AUSTRALASIAN JOURNAL OF COMBINATORICS
Volume **42** (2008), Pages 201–209

Trees with domination subdivision number one

S. Benecke[*] C. M. Mynhardt[†]

Department of Mathematics and Statistics
University of Victoria
P. O. Box 3060 STN CSC
Victoria, BC V8W 3R4
Canada
`stephen@math.uvic.ca` `mynhardt@math.uvic.ca`

Abstract

The domination subdivision number $\mathrm{sd}_\gamma(G)$ of a graph G is the minimum number of edges that must be subdivided to increase the domination number of G. We present a simple characterization of trees with $\mathrm{sd}_\gamma = 1$ and a fast algorithm to determine whether a tree has this property.

1 Introduction

For a given graph parameter π we define the *π-subdivision number* sd_π to be the minimum number of edges that must be subdivided, where each edge may be subdivided at most once, to change π. The type of change required (increase or decrease) depends on π. For example, the subdivision of edges of a graph never increases its chromatic or clique number, so we would require the change to be a decrease. Domination type parameters, on the other hand, such as the domination number γ, total domination number γ_t, connected domination number γ_c and independence number β generally increase when an edge is subdivided.

The domination subdivision number sd_γ was introduced by Velammel [12] and since then also studied in [1, 2, 3, 6, 7, 9]. Results on total domination subdivision numbers and connected domination numbers appear in [4] and [5], respectively, and references to other work on these topics may be found there.

Of interest to us are Velammel's bounds [12] on the domination subdivision number of trees, namely that $1 \le \mathrm{sd}_\gamma(T) \le 3$ for any tree T, and the characterizations by Karami and Sheikholeslami [9], and Aram, Favaron and Sheikholeslami [1], respectively, of trees which achieve the lower and upper bound. Thus it seems that the partition of the class of trees into three subclasses according to the domination

[*] Supported by the Skye Foundation and the National Research Foundation of South Africa.
[†] Supported by the Natural Sciences and Engineering Research Council of Canada.

subdivision number has been achieved. However, the characterization of trees with $\mathrm{sd}_\gamma = 1$ given in [9] is complex and involves concepts which themselves have not been characterized.

We present a simple characterization of trees with domination subdivision number one in terms of vertices which do not appear in any minimum dominating set of the tree. These vertices were characterized by Mynhardt in [11]. We also give a fast algorithm to determine whether a given tree has domination subdivision number one.

2 Definitions and background

We generally follow the notation and terminology of [8]. As usual $\gamma(G)$ denotes the domination number of G. The set $S \subseteq V(G)$ is called a *γ-set* if it is a dominating set with $|S| = \gamma(G)$. The set $\mathcal{N}(G)$ consists of those vertices which are not contained in any γ-set of G. For $A, B \subseteq V(G)$, we abbreviate "A dominates B" to "$A \succ B$"; if $B = V(G)$ we write $A \succ G$ and if $B = \{b\}$ we write $A \succ b$. Further, $N(v) = \{u \in V(G) : uv \in E(G)\}$ and $N[v] = N(v) \cup \{v\}$ denote the *open* and *closed neighbourhoods*, respectively, of a vertex v of G. The *closed neighbourhood of a set* $S \subseteq V(G)$, denoted by $N[S]$, is the set $\cup_{s \in S} N[s]$. If $s \in S$, then the *private neighbourhood of* s relative to S, denoted by $\mathrm{pn}(s, S)$, is the set $N[s] - N[S - \{s\}]$.

