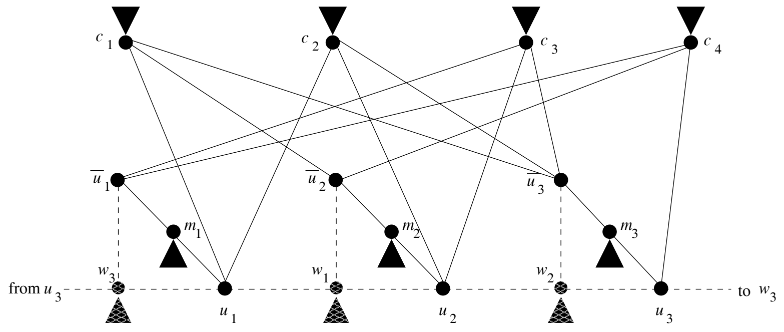

Figure 7: Graph for the instance $C = \{(u_1, \overline{u_2}, \overline{u_3}), (u_1, u_2, u_3), (\overline{u_1}, u_2, \overline{u_3}), (\overline{u_1}, \overline{u_2}, u_3)\}$ of NAE-3SAT. (Each filled triangle adjacent to a vertex v represents a v-3-preclusion cycle, where v is the vertex adjacent to the triangle.)

First, let t be a solution to NAE-3SAT. Then we construct an ODE-pair (R, B) as follows. For each TRUE variable u_i, put the literal vertex $u_i \in B$ and the literal vertex $\overline{u_i} \in R$, and for each FALSE variable u_i, put the literal vertex $\overline{u_i} \in B$ and the literal vertex $u_i \in R$. Clearly $|R| = n$.

We claim that

$$N(R) = N(B) = Z = \{\, c_j : c_j \in C \,\} \cup \{\, m_i : u_i \in U \,\} \cup \{\, w_i : u_i \in U \,\}.$$

Note that Z contains all vertices adjacent to $B \cup R$. Thus, we need only show that each vertex in Z is adjacent to at least one vertex in both R and B. Consider first a clause vertex c_j. Since there is at least one TRUE literal and at least one FALSE literal in the clause c_j, at least one of the literal vertices adjacent to c_j must be in R, and at least one must be in B. Now consider a middle vertex m_i. It is adjacent to both u_i and $\overline{u_i}$ and thus is both red- and blue-dominated. Similarly, each w_i is adjacent to both a literal vertex and its complement.

Conversely, consider an ODE-pair $\{R, B\}$ with $|R| \leq k$. Note that the only vertices that are not on a v-k-preclusion cycle are the literal vertices. It follows then, from Lemma 2.4, that only the literal vertices can be colored. Moreover, since $|R| \geq 1$, there must be at least one literal vertex in R. Note that then its complement must be in B. So let u_i and $\overline{u_i}$ be colored; then w_i is dominated. Since the only other neighbors of w_i are u_{i+1} and $\overline{u_{i+1}}$, one of these must be colored. This argument can be applied iteratively to show that all literal vertices are colored.

Now, each clause vertex is adjacent to three literal vertices, each of which is colored. It follows that a clause vertex must be adjacent to both a member of R and a member of B. Assigning a value of TRUE to the literals in the NAE-3SAT instance corresponding to the literal vertices in B provides a solution to NAE-3SAT. □

Corollary 4.4. MIN-ODE *is NP-complete even when restricted to bipartite graphs.*

4.3 Minimum domination equivalence

In this section we prove that the following problem is NP-complete.

Minimum Domination Equivalence (MIN-DE)
Instance: A graph $G = (V, E)$ and an integer k.
Question: Is there a DE-pair $\{R, B\}$ such that $|R| \leq k$?

Theorem 4.5. MIN-DE *is NP-complete.*

Proof. The proof closely follows the proof of Theorem 4.3. The only difference is that the constructed graph G includes the edge $(u_i, \overline{u_i})$ in each truth-setting component. □

Note that adding the edges between literal vertices makes it so that the constructed graph G is not bipartite.

5. Concluding Remarks

One can generalize the problem to having k sets with the same neighborhoods. For trees, a DE-triple never exists, but one can determine all parameters about ODE-triples etc. by using an immediate generalization of Observation 2.6.

Also, one can easily adapt our dynamic programming algorithms on trees so that they compute optimal sets when the goal is to minimize or maximize the number of vertices that are dominated, rather than the number of vertices that are doing the dominating. McRae [7] has shown that in general minimizing the number of dominated neighbors for a DE- or ODE-pair is NP-complete.

Acknowledgments

We would like to thank Edward Doyle and David Jacobs (Clemson University) and Alice McRae (Appalachian State University) for several helpful comments.

References

[1] R. B. Borie, R.G. Parker and C.A. Tovey, Deterministic Decomposition of Recursive Graph Classes, *SIAM J. Discrete Math.,* **4** (1991), 481–501.

[2] M. R. Garey and D.S. Johnson, *Computers and Intractability: A guide to the theory of NP-completeness*, Freeman, 1979.

[3] P. Heggernes and J.A. Telle, Partitioning graphs into generalized dominating sets, *Nordic J. Comput.,* **5** (1998), 128–142.

[4] D. P. Jacobs and R.E. Jamison, A note on equal unions in families of sets, *Discrete Math.,* **241** (2001), 387–393.

[5] B. Lindström, A theorem on families of sets, *J. Combin Theory Ser. A,* **13** (1972), 274–277.

[6] W. McCuaig, Brace generation, *J. Graph Theory,* **38** (2001), 124–169.

[7] A. McRae, Personal communication, 2001.

[8] O. Ore, *Theory of Graphs*, American Math Society, 1962.

[9] C. Payan and N.H. Xuong, Domination-balanced graphs, *J. Graph Theory,* **6** (1982), 23–32.

[10] N. Robertson, P.D. Seymour and R. Thomas, Permanents, Pfaffian orientations, and even directed circuits, *Ann. of Math.,* **150**(2)(1999), 929–975.

[11] H. Tverberg, *On equal unions of sets*, in Studies in Pure Mathematics, L. Mirsky (Ed.), Academic Press, 1971, 249–250.

[12] T. V. Wimer, S.T. Hedetniemi and R. Laskar, A methodology for constructing linear graph algorithms, *Congr. Numer.,* **50** (1985), 43–60.