New results on broadcast domination and multipacking

by

Feiran Yang
B.Sc., Sichuan University, 2011
B.Sc., University of Prince Edward Island, 2012

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Mathematics and Statistics

New results on broadcast domination and multipacking

by

Feiran Yang
B.Sc., Sichuan University, 2011
B.Sc., University of Prince Edward Island, 2012

Supervisory Committee

Dr. Gary MacGillivray, Co-supervisor
(Department of Mathematics and Statistics)

Dr. Richard Brewster, Co-supervisor
(Department of Mathematics and Statistics, Thompson Rivers University)

Supervisory Committee

Dr. Gary MacGillivray, Co-supervisor
(Department of Mathematics and Statistics)

Dr. Richard Brewster, Co-supervisor
(Department of Mathematics and Statistics, Thompson Rivers University)

ABSTRACT

Let $G = (V, E)$ be a graph and f be a function such that $f : V \to \{0, 1, 2, \ldots, diam(G)\}$. Let $V_f^+ = \{v : f(v) > 0\}$. If for every vertex $v \notin V_f^+$ there exists a vertex $w \in V_f^+$ such that $d(v, w) \leq f(w)$, then f is called a *dominating broadcast* of G. The quantity $\sum_{v \in V} f(v)$ is called the *cost* of the broadcast. The minimum cost of a dominating broadcast is called the *broadcast domination number* of G, and is denoted by $\gamma_b(G)$. A subset $S \subseteq V$ is a *multipacking* if for every $v \in V$ and for every $1 \leq k \leq rad(G)$, $|N_k[v] \cap S| \leq k$. The *multipacking number* of G is the maximum cardinality of a multipacking of G, and is denoted by $mp(G)$.

In the first part of the thesis, we describe how linear programming can be used to give a $O(n^3)$ algorithm to find the broadcast domination number and multipacking number of strongly chordal graphs. Next, we restrict attention to trees, and describe linear time algorithms to compute these numbers. Finally, we introduce k-broadcast domination and k-multipacking, develop the basic theory and give a bound for the 2-broadcast domination number of a tree in terms of its order.

Contents

List of Figures

Chapter 1

Introduction

Broadcast domination was introduced by Erwin in 2002 [9]. For a graph G, each vertex v in G is assigned a *strength* $f(v) \in \{0, 1, 2, ..., diam(G)\}$. Let V_f^+ denote the set of vertices with positive strength. Let $u \in V(G)$. If there exists a vertex $v \in V_f^+$ such that $d(u, v) \leq f(v)$, then we say u *hears* the broadcast at v. If every vertex of G hears at least one broadcast, then f is called a *dominating broadcast*. The *cost of* f is the quantity $\sum_{v \in V} f(v)$. The *broadcast domination number of* G is the minimum cost of a dominating broadcast of G, and is denoted by $\gamma_b(G)$. Notice that if we restrict $f(v) \in \{0, 1\}$ for all $v \in V$, then f is the characteristic function of a dominating set. It follows immediately that $\gamma_b(G)$ is at most its domination number, $\gamma(G)$.

For each vertex i, let x_{ik} be an indicator variable giving the truth value of the statement "the strength of the broadcast at vertex i equals k". The definition of $\gamma_b(G)$ then leads to the following $0-1$ integer program, $B(G)$:

$$\text{Minimize } \sum_{i=1}^{n} k x_{ik},$$

subject to:

$$\sum_{d(i,j) \leq k} x_{ik} \geq 1 \text{ for each vertex } j,$$

$$x_{ik} \in \{0, 1\} \text{ for each vertex } i \text{ and integer } k \in \{0, 1, ..., diam(G)\}.$$

The dual of broadcast domination is multipacking, $MP(G)$:

$$\text{Maximize } \sum_{j=1}^{n} y_j,$$

subject to:

$$\sum_{d(i,j) \leq k} y_j \leq k \text{ for each vertex } i \text{ and integer } k \in \{0, 1, ..., diam(G)\},$$

$$y_j \geq 0 \text{ for each } j.$$

In the multipacking integer program, each y_j is an indicator variable that gives

the truth value of the statement "vertex j is in the multipacking". The constraints impose the restriction that, for any vertex i, and all integers k with $1 \leq k \leq diam(G)$, there are at most k vertices in the multipacking at distance at most k from i. Hence a *multipacking* is a subset $S \subseteq V$ such that for every $u \in V$ and every $1 \leq k \leq diam(G)$, $|N_k[v] \cap S| \leq k$ (for a definition see Chapter 2). If you break the constraint of vertex v with respect to distance $k > rad(G)$, then you must break the constraint of the central vertex with respect to distance $rad(G)$. Thus we can replace the range of k by $k \in \{0, 1, ..., rad(G)\}$. The *multipacking number of* G is the maximum cardinality of a multipacking of G, and is denoted by $mp(G)$.