A *support vertex* of a tree is a vertex adjacent to a leaf. For any leaf ℓ we denote the support vertex adjacent to ℓ by s_ℓ, and for any vertex v of T we denote a pendant edge added at v by $\overset{\circ}{\overrightarrow{vw}}$. Let $\mathcal{L}(T)$ denote the set of leaves of a tree T and define $\mathcal{L}'(T) = \{v \in V(T) : T + \overset{\circ}{\overrightarrow{vw}}$ has a γ-set set containing $w\}$. In [9] the authors define the sets

$$\begin{aligned}
V_0(T) &= \{v \in V(T) : \gamma(T - v) = \gamma(T)\}, \\
V_+(T) &= \{v \in V(T) : \gamma(T - v) > \gamma(T)\}, \\
V_-(T) &= \{v \in V(T) : \gamma(T - v) < \gamma(T)\}, \qquad (1)\\
W_0(T) &= \{v \in V(T) : \gamma(T + \overset{\circ}{\overrightarrow{vw}}) = \gamma(T)\}, \\
W_+(T) &= \{v \in V(T) : \gamma(T + \overset{\circ}{\overrightarrow{vw}}) > \gamma(T)\}.
\end{aligned}$$

They then define the collection $\mathcal{B}$ of trees as follows. Let T_i, $i = 1, 2$, be two trees, one of which has order at least two, and $u_i \in V(T_i)$. The collection $\mathcal{B}$ consists of all trees T of order at least three such that each $T \in \mathcal{B}$ satisfies one of the following properties:

P1. $T = T_1 \cup T_2 \cup \{u_1u_2\}$, where $u_i \in (V_0(T_i) \cup V_+(T_i)) \cap W_+(T_i)$, $i = 1, 2$;

P2. $T = T_1 \cup T_2 \cup \{u_1u_2\}$, where $u_1 \in W_0(T_1) - \mathcal{L}'(T_1)$ and $u_2 \in V_-(T_2)$;

P3. there exists a vertex $u \in \mathcal{L}(T) \cap \mathcal{N}(T)$.

These definitions lead to a characterization of trees with $\mathrm{sd}_\gamma = 1$.

Theorem 1 [9] *For a tree T of order $n \geq 3$, $\mathrm{sd}_\gamma(T) = 1$ if and only if $T \in \mathcal{B}$.*

This characterization uses the sets in (1) which are, however, not characterized. Property **P3** provides a clue to a simpler characterization in terms of the set $\mathcal{N}(T)$, which was characterized in [11].

3 The characterization

Theorem 2 *For a tree T of order $n \geq 3$, $\mathrm{sd}_\gamma(T) = 1$ if and only if T has (i) a leaf $u \in \mathcal{N}(T)$ or (ii) an edge xy with $x, y \in \mathcal{N}(T)$.*

Proof. Assume (i) holds, let T' be the tree obtained by joining a new vertex w to u, and consider any γ-set D' of T'. Since $w \in \mathcal{L}(T')$ and u is the support vertex s_w, $|\{u, w\} \cap D'| = 1$. Let $D = (D' - \{u, w\}) \cup \{u\}$ and note that $|D| = |D'|$. Then D is a dominating set of T containing u, hence by hypothesis, $|D| > \gamma(T)$. But T' is isomorphic to the tree obtained by subdividing the edge uv of T and it follows that $\mathrm{sd}_\gamma(T) = 1$.

Assume (ii) holds, let T' be the tree obtained by subdividing xy with the vertex w and consider any γ-set D' of T'. Since $D' \succ w$, $D' \cap \{x, y, w\} \neq \phi$, and since D' is a minimal dominating set of T', $|D' \cap \{x, y, w\}| \leq 2$. If $w \notin D'$, then D' is a dominating set of T containing x or y and by hypothesis $|D'| > \gamma(T)$, so we are done. Hence assume $w \in D'$ and, without loss of generality, $x \notin D'$. Then $D = (D' - \{w\}) \cup \{x\}$ is a dominating set of T containing x, so as before, $|D| = |D'| > \gamma(T)$. Therefore $\mathrm{sd}_\gamma(T) = 1$.

