

# On Domination and Reinforcement Numbers in Trees

Jean R. S. Blair\*    Wayne Goddard†    Stephen T. Hedetniemi†  
Steve Horton\*    Patrick Jones‡    Grzegorz Kubicki§

## Abstract

The reinforcement number of a graph is the smallest number of edges that have to be added to a graph to reduce the domination number. We introduce the  $k$ -reinforcement number of a graph as the smallest number of edges that have to be added to a graph to reduce the domination number by  $k$ . We present an  $O(k^2n)$  dynamic programming algorithm for computing the maximum number of vertices that can be dominated using  $\gamma(G) - k$  dominators for trees. A corollary of this is a linear-time algorithm for computing the  $k$ -reinforcement number of a tree. We also discuss extensions and related problems.

**Key Words:** Domination, Reinforcement, Trees

## 1 Introduction

A common question is how does the value of a parameter change as edges or vertices are added or removed. In this case the parameter is the domination number: the minimum number of vertices to dominate the graph. Domination-critical graphs, where any additional edge decreases the domination number, were introduced by Sumner and Blich [14]. We consider the best edges: how many

---

\*United States Military Academy, West Point, NY 10996 USA, [jean.blair@usma.edu](mailto:jean.blair@usma.edu), [steve.horton@usma.edu](mailto:steve.horton@usma.edu)

†Clemson University, Clemson, SC 29634 USA, [goddard@cs.clemson.edu](mailto:goddard@cs.clemson.edu), [hedet@cs.clemson.edu](mailto:hedet@cs.clemson.edu)

‡Vanderbilt University, Nashville TN 37240 USA [robomann@yahoo.com](mailto:robomann@yahoo.com)

§University of Louisville, Louisville, KY 40292 USA, [gkubicki@louisville.edu](mailto:gkubicki@louisville.edu)

edges must be added to reduce the domination number, or a related parameter. This concept was introduced by Kok and Mynhardt [11] and discussed in Chapter 17 in [8].

Let  $G = (V, E)$  be a graph. For  $v \in V$ , the *(open) neighborhood of  $v$*  is the set  $N(v) = \{u \in V \mid uv \in E\}$ . For a set  $S \subseteq V$ , the *open neighborhood of  $S$*  is  $N(S) = \bigcup_{v \in S} N(v)$  and the *closed neighborhood of  $S$*  is  $N[S] = N(S) \cup S$ . A *dominating set*  $S \subseteq V$  satisfies  $N[S] = V$ . The *domination number*  $\gamma(G)$  of a graph is the cardinality of a smallest dominating set. A dominating set  $S$  with  $|S| = \gamma(G)$  is known as a  $\gamma$ -set. Numerous variants of this problem appear in the literature [8, 7].

## 1.1 Reinforcement Number

Kok and Mynhardt [11] defined the *reinforcement number* of a graph  $r(G)$  as the minimum number of edges that have to be added to  $G$  so that the resulting graph  $G'$  satisfies  $\gamma(G') < \gamma(G)$ . (If  $\gamma(G) = 1$ , then they defined  $r(G) = 0$ .) One can readily generalize this idea. For example, the question for the fractional domination number (that is,  $r_f$ ) was considered by Domke and Laskar [3]. See also [2, 16].

A natural extension is to define the  *$k$ -reinforcement number* of a graph  $r^k(G)$  as the minimum number of edges that must be added to  $G$  so that  $\gamma(G') \leq \min\{\gamma(G) - k, 1\}$  for the resulting graph  $G'$ . The  $k$ -reinforcement problem applies to a variety of settings modeled by graphs where dominators have costs but where edges can be added to the graph (incurring less cost), eliminating the need for some of the dominators. For example, in a network it might be very expensive to set up a new mirror of a database, but relatively cheap to add a link.

It is easy to show that the  $k$ -reinforcement number is determined by how much domination  $\gamma(G) - k$  vertices can do in  $G$ . For ease of notation, we define  $d(\ell, G)$  as the maximum of  $|N[S]|$  taken over all subsets  $S$  with  $|S| \leq \ell$  (the inequality is to cater for the case where  $\ell$  exceeds the order). Now, consider a set  $F$  of augmenting edges and a  $\gamma$ -set  $S$  of  $G \oplus F$ , the graph formed by adding  $F$  to  $G$ . It is clear that  $|F| \geq |V - N_G[S]|$ . On the other hand, given any set  $S \subseteq V$  one can always choose an augmenting set  $F$  of cardinality  $|V - N_G[S]|$  such that  $S$  dominates  $G \oplus F$ . It follows that:

**Lemma 1.1** *For any graph  $G$  of order  $n$ ,  $r^k(G) = n - d(\gamma(G) - k, G)$ .*

This paper focuses on properties of the reinforcement number and its variants and the calculation of this for tree. Hsu [9] provided an algorithm to calculate  $d(\ell, T)$  on a tree  $T$  in  $O(\ell n^3)$  time. The problem of calculating  $d(\ell, G)$  is a special case of the maximum coverage problem. Megiddo, Zemel and Hakimi [12] gave an algorithm which calculates  $d(\ell, T)$  in  $O(\ell n^2)$  time (but actually solves a more general problem). However, we need the case where  $\ell$  is  $\gamma - 1$ . So we provide an algorithm for calculating  $d(\gamma - k, T)$  which runs in time  $O(k^2 n)$ . As an aside, we also provide an algorithm that improves on the previous algorithms to  $O(\ell^2 n)$ .

This paper is organized as follows. In the next section, we present an  $O(k^2 n)$  dynamic programming algorithm for computing the maximum number of vertices that can be dominated using  $\ell = \gamma(G) - k$  dominators for trees. Following, we extend some previous results for the reinforcement problem. We then use the algorithm to produce a linear-time algorithm for finding the reinforcement number of a tree. We conclude with some directions for further research. But first a comment on the hardness of the problem in general.

## 1.2 Intractability

Since it is a generalization of the NP-complete domination number problem, it is clear that the decision problem

MAXIMUM COVERAGE

*Instance:* graph  $G$  and integers  $\ell$  and  $s$

*Question:* Is  $d(\ell, G) \geq s$ ?

is NP-complete. For fixed  $\ell$  the problem is, of course, polynomial-time computable by considering all  $\binom{n}{\ell}$  subsets. However, it is unlikely that the problem is fixed-parameter tractable, since domination is believed hard (see [4] for definitions and discussion).

On the other hand, the reinforcement number problem is hard even for specific values of the parameters. For example, consider the decision problem:

REINFORCEMENT

Instance: graph  $G$

Question: Is  $r^1(G) \leq 1$ ?

**Lemma 1.2** *The problem REINFORCEMENT is NP-complete.*

**Proof.** The proof is a reduction from 3SAT and uses a minor modification of the standard construction for domination (see [6]). For a boolean formula  $\phi$ , we produce a graph  $G_\phi$  such that  $\phi$  is satisfiable exactly when  $r^1(G_\phi) \leq 1$ .

Suppose input  $\phi$  in conjunctive normal form has  $c$  clauses and a total of  $m$  variables. For each clause, create a vertex. For each variable  $v$ , create a  $K_4$  with one vertex labeled  $v$  and one labeled  $\bar{v}$ . Then for each clause, join the clause-vertex to the three vertices corresponding to the three literals that are in that clause. Finally, add a single vertex  $a$  and join it to all clause-vertices. The result is a graph  $G_\phi$ . For example, the graph for  $(x \vee y \vee z) \wedge (\bar{x} \vee y \vee \bar{z})$  is shown in Figure 1.

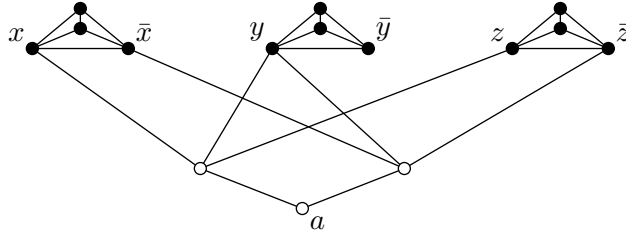


Figure 1: Example  $G_\phi$

Claim: the mapping  $\phi \mapsto G_\phi$  is the desired reduction. It is clear that  $\gamma(G_\phi) = m + 1$ . If  $\phi$  has a satisfying assignment, then let  $D$  be the set of the  $m$  vertices corresponding to the true literals in the assignment. Then each  $K_4$  is dominated and each clause-vertex is dominated. Only vertex  $a$  is undominated; thus  $r^1(G_\phi) \leq 1$ .

Conversely, suppose  $r^1(G_\phi) \leq 1$ . That is, there is a set  $D$  of size  $m$  that dominates all but one vertex. Then  $D$  must contain one vertex from each  $K_4$ , since it cannot miss two of the unlabeled vertices. It follows that  $D$  does not dominate  $a$  and so must dominate every clause-vertex. That is, if one sets all the literals corresponding to vertices in  $D$  to true, one has a satisfying assignment.

That is, we have shown that 3SAT reduces to REINFORCEMENT. ■

## 2 Maximum Coverage Algorithms for Trees

In this section we give an  $O(\ell^2 n)$ -time algorithm for computing the maximum number of vertices in a tree  $T$  that can be dominated by a set of  $\ell$  vertices. We then give another algorithm which runs in time  $O((\gamma - \ell)^2 n)$ . We present the  $O(\ell^2 n)$  algorithm first because it is simpler.

### 2.1 Postorder traversal of the edges

The algorithms use a standard dynamic-programming postorder-traversal approach, motivated by the methodology of Borie et al. [1] and Wimer (see for example [15]). This uses the fact that a rooted tree on  $n$  vertices can be built up from  $n$  trivial trees by the repeated process of adding an edge between the roots of two rooted trees and making one of these the root of the new tree. There is a set  $\Psi$  of parameters which is calculated for each subtree. The main detail of the algorithm is the choice of  $\Psi$ ; a recursive formula for  $\Psi$  on the combined tree in terms of  $\Psi$  of the two subtrees; and a formula for extracting the desired value from the set  $\Psi$  at the root.

Consider a rooted tree  $T$ . For any vertex  $v$ , we let  $C(v)$  denote the children of  $v$ , and let  $T_v$  denote the subtree consisting of  $v$  and its descendants. For efficient implementation, one uses a postorder traversal and processes the edges one at a time bottom up. At each vertex  $v$  the current value of  $\Psi$  is maintained. The invariant is that

*at each step and for every vertex  $v$ , the value of  $\Psi$  gives the correct values of the parameters for the descendant subtree rooted at  $v$  induced by the already-processed edges.*

Thus the overall algorithm is to (1) initialize  $\Psi$  for all vertices to the value of the trivial tree; (2) call the recursive procedure PROCESS on the root vertex; and (3) extract the desired value from  $\Psi$  at the root. The recursive code has the outline as follows.

```

PROCESS(vertex  $v$ ) {
    for each child  $c \in C(v)$  in turn {
        process(vertex  $c$ )
        apply formula to edge  $vc$ 
    }
}

```

The running time is dominated by the  $n - 1$  applications of the formula. (We maintain suitable child–parent pointers.)

## 2.2 Maximum coverage using $\ell$

The goal is to determine  $d(\ell, T)$ . The approach of [15] is to define a set of *properties* and create a parameter for each property. In this case, for each property  $\Pi$  and each  $i$  in the range  $0 \leq i \leq \ell$ , we are interested in the maximum number of dominated vertices in  $T_v$  using  $i$  vertices such that the set has property  $\Pi$ . To simplify the formulas, it is better to allow the set to be smaller: We define:

*$d_v(\Pi, i)$  is the maximum  $|N_{T_v}[S]|$  such that  $S \subseteq T_v$ ,  $|S| \leq i$  and  $S$  has property  $\Pi$ .*

It follows that  $d_v(\Pi, i)$  is nondecreasing as a function of  $i$ .

There are three properties—the three possibilities for the root vertex  $v$ :

Property  $\mathcal{I}$  (“in”) means  $v \in S$ .

Property  $\mathcal{D}$  (“dominated”) means  $v \notin S$  and  $\exists c \in C(v) \cap S$ .

Property  $\mathcal{U}$  (“undominated”) means  $v \notin S$  and  $C(v) \cap S = \emptyset$ .

For example, at least one of  $d_v(\mathcal{I}, \gamma(T_v))$  and  $d_v(\mathcal{D}, \gamma(T_v))$  is the order of the subtree.

For the trivial 1-vertex tree we have

$$d_v(\mathcal{I}, j) = 1 \text{ for } j \geq 1, \quad \text{and} \quad d_v(\mathcal{U}, j) = 0 \text{ for } j \geq 0.$$

All other possibilities are impossible; we initialize those values to  $-\infty$ .



---

**for**  $i \leftarrow \ell$  **down to** 0 **do**

$$1 \quad d_v(\mathcal{I}, i) \leftarrow \max_{0 \leq x \leq i} d_v(\mathcal{I}, x) + \max \begin{cases} d_c(\mathcal{I}, i - x), \\ d_c(\mathcal{D}, i - x), \\ d_c(\mathcal{U}, i - x) + 1 \end{cases}$$

$$2 \quad d_v(\mathcal{D}, i) \leftarrow \max_{0 \leq x \leq i} \max \begin{cases} d_v(\mathcal{D}, x) + \max \begin{cases} d_c(\mathcal{I}, i - x), \\ d_c(\mathcal{D}, i - x), \\ d_c(\mathcal{U}, i - x) \end{cases} \\ d_v(\mathcal{U}, x) + d_c(\mathcal{I}, i - x) + 1 \end{cases}$$

$$3 \quad d_v(\mathcal{U}, i) \leftarrow \max_{0 \leq x \leq i} d_v(\mathcal{U}, x) + \max \begin{cases} d_c(\mathcal{D}, i - x), \\ d_c(\mathcal{U}, i - x) \end{cases}$$

Figure 2: Processing edge  $vc$

---

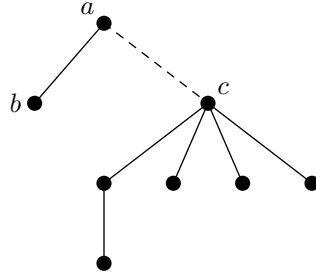


Figure 3: Example Tree  $T$

We now process the edge  $ac$  ( $a$  is the parent and  $c$  the child). The first computation performed is  $d_a(\mathcal{I}, 3)$ :

$$\begin{aligned} d_a(\mathcal{I}, 3) &\leftarrow \max_{0 \leq x \leq 3} d_a(\mathcal{I}, x) + \max \begin{cases} d_c(\mathcal{I}, 3 - x), \\ d_c(\mathcal{D}, 3 - x), \\ d_c(\mathcal{U}, 3 - x) + 1 \end{cases} \\ &= 7. \end{aligned}$$

The entire calculation for  $d_a$  is summarized in the following table:

final $d_a(\Pi, i)$				
	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>
$\mathcal{I}$	$-\infty$	3	7	8
$\mathcal{D}$	$-\infty$	6	7	8
$\mathcal{U}$	0	3	4	5

Since  $a$  is the root vertex of the example tree, we take the maximum value from the  $i = 3$  column to answer the original question. As  $d_a(\mathcal{I}, 3) = 8$ , we conclude that 8 vertices of  $T$  can be dominated with 3 vertices.

It is easy to modify the algorithm to keep track of the vertices that form an optimal partial dominating set. In the above example, the value  $d_a(\mathcal{I}, 3)$  is determined by  $d_a(\mathcal{I}, 1) = 2$  and  $d_c(\mathcal{I}, 2) = 6$ . This indicates that both  $a$  and  $c$  were in, as was one other vertex of  $T_c$ .

**Theorem 2.1** *Given tree  $T$  of order  $n$ , and an integer  $0 \leq \ell \leq \gamma(T)$ , the algorithm computes  $d(\ell, T)$  and has time complexity  $O(\ell^2 n)$ .*

**Proof.** By the earlier discussion, we need only determine the time required to execute the **for** loop. Since each value of  $i$  requires at most  $O(i + 1)$  work, the **for** loop takes  $\sum_{i=0}^{\ell} O(i + 1) = O(\ell^2)$  time.

By multiplying this by the number of iterations that the formula is invoked, it follows that the total complexity is  $O(k^2 n)$ . ■

### 2.3 Maximum coverage saving $k$

The goal is to determine  $d(\gamma(T) - k, T)$  efficiently. We will use the shorthand  $(\gamma - k)$ -set to mean the optimal set. The key is to observe that one may assume that every subtree  $T_v$  has at least  $\gamma(T_v) - k$  vertices, or rather, almost this much.

**Lemma 2.2** *Suppose graph  $G$  is formed from the disjoint union of graphs  $F$  and  $H$  by selecting a vertex  $v$  of  $F$  and joining  $v$  to some of the vertices of  $H$ . For  $k \geq 1$ , if  $S$  is a  $(\gamma - k)$ -set of  $G$ , then*

$$\gamma(F) - k - 2 \leq |S \cap F| \leq \gamma(F) + 1.$$

**Proof.** Suppose  $|S \cap F| > \gamma(F) + 1$ . Then if one modifies  $S$  by replacing  $S \cap F$  by  $v$  and a  $\gamma$ -set of  $F$ , the closed neighborhood does not decrease, and yet the cardinality of the set does, a contradiction. This establishes the upper bound.

Let  $a$  denote the number of vertices of  $G$  needed to dominate all of  $V(H) - N(v)$ . It follows that  $|S \cap H| \leq a + 1$ , and thus  $|S \cap F| \geq (\gamma(G) - k) - (a + 1)$ . But  $\gamma(G) \geq \gamma(F) + a - 1$ , since no vertex can dominate both part of  $V(H) - N(v)$  and  $V(F) - \{v\}$ . This establishes the lower bound. ■

Let  $\mathcal{T}$  denote the set of subtrees of  $T$  that appear in the postorder edge traversal algorithm above. (We start with  $n$  1-vertex graphs and each edge processing creates another one.) The point is that for any  $F \in \mathcal{T}$ , it satisfies the condition of the above lemma (with  $H = G - V(F)$ ). That is:

**Corollary 2.3** *Any  $(\gamma - k)$ -set  $S$  of  $G$  is such that  $\gamma(F) - k - 2 \leq |S \cap F| \leq \gamma(F) + 1$  for every  $F \in \mathcal{T}$ .*

For  $-1 \leq i \leq k + 2$  we define:

*$e_v(\Pi, i)$  is the maximum  $|N_{T_v}[S]|$  such that  $S \subseteq T_v$ ,  $|S| \leq \gamma(T_v) - i$ ,  $S$  has property  $\Pi$ , and  $\gamma(F) - k - 2 \leq |S \cap F| \leq \gamma(F) + 1$  for every  $F \in \mathcal{T}$  that lies within  $T_v$ .*

The same three properties are used. For example,  $e_v(\mathcal{I}, -1) = |T_v|$  by taking a dominating set of  $T_v$  and adding  $v$  if necessary. For the trivial 1-vertex tree, the only possibilities are:

$$e_v(\mathcal{I}, j) = 1 \text{ for } j = -1, 0, \quad \text{and} \quad e_v(\mathcal{U}, j) = 0 \text{ for } j = -1, 0, 1.$$

As before, the overall value is extracted at the root  $r$  at the end:

$$d(\gamma(T) - k, T) = \max\{e_r(\mathcal{I}, k), e_r(\mathcal{D}, k), e_r(\mathcal{U}, k)\}.$$

The formulas are similar to those given in Figure 2. Some of the changes are minor: the **for** loop runs from  $k + 2$  down to  $-1$ ; the range of  $x$  for the maximization is increased. However, there is one major change: there are two possibilities for the domination number of the new tree in terms of the domination numbers of the subtrees:

$\gamma(T_v^{\text{new}})$  is  $B$  or  $B - 1$  where  $B = \gamma(T_v^{\text{old}}) + \gamma(T_c)$ .

We say that the edge  $e = vc$  is a *reducer* if  $\gamma(T_v^{\text{new}}) = B - 1$ . If the edge is not a reducer, the formulas are virtually the same; but for a reducer, we have to save  $i + 1$  on the domination numbers of the two constituent subtrees. For example, we give the formula for  $e_v(\mathcal{I}, i)$  below. The formulas for  $e_v(\mathcal{D}, i)$  and  $e_v(\mathcal{U}, i)$  are similar.

---


$$e_v(\mathcal{I}, i) \leftarrow \max_{-1 \leq x \leq i+1+\delta} e_v(\mathcal{I}, x) + \max \begin{cases} e_c(\mathcal{I}, i + \delta - x), \\ e_c(\mathcal{D}, i + \delta - x), \\ e_c(\mathcal{U}, i + \delta - x) + 1 \end{cases}$$

where  $\delta = 1$  if reducer and 0 otherwise,

and  $e_w(\Pi, j)$  is considered  $-\infty$  for  $j > k + 2$  (and all  $w, \Pi$ ).

---

Figure 4: Processing edge  $vc$ :  $e_v(\mathcal{I}, i)$

---

As an example, consider the tree of Figure 3. Assume  $k = 1$  and we have already processed all the edges in the subtree rooted at  $c$ , and (separately) the edge  $ab$ . The edge  $ac$  is not a reducer:  $\gamma(T_v^{\text{new}}) = 3$ ,  $\gamma(T_v^{\text{old}}) = 1$  and  $\gamma(T_c) = 2$ . The tables below give the current values of  $e_a$  and  $e_c$ . (Remember that  $i$  is the number of dominators being saved.)

		$e_a(\Pi, i)$				
		-1	0	1	2	3
$\mathcal{I}$		2	2	$-\infty$	$-\infty$	$-\infty$
$\mathcal{D}$		2	2	$-\infty$	$-\infty$	$-\infty$
$\mathcal{U}$		0	0	0	$-\infty$	$-\infty$

		$e_c(\Pi, i)$				
		-1	0	1	2	3
$\mathcal{I}$		6	6	5	$-\infty$	$-\infty$
$\mathcal{D}$		5	4	3	$-\infty$	$-\infty$
$\mathcal{U}$		2	2	2	0	$-\infty$

The calculation for  $e_a$  is summarized in the following table:

		final $e_a(\Pi, i)$				
		-1	0	1	2	3
$\mathcal{I}$		8	8	7	3	$-\infty$
$\mathcal{D}$		8	8	7	6	$-\infty$
$\mathcal{U}$		5*	5	4	3	0

Note the value  $e_a(\mathcal{U}, -1)$  marked with  $*$ . If one is forced to not dominate the root, but allowed to use four vertices, then the optimal placement is to choose (for instance) the four children of  $c$ ; but in a  $\gamma$ -set, one would only spend two vertices in  $T_c$ , and thus this possibility is excluded by the definition of  $e_a$ .

By an argument similar to the proof of Theorem 2.1 it follows that:

**Theorem 2.4** *Given tree  $T$  of order  $n$ , and an integer  $0 \leq k \leq \gamma(T)$ , the algorithm computes  $d(\gamma(T) - k, T)$  and has time complexity  $O(k^2n)$ .*

By Lemma 1.1 it follows that we have a linear-time algorithm for the  $k$ -reinforcement number  $r^k(T)$  of a tree  $T$  for fixed  $k$ .

### 3 Reinforcement in Trees

In this section we consider the properties of reinforcement in trees.

The value for the path is easily calculated:

**Observation 3.1** *Let  $n = 3m + i$  with  $i \in \{1, 2, 3\}$ . For  $k \leq m$  the path has  $r^k(P_n) = 3(k - 1) + i$ .*

#### 3.1 Maximum and minimum reinforcement numbers

The canonical upper and lower bounds of Kok and Mynhardt [11] have analogues for  $k$ -reinforcement:

**Lemma 3.2** *For any graph  $G$  of order  $n$ , and integer  $0 \leq k < \gamma(G)$ ,*

$$k \leq r^k(G) \leq \frac{kn}{\gamma(G)}.$$

**Proof.** The lower bound follows since the domination number can be decreased by only one by the addition of an edge. For the upper bound, it is clear that  $d(\ell, G) \geq \ell(n/\gamma)$  for  $0 \leq \ell \leq \gamma$ , and so  $r^k(G) = n - d(\gamma - k, G) \leq n - (\gamma - k)(n/\gamma) = kn/\gamma$ . ■

Equality is easy to achieve. For the lower bound, one example is a *wounded spider*: let  $W_s$  denote the tree resulting from a star with  $s$  leaves where every edge except one is subdivided once. Here  $\gamma(W_s) = s$  and  $r^k(W_s) = k$  for  $1 \leq k \leq s-1$ . The tree  $W_6$  is shown in Figure 5.

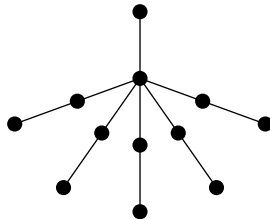


Figure 5: A tree with  $r^k = k$  for  $1 \leq k \leq 5$

For the upper bound, there is equality for the disjoint union of equal-size stars. If the stars are not  $K_2$ , one can add edges joining the leaf on one star to another star, and so create a tree while preserving the reinforcement number. Indeed, we define  $C(a, b)$  as the caterpillar on  $n = a(b+3)$  vertices obtained from the path with  $3a$  vertices by adding  $b$  new end-vertices adjacent to every third vertex on the path, starting with the second vertex. Figure 6 shows  $C(3, 2)$ . For  $b \geq 1$ , the caterpillar  $C(a, b)$  has  $\gamma = a$  and  $r^k = k(b+3)$  for  $k < a$ , and hence there is equality in the upper bound of Lemma 3.2.

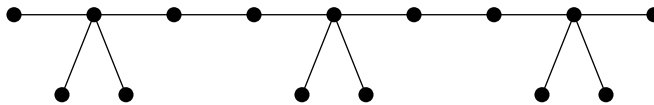


Figure 6: A tree with maximum  $r^2$

As a corollary of the above we get:

**Corollary 3.3** For any tree  $T$  of order  $n$ , and integer  $k \geq 0$

$$k \leq r^k(T) \leq \frac{kn}{k+1},$$

and these bounds are sharp.

**Proof.** The upper bound of Lemma 3.2 is maximized at  $\gamma = k + 1$ . ■

For most values of  $n$ ,  $k$  and  $\gamma$  one can construct a tree  $T$  with  $r^k(T) = \lfloor kn/\gamma \rfloor$ . However, the upper bound in Lemma 3.2 is not achievable for every combination

of  $n$ ,  $k$  and  $\gamma$ . In particular, when  $\gamma = n/2$ , there does not exist a tree  $T$  with  $r(T) = 2$ . This can be established from the characterization of connected graphs with domination number  $n/2$ . That involves the corona of a graph: recall that the *corona*  $Cor(G)$  of graph  $G$  is obtained by adding a new end-vertex adjacent to each original vertex. For example, the wounded spider  $W_s = Cor(K_{1,s-1})$ .

**Theorem 3.4** [13, 5] *A connected graph  $G$  has  $\gamma(G) = n/2$  if and only if  $G = C_4$  or  $G = Cor(H)$  for some connected graph  $H$ .*

It turns out that when the domination of a graph is maximum, the reinforcement number is small.

**Lemma 3.5** *For a graph  $H$  of order  $m$ ,  $r^k(Cor(H)) = k$  for  $1 \leq k \leq m - \gamma(H)$ .*

**Proof.** We have  $\gamma(Cor(H)) = m$ . Let  $S$  be a minimum dominating set of  $H$  and let  $W = V(Cor(H) - N[S])$ . It follows that  $d(m - k, Cor(H)) \geq 2m - k$ , achieved by adding  $\gamma(H) - k$  of  $W$  to  $S$ . Thus by Lemmas 1.1 and 3.2 the result follows. ■

**Corollary 3.6** *For a tree  $T$  with  $\gamma(T) = n/2$ ,  $r^k(T) = k$  for  $1 \leq k \leq n/4$ .*

The tree  $Cor(W_s)$  shows that the range of  $k$  cannot be extended.

### 3.2 Vertex removal

There is also the question of how the reinforcement number of a tree changes as it is altered. In particular, consider the impact of vertex removal. The removal of a vertex may dramatically decrease the reinforcement number. An example is given by taking the caterpillar  $C(2, b)$  (the double-star with the central edge subdivided twice): removing one of the large-degree vertices decreases the reinforcement number from  $b + 3$  to 1. Vertex removal may also dramatically increase the reinforcement number. An example is given by taking the caterpillar  $C(2, b)$  and subdividing the central edge: removing the new vertex increases the reinforcement number from 1 to  $b + 3$ .

### 3.3 Extension property for $\gamma - 1$

It turns out that there is a special property for 1-reinforcement in a tree: there always exists a choice of the set  $S$  that achieves  $d(\gamma - 1, T)$  that can be extended to a  $\gamma$ -set. This result does not extend to other graphs (even cacti, that is, graphs where each block is either  $K_2$  or a cycle), and does not hold true for  $\gamma - 1$  replaced by any  $\gamma - i$  with  $i \geq 2$  (see below).

**Theorem 3.7** *In a tree  $T$  there exists a  $(\gamma - 1)$ -set that is part of a  $\gamma$ -set.*

**Proof.** By starting with any  $\gamma$ -set, it is clear that there exists a spanning forest  $F$  of  $T$  that is the disjoint union of  $\gamma$  stars. Fix  $F$  and label any edge not in  $F$  as *surplus*. Let  $S$  be a  $(\gamma - 1)$ -set of  $T$ .

Consider a surplus edge  $e = v_1v_2$ : clearly  $\gamma(T - e) = \gamma(T)$ . For  $T_1$  either component of  $T - e$ , we say that  $T_1$  is overfull if  $|S \cap T_1| > \gamma(T_1)$ , and full if  $|S \cap T_1| = \gamma(T_1)$ . If  $T_1$  is overfull, then we can rearrange the vertices of  $S \cap T_1$  to dominate  $T_1$  and move one over to the other end of  $e$  if necessary, and not decrease the closed neighborhood of  $S$ . By repeated application of this, it follows that

*there is a choice of  $S$  such that for every surplus edge  $e$  neither component of  $T - e$  is overfull.*

For this  $S$  it follows that exactly one component of  $T - e$  is full. So, orient each surplus edge away from the full side. If  $T_1$  is the full component of  $T - e$ , and  $e'$  is a surplus edge inside  $T_1$ , then, by counting, the edge  $e'$  must be oriented towards  $e$ . It follows that there is a component  $F_0$  in  $F$  (the star-forest) such that every surplus edge incident with  $F_0$  is oriented towards  $F_0$ .

Now, consider a surplus edge  $e = v_1v_2$  with  $v_2 \in F_0$ . Suppose  $S$  does not dominate  $T_1$ , the component of  $T - e$  containing  $v_1$ . Then if we replace  $S \cap T_1$  with a minimum dominating set of  $T_1$ , we have a set  $S'$  with  $|S'| = |S|$  and  $|N[S']| \geq |N[S]|$ —we might lose  $v_2$  but we gain (at least) one vertex in  $T_1$ . By repeating the process, we end up with an  $S''$  which dominates all the vertices outside  $F_0$ .

But  $F_0$  can be dominated by one vertex, say  $x$ . Thus  $S''$  is part of the  $\gamma$ -set  $S'' \cup \{x\}$ . ■

It follows that the original reinforcement number is determined by the smallest private neighborhood of a vertex in some  $\gamma$ -set:

**Corollary 3.8** *For every tree  $T$*

$$r(T) = \min_{D \ni v} |N[v] - N[D - v]|$$

where the minimum is taken over all  $\gamma$ -sets  $D$  and all  $v \in D$ .

To see that Theorem 3.7 does not generalize to  $\gamma - 2$ , consider the octopus created from the star on three edges by subdividing each edge. Then the unique  $(\gamma - 2)$ -set is the central vertex, but this is not contained in any minimum dominating set. To see that Theorem 3.7 does not generalize to other graphs, consider the following cactus on 7 vertices. Start with two disjoint  $P_3$ 's and add a new vertex  $w$  adjacent to the middle and one end of each  $P_3$ . Then  $\{w\}$  is the unique  $(\gamma - 1)$ -set, but is not contained in any minimum dominating set.

## 4 Total Reinforcement

Recall that the total domination number  $\gamma_t(G)$  of a graph  $G$  is the minimum cardinality of a set  $S$  such that  $N(S) = V$ ; that is, every vertex is adjacent to an element of  $S$ . A  $\gamma_t$ -set is one that achieves the bound. One can then define  $r_t^k(G)$  as the minimum number of edges that must be added to reduce the total domination number by  $k$  (or to its minimum, viz. 2). In this section we sketch how to extend the tree algorithm for  $r^k$  to  $r_t^k(G)$ .

Consider an augmenting edge-set  $F$  and a  $\gamma_t$ -set  $S$  of  $G \oplus F$ . Let  $I(S)$  denote the number of isolated vertices in  $\langle S \rangle$ . It is clear that  $|F|$  is at least  $|V - N_G[S]| + |I(S)|/2$ ; on the other hand, given any set  $S$ , one can always choose an augmenting set  $F$  of cardinality  $\lceil |V - N_G[S]| + |I(S)|/2 \rceil$  such that  $S$  total dominates  $G \oplus F$ . It follows that:

**Lemma 4.1** *For any graph  $G$ ,  $r_t^k(G)$  is the ceiling of the minimum of  $|V - N_G[S]| + |I(S)|/2$  taken over all subsets  $S \subseteq V$  of cardinality at most  $\gamma_t(G) - k$ .*

## 4.1 Algorithm

One can extend those ideas to a linear-time algorithm for the total domination version of reinforcement. The approach is basically the same, except we must now subdivide the property  $\mathcal{I}$  into two cases based on whether the vertex is dominated or not. Call these properties  $\mathcal{ID}$  and  $\mathcal{IU}$  respectively. The following table gives the property of the combined graph given the properties of the subtrees:

		old $v$			
		$\mathcal{ID}$	$\mathcal{IU}$	$\mathcal{D}$	$\mathcal{U}$
		$\mathcal{ID}$	$\mathcal{ID}$	$\mathcal{D}$	$\mathcal{D}$
c		$\mathcal{IU}$	$\mathcal{ID}$	$\mathcal{D}$	$\mathcal{D}$
		$\mathcal{D}$	$\mathcal{ID}$	$\mathcal{IU}$	$\mathcal{D}$
		$\mathcal{U}$	$\mathcal{ID}$	$\mathcal{IU}$	$\mathcal{D}$