The thesis is organized as follows. We introduce notation and terminology, and survey previous results, in Chapter 2. In Chapter 3 we consider the linear programming relaxation of $B(G)$. When the constraint matrix for $B(G)$ is totally balanced, both $B(G)$ and $MP(G)$ have integer optima which can be found using an algorithm due to Farber [1]. Results of Lubiw show that the constraint matrix for $B(G)$ is totally balanced when the graph is strongly chordal [19]. We combine Lubiw's results, Farber's algorithm and a doubly lexical ordering algorithm by Spinrad [25] to give an $O(n^3)$ algorithm to find the broadcast domination number and multipacking number of any strongly chordal graph. In Chapter 4 we restrict our attention to trees and simplify the linear time algorithms due to Dabney, Dean and Hedetniemi for the broadcast domination number [6], and Teshima for the multipacking number [26]. Our algorithms are based on ideas of Herke [15] (also see [16]) and Teshima [26]. In Chapter 5 we introduce k-broadcast domination and k-multipacking. We give some basic theorems and relate these to broadcast domination and multipacking, which they generalize. We then focus on $k = 2$ and give a bound for the 2-broadcast domination number of a tree in terms of its order. This implies a bound for all graphs. The thesis concludes with some suggestions for future research.

Chapter 2

Background

The purpose of this section is to survey previous work on broadcast domination and multipacking. Our notation is standard and follows [4]. Only terminology which is not standard in graph theory is explicitly defined in the thesis. In particular, the *closed neighbourhood* of v is the set of vertices $N(v) \cup \{v\}$, and is denoted as $N[v]$. The *k-closed neighbourhood* of v is the set of vertices $N_k[v] = \{u \in V(G) : d(u, v) \leq k\}$.

2.1 Broadcast domination on graphs

In 2002, Erwin introduced broadcast domination in the paper *Dominating broadcasts in graphs* [9] as a model of real life communication networks (also see [8]). He provided upper and lower bounds for $\gamma_b(G)$, and also computed its exact value for some graph families.

Proposition 2.1.1. [9] *For every nontrivial connected graph G,*

$$\gamma_b(G) \leq min\{rad(G), \gamma(G)\}.$$

Theorem 2.1.2. [9] *If G is a nontrivial connected graph, then*

$$\gamma_b(G) \geq \left\lceil \frac{diam(G) + 1}{3} \right\rceil.$$

Corollary 2.1.3. [9] *For every integer $n \geq 2$,*

$$\gamma_b(P_n) = \gamma(P_n) = \left\lceil \frac{n}{3} \right\rceil.$$

In fact we will later show that $\lceil \frac{n}{3} \rceil$ is an upper bound for the broadcast domination number of all graphs. Erwin also investigates graphs with $\gamma_b(G) \leq 3$.

Proposition 2.1.4. [9] *Let G be a connected graph and $k = min\{rad(G), \gamma(G)\}$. If $1 \leq k \leq 3$, then $\gamma_b(G) = k$.*

If a graph G has $rad(G) = \gamma_b(G)$, then we call G *radial*. This is an important class of graphs with respect to broadcast domination.

Later in 2006, in the paper *Broadcasts in graphs* [7] by Dunbar, Erwin, Haynes, Hedetniemi and Hedetniemi, broadcast domination was investigated more deeply. In the paper, they define a key concept called an efficient broadcast. A dominating broadcast is *efficient* if no vertex hears two different broadcasts.

Theorem 2.1.5. [7] *Every graph G has an optimal efficient dominating broadcast.*

The idea is whenever you have an inefficient broadcast, you can always merge two broadcast neighbourhoods that overlap so that they are covered by a single broadcast. The new broadcast is heard by all vertices that heard the original broadcast and the total of the broadcast does not change. This step is repeated until efficient broadcast is found.

We know that the dominating set problem for graphs is NP-complete [18]. People thought that the broadcast domination problem, which can be regarded as a generalized domination problem, should also be NP-complete. However later in 2006, in the paper *Optimal broadcast domination in polynomial time* [13], Heggernes and Lokshtanov used the fact that every graph has an optimal efficient dominating broadcast to give a polynomial time algorithm to find the broadcast domination number of any graph.

Theorem 2.1.6. [13] *The broadcast domination number of a graph G can be found in $O(n^6)$.*

To find an optimal dominating broadcast, Heggernes and Lokshtanov first consider the ball graph of the original graph. A *ball graph* is a graph whose vertices are the broadcast neighbourhoods of the original graph and two vertices are adjacent if if a vertex in one of these neighbourhoods is adjacent to a vertex in the other. They use the fact that there is an efficient optimal dominating broadcast to prove that there is an optimal dominating broadcast for which the ball graph is a path or cycle. The idea is to assume for each vertex $v \in V$, v is an end-point of a ball graph which is a path.