Conversely, suppose $\mathrm{sd}_\gamma(T) = 1$ and let uv be an edge such that the tree T' obtained by subdividing uv with the vertex w satisfies $\gamma(T') = \gamma(T) + 1$. If $u \in \mathcal{L}(T)$, then $u \in \mathcal{N}(T)$, because if there exists a γ-set D of T with $u \in D$, then $D' = (D - \{u\}) \cup \{w\} \succ T'$ and $|D'| = \gamma(T)$, a contradiction. Thus in this case (i) holds and we are done. We may therefore assume that $\{u, v\} \cap \mathcal{L}(T) = \phi$.

If $\{u, v\} \subseteq \mathcal{N}(T)$, then (ii) holds with $xy = uv$ and we are done, so suppose $\{u, v\} \nsubseteq \mathcal{N}(T)$. If there exists a γ-set D of T with $u, v \in D$, then $D \succ T'$, a contradiction. Thus assume D to be a γ-set of T with (without loss of generality) $\{u, v\} \cap D = \{u\}$. We first show that

$$v \in \mathrm{pn}(u, D). \tag{2}$$

If $u' \in (N(v) \cap D) - \{u\}$, then in T', $u \succ w$, $u' \succ v$, and all other vertices in T' are dominated by the same vertices in D that dominate them in T. Then $D \succ T'$, a contradiction.

Let $N(v) - \{u\} = \{w_1, ..., w_t\}$, where $t \geq 1$ because v is not a leaf, let T_i be the subtree of $T - vw_i$ containing w_i and $D_i = D \cap V(T_i)$, $i = 1, ..., t$. Since $v \notin D$, $D_i \succ T_i$ and thus D_i is a γ-set of T_i, for otherwise we can obtain a dominating set D^* of T with $|D^*| < |D|$. Also,

$$w_i \in \mathcal{N}(T_i) \text{ for each } i = 1, ..., t, \tag{3}$$

for otherwise we can obtain a γ-set S of T with $u, w_i \in S$ for some i and $v \notin S$, which would contradict (2) with S instead of D.

Let T_v be the subtree of $T - \{vw_i : i = 1, ..., t\}$ containing v, and $D_v = D \cap V(T_v)$. Then $D_v \succ T_v$ and hence, as in the case of D_i and T_i, D_v is a γ-set of T_v. Moreover,

$$v \in \mathcal{N}(T_v), \tag{4}$$

for if S_v is a γ-set of T_v with $v \in S_v$, then $D' = (S_v - \{v\}) \cup \{w\} \cup \bigcup_{i=1}^{t} D_i$ dominates T' and $|D'| = |D|$, contradicting $\gamma(T') = \gamma(T) + 1$. Now we show that

$$v \in \mathcal{N}(T). \tag{5}$$

Suppose that S is a γ-set of T containing v, $S_v = S \cap V(T_v)$ and $S_i = S \cap V(T_i)$, $i = 1, ..., t$. Since $v \in S_v$, $S_v \succ T_v$. By (4), S_v is not a γ-set of T_v, therefore $|S_v| > |D_v|$ and so $|S_i| < |D_i|$ for some i, without loss of generality say $i = 1$. Now $S_1 \not\succ T_1$ and, since w_1 is the only vertex of T_1 dominated by any vertex in $S - S_1$, it follows that $S_1 \succ T_1 - w_1$. But this implies that $S_1 \cup \{w_1\}$ is a γ-set of T_1 containing w_1, contradicting (3). Thus (5) holds. Finally, we show that

$$w_i \in \mathcal{N}(T) \text{ for each } i = 1, ..., t. \tag{6}$$

Suppose that X is a γ-set of T containing w_i (i fixed), $X_v = X \cap V(T_v)$ and $X_i = X \cap V(T_i)$. Then $X_i \succ T_i$ and $w_i \in X_i$, so by (3), $|X_i| > \gamma(T_i)$. But then either $|X_j| < \gamma(T_j)$ for some $j \neq i$ and we obtain a contradiction as above, or $|X_v| < \gamma(T_v)$ and $X_v \succ T_v - v$, so that $X_v \cup \{v\}$ is a γ-set of T_v containing v, contradicting (4) and proving (6).

By (5) and (6), for any $i = 1, ..., t$, vw_i is an edge of T with $v, w_i \in \mathcal{N}(T)$ and therefore (ii) holds. ■

4 The algorithm

We now present an algorithm to determine whether a given tree has domination subdivision number one. The set $\mathcal{N}(T) \subset V(T)$ plays a crucial role in the characterization in Theorem 2. In [11] Mynhardt provided a characterization of $\mathcal{N}(T)$, and our algorithm is based on the latter characterization.

Given a tree T and vertex $v \in V(T)$, we wish to decide whether or not $v \in \mathcal{N}(T)$. Let T_v denote the tree T rooted at v, $C(u)$ the set of all children of $u \in V(T)$ and T_u the subtree rooted at u containing u and all its descendents. A path P in T is said to be a $u - L$ path if P joins u to a leaf of T_v. Denote the length of P by $l(P)$ and, for $j = 0, 1, 2$, let

$$C^j = \{x \in C(u) : T_x \text{ contains an } x - L \text{ path } P \text{ with } l(P) \equiv j \pmod{3}\}.$$

We emphasize the tree T_v by writing $C^j_{T_v}(u)$ if necessary.

The characterization of $\mathcal{N}(T)$ involves a pruning of T_v to a tree $\overline{T}_v$ as follows. Let u be a branch vertex at maximum distance from v and assign a *priority* to each

$w \in C(u)$, where $w^0 \in C^0(u)$ has a higher priority than $w^1 \in C^1(u)$, which has a higher priority than $w^2 \in C^2(u)$. If $z \in C(u)$ has highest priority among the vertices in $C(u)$, delete each $w \in C(u) - \{z\}$ and all its descendents. In the resulting tree u has degree 2. This pruning process is repeated until no branch vertex except possibly u remains, leaving a tree $\overline{T}_v$. For ease of notation let $\overline{C}^j(v)$ denote $C^j_{\overline{T}_v}(v)$ for $j = 0, 1, 2$.

Theorem 3 [11] *For any tree T and any vertex v of T, $v \in \mathcal{N}(T)$ if and only if $\overline{C}^0(v) = \emptyset$ and $\overline{C}^1(v) \neq \emptyset$.*

We represent T by its adjacency list *TAL* and T_v by its adjacency list *TvAL*. The main algorithm, called FINDN(), takes a tree T (or rather its adjacency list) as input and produces the set $\mathcal{N}(T)$ as a list of vertices. It calls the subalgorithm CONSTRUCTTV(), to produce the tree T rooted at v, as well as the subalgorithm PRUNING(), that essentially does the pruning procedure described above. Lastly, CHECKC() returns true if the vertex v is in $\mathcal{N}(T)$ and false otherwise. If $v \in \mathcal{N}(T)$, it is added to a list *NList*, which will eventually represent the whole set $\mathcal{N}(T)$.

We follow the algorithm format used in [10]. Throughout the following algorithms, external functions are used in an intuitive manner. These functions are named so that their use is clear.

```
Algorithm 4.1: FINDN(TAL)

  comment: { TAL is the adjacency list for a tree T
           { and n is the order of the tree.

  external EMPTYLIST(), ADDTOLIST()

  NList ← EMPTYLIST()
  for v ← 1 to n
         { (TvAL, ScanQ) ← CONSTRUCTTV(TAL, v)
    do   { label ← PRUNING(TvAL, ScanQ)
         { if CHECKC(TvAL, label, v)
         {   then ADDTOLIST(NList, v)
  return (NList)
```

The algorithm CONSTRUCTTV() takes the adjacency list of T and the vertex v as input, and constructs the tree rooted at v by way of a breadth-first search (BFS) through the vertices of T. Since we are only concerned with a tree as input graph, the adjacency list for T_v is obtained by simply removing the parent of u from its (out-) neighbourhood, for every u in T. Also, the order in which the vertices are searched is saved in *ScanQ*, to be used in the PRUNING() algorithm.

```
Algorithm 4.2: CONSTRUCTTV($TAL, v$)

comment: { $TAL$ is the adjacency list for a tree $T$,
         { while $v$ is a vertex of $T$.

external REMOVE()

$TvAL \leftarrow TAL$
$ScanQ(1) \leftarrow v$
$QSize \leftarrow 1$
$k \leftarrow 1$
repeat
 $x \leftarrow ScanQ(k)$
 for $w \in TvAL(x)$
    do { $QSize \leftarrow QSize + 1$
       { $ScanQ(QSize) \leftarrow w$
       { $TvAL(w) \leftarrow$ REMOVE($TAL(w), x$)
 $k \leftarrow k + 1$
until $k > QSize$
return ($TvAL, ScanQ$)
```

```
Algorithm 4.3: PRUNING($TvAL, ScanQ$)

comment: { $TvAL$ is the adjacency list for a tree $T_v$ of order $n$,
         { while $ScanQ$ is a BFS ordered list of $V(T_v)$
         { with root $v$ as first vertex.

external ISEMPTY(), MOD(), MIN()

for $i \leftarrow n$ downto 2
   do { $x \leftarrow ScanQ(i)$
      { if ISEMPTY($TvAL(x)$)
      {   then $label(x) \leftarrow -1$
      {   else { $tmplist \leftarrow$ MOD($label(TvAL(x)) + 1, 3$)
      {        { $label(x) \leftarrow$ MIN($tmplist$)
return ($label$)
```

```
Algorithm 4.4: CheckC($TvAL$, $label$, $v$)

comment: { $TvAL$ is the adjacency list for $T_v$,
         { a tree rooted at vertex $v$,
         { while $label$ is a list of vertex labels.

external Mod()

$C0Empty \leftarrow$ true
$C1Empty \leftarrow$ true
for $u \in TvAL(v)$
    do { $a \leftarrow$ Mod$(label(u) + 1, 3)$
       { if $a = 0$
       {   then { $C0Empty \leftarrow$ false
       {        { exit for
       {   else if $a = 1$ and $C1Empty$
       {   then $C1Empty \leftarrow$ false
return ($C0Empty$ and not $C1Empty$)
```

The pruning procedure may be seen as a labelling of $V(T) - \{v\}$ with labels $-1, 0, 1, 2$. For the purposes of determining $\mathcal{N}(T)$, we do not need to work with the tree $\overline{T}_v$. Only the sets $\overline{C}^j(v)$, $j = 0, 1, 2$, are important. Thus, the algorithm Pruning() takes $TvAL$ (the adjacency list of T_v) as input and labels the vertices of T_v in a reversed BFS order up to the root vertex v. All leaves receive a label of -1, since they have no children in T_v. Any vertex x that is not a leaf, with children $c_1, c_2, \ldots, c_k$ in T_v (and labels $l_1, l_2, \ldots, l_k$) receives the label $label(x) = \min_{1 \le i \le k} l_i + 1 \pmod 3$.

This labelling captures the idea behind the pruning process without actually removing vertices from T_v (and thus without obtaining $\overline{T}_v$). Note that the label of v has no significance and is never used. By assigning labels to the vertices of T_v through a reversed BFS order, the next branch vertex considered will always be one furthest away from v. Also, we do not need to know exactly which vertices the sets $\overline{C}^j(v)$ contain – only whether $\overline{C}^0(v) = \emptyset$ and $\overline{C}^1(v) \neq \emptyset$, these properties being the criteria for v to be in the set $\mathcal{N}(T)$.

Finally, CheckC() takes the adjacency list $TvAL$ of T_v, as well as the labelling *label* as input, and returns true if $\overline{C}^0(v) = \emptyset$ and $\overline{C}^1(v) \neq \emptyset$.

The time complexity of the algorithm FindN() is easily verified to be $\mathcal{O}(n^2)$. The number of operations for the subalgorithm ConstructTv(), based on a BFS search through the vertices of the tree T, depends on the sum of the degrees of the vertices of T. Since the input for the algorithm is always a tree, this subalgorithm has complexity $\mathcal{O}(n)$, where n is the order of T. For both the subalgorithms Pruning() and CheckC(), the number of **for**-loops are bounded by a constant multiple of n, and thus each of these algorithms have complexity $\mathcal{O}(n)$ as well. These subalgorithms are all executed consecutively for each **for**-loop in FindN(). Since the main **for**-loop in FindN() repeats n times, a complexity of $\mathcal{O}(n^2)$ is obtained.

The algorithm FindN() needs only be adjusted slightly to determine whether or not a given tree T has domination subdivision number one. One could of course determine $\mathcal{N}(T)$ first and then check to see if the set contains a leaf or two adjacent vertices. However, it may be more efficient to stop the algorithm as soon as an appropriate vertex (or pair of vertices) is found to be in $\mathcal{N}(T)$. The algorithm IsSDOne() takes the adjacency list of T as input (same as in the case of FindN()) and returns true if $\mathrm{sd}_\gamma(T) = 1$.

```
Algorithm 4.5: IsSDOne(TAL)

comment: TAL is the adjacency list for a tree T of order n.
external EmptyList(), AddToList(), Intersection(), Length()

SDOne ← false
NList ← EmptyList()
for v ← 1 to n
  do
    (TvAL, ScanQ) ← ConstructTv(TAL, v)
    label ← Pruning(TvAL, ScanQ)
    if CheckC(TvAL, label, v)
      then
        AddToList(NList, v)
        tmp ← Intersection(TAL(v), NList)
        if Length(TAL(v)) = 1 or not IsEmpty(tmp)
          then
            SDOne ← true
            exit for
return (SDOne)
```

References

[1] H. Aram, O. Favaron and S. M. Sheikholeslami, Trees with domination subdivision number three, preprint.

[2] A. Bhattacharya and G. R. Vijayakumar, Effect of edge-subdivision on vertex-domination in a graph, *Discuss. Math. Graph Theory* **22** (2002), 335-347.

[3] O. Favaron, T. W. Haynes and S. T. Hedetniemi, Domination subdivision numbers in graphs, *Util. Math.* **66** (2004), 195-209.

[4] O. Favaron, H. Karami and S. M. Sheikholeslami, Total domination and total domination subdivision numbers, *Australas. J. Combin.* **38** (2007), 229-235.

[5] O. Favaron, H. Karami and S. M. Sheikholeslami, Connected domination subdivision numbers of graphs, *Util. Math.*, to appear.

[6] T. W. Haynes, S. M. Hedetniemi and S. T. Hedetniemi, Domination and independence subdivision numbers of graphs, *Discuss. Math. Graph Theory* **20** (2000), 271-280.

[7] T. W. Haynes, S. M. Hedetniemi, S. T. Hedetniemi, D. P. Jacobs, J. Knisely and L. C. van der Merwe, Domination subdivision numbers, *Discuss. Math. Graph Theory* **21** (2001), 239–253.

[8] T. W. Haynes, S. T. Hedetniemi and P. J. Slater, *Fundamentals of Domination in Graphs* (Marcel Dekker, New York, 1998).

[9] H. Karami and S. M. Sheikholeslami, Trees whose domination subdivision number is one, *Australas. J. Combin.* 40 (2008), 161–166.

[10] W. Kocay and D. L. Kreher, *Graphs, Algorithms, and Optimization*, Chapman & Hall/CRC, Boca Raton, Florida, 2005.

[11] C. M. Mynhardt, Vertices contained in every minimum dominating set of a tree, *J. Graph Theory* **31** (1999), 163–177.

[12] S. Velammel, *Studies in Graph Theory: Covering, Independence, Domination and Related Topics*, Ph.D. Thesis (Manonmaniam Sundaranar University), 1997.

(Received 20 Sep 2007)