We then need to calculate  $|N_G[S]| - |I(S)|/2$  for the best  $S$ —the reinforcement number is the complement of this rounded up. The adjustments in the formulas are as follows: an edge joining  $\mathcal{ID}$  to  $\mathcal{IU}$  means  $+\frac{1}{2}$ , an edge joining  $\mathcal{IU}$  to  $\mathcal{IU}$  means  $+1$ , an edge joining  $\mathcal{ID}/\mathcal{IU}$  to  $\mathcal{U}$  means  $+1$ . The possibilities for the trivial tree are  $\mathcal{IU}$  saving nothing (value= $\frac{1}{2}$ ), and  $\mathcal{U}$  saving zero or one (value= $0$ ).

We omit the details that allow us to conclude:

**Theorem 4.2** *There is a linear-time algorithm to compute  $r_t^k(T)$  for a tree  $T$  for fixed  $k$ .*

## 5 Conclusion

We provided fast algorithms for computing in trees the maximum domination achievable using  $k$  vertices, or using  $k$  vertices less than the minimum required to dominate the tree. It seems likely that these ideas can be extended to other graph families that have a recursive definition. We also explored the minimum and maximum values of the  $k$ -reinforcement numbers for trees. It might be interesting to see how these parameters behave for other families.

## Acknowledgments

We would like to thank Brian Dean for pointing out some of the related literature.

## References

- [1] R.B. BORIE, R.G. PARKER, AND C.A. TOVEY, *Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families*, *Algorithmica*, 7 (1992), pp. 555–581.
- [2] X. CHEN, L. SUN, AND D. MA, *Bondage and reinforcement number of  $\gamma_f$  for complete multipartite graph*, *Journal of Beijing Institute of Technology*, 12 (2003), pp. 89–91.
- [3] G.S. DOMKE, AND R.C. LASKAR, *The bondage and reinforcement numbers of  $\gamma_f$  for some graphs*, *Discrete Mathematics*, 167/168 (1997), pp. 249–259.
- [4] R.G. DOWNEY AND M.R. FELLOWS, *Fixed-parameter tractability and completeness. I. Basic results*, *SIAM Journal on Computing*, 24 (1995), pp. 873–921.
- [5] J.F. FINK, M.S. JACOBSON, L.F. KINCH, AND J. ROBERTS, *On graphs having domination number half their order*, *Periodica Mathematica Hungarica* 16 (1985), pp. 287–293.
- [6] GAREY, MICHAEL R. AND JOHNSON, DAVID S., *Computers and intractability: A guide to the theory of NP-completeness* W. H. Freeman and Co., San Francisco, 1979.
- [7] T.W. HAYNES, S.T. HEDETNIEMI, AND P.J. SLATER, *Fundamentals of Domination in Graphs*, Marcel Dekker, New York, 1998.
- [8] T.W. HAYNES, S.T. HEDETNIEMI, AND P.J. SLATER, *Domination in Graphs: Advanced Topics*, Marcel Dekker, New York, 1998.
- [9] W.-L. HSU, *The distance-domination numbers of trees*, *Operations Research Letters*, 1 (1981/82), pp. 96–100.

- [10] P. C. JONES, *An algorithm to determine the reinforcement number of a tree*, M.A. Thesis, University of Louisville, 2002.
- [11] J. KOK AND C. M. MYNHARDT, *Reinforcement in graphs*, *Congressus Numerantium*, 79 (1990), pp. 225–231.
- [12] MEGIDDO, NIMROD AND ZEMEL, EITAN AND HAKIMI, S. LOUIS, *The maximum coverage location problem*, *SIAM Journal on Algebraic and Discrete Methods*, 4 (1983), pp. 253–261,
- [13] C. PAYAN, AND N.H. XUONG, *Domination-balanced graphs*, *Journal of Graph Theory*, 6 (1982), pp. 23–32,
- [14] D.P. SUMNER AND P. BLITCH *Domination critical graphs*, *Journal of Combinatorial Theory. Series B*, 34 (1983), pp. 65–76.
- [15] T. WIMER AND S. HEDETNIEMI, *K-terminal recursive families of graphs*, in *Proceedings of the 250th Anniversary Conference on Graph Theory*, Fort Wayne, IN, 1986.
- [16] ZHANG, JING HUA AND LIU, HAI LONG AND SUN, LIANG, *Independence bondage number and reinforcement number of some graphs*, *Transactions of Beijing Institute of Technology. Beijing Ligong Daxue Xuebao*, 23 (2003), pp. 140–142.