This step finds all possible optimal dominating broadcasts for which the ball graph is a path. Then they consider the case when the ball graph is a cycle. They first remove a broadcast neighbourhood from the original graph and so that the remaining subgraph should have a ball graph which is a path. The running time when the ball graph is a path is $O(n^4)$, and when the ball graph is a cycle it is $O(n^6)$.

The term *multipacking* was first introduced by Teshima in her Master's thesis *Broadcasts and multipackings in graphs* [26] in 2012. She considers broadcast domination as an integer programming problem and also its LP relaxation. Then she uses the linear programming dual to give the definition of multipacking. In her thesis she gives some basic results of multipacking and gives a linear time algorithm to find an optimal multipacking for trees. In Chapter 4, we will give a simplified algorithm like the one in her thesis. She also gives a bound of the broadcast domination number in terms of the irredundance number (which we will not define in this thesis). Another main result of her thesis relates to the broadcast domination number and the multipacking number of trees, and will appear later in this chapter.

Multipacking is studied further in the paper *On the difference between broadcast and multipacking number of graphs* [14] by Hartnell and Mynhardt in 2014. They extend Erwin's inequality chain with the addition of multipacking number.

Theorem 2.1.7. [14] *For any connected graph G,*

$$\left\lceil \frac{diam(G)+1}{3} \right\rceil \leq mp(G) \leq \gamma_b(G) \leq min\{rad(G), \gamma(G)\}.$$

Proof: For a graph G, consider a diametrical path $v_0, v_1, ..., v_{diam(G)}$. It has $diam(G)+1$ vertices. The set $\{v_0, v_3, ...\}$ is a multipacking. Therefore $mp(G) \geq \left\lceil \frac{diam(G)+1}{3} \right\rceil$. The inequality $mp(G) \leq \gamma_b(G)$ comes directly from duality theorem of linear programming and $\gamma_b(G) \leq min\{rad(G), \gamma(G)\}$ is from Proposition 2.1.1. □

They also study the ratio between $mp(G)$ and $\gamma_b(G)$.

Theorem 2.1.8. [14] *For any connected graph G,*

$$\gamma_b(G)/mp(G) < 3.$$

Proof: Since $\left\lceil \frac{diam(G)+1}{3} \right\rceil \leq mp(G)$, we have

$$3mp(G) \geq diam(G) + 1 > rad(G) \geq \gamma_b(G).$$

So $\gamma_b(G)/mp(G) < 3$. □

Interestingly, no graph G has been found such that $\gamma_b(G)/mp(G) > 2$. The small example shown in Figure 2.1 has $\gamma_b(G) = 4$ and $mp(G) = 2$.

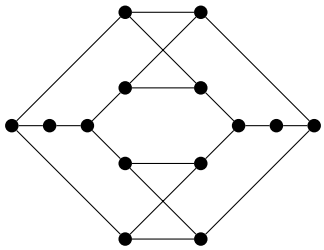


Figure 2.1: A graph G with $\gamma_b(G) = 4$, $mp(G) = 2$.

There is also an example that shows for any given integer n, there is a graph G that has $V(G) \geq n$ and $\gamma_b(G)/mp(G) = 4/3$: make many copies of the graph in Figure 2.2 and connect them together in series by joining vertex r_i to vertex l_{i+1}. The resulting graph will have $\gamma_b(G)/mp(G) = 4/3$.

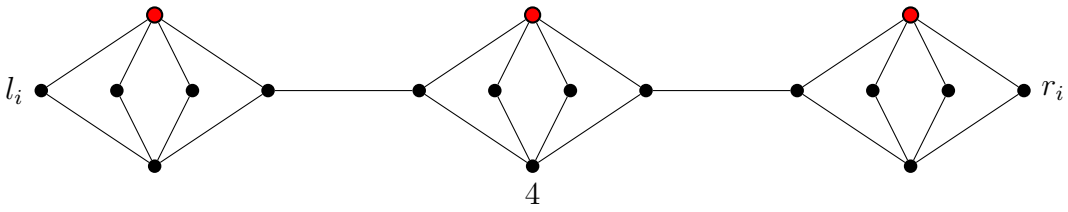


Figure 2.2: A graph G with $\gamma_b(G) = 4$, $mp(G) = 3$.

Suppose we have k copies of this graph joined together as described, we can see that for each copy we have $\gamma_b = 4$ and $mp = 3$, thus $\gamma_b(G) \leq 4k$ and $mp(G) \geq 3k$. Hartnell and Mynhardt prove that these are equalities.

Brewster, Mynhardt and Teshima study fractional broadcasting and fractional multipacking in the paper *New bounds for the broadcast domination number of a graph* [3]. These arise from considering the linear programming relaxations of the integer programming formulations $B(G)$ and $MP(G)$. Here, the broadcast strength can be a fraction, and a vertex can be consider to be fractionally in multipacking. For example, we can assign $1/3$ strength to all vertices in C_4 and this will give us a dominating broadcast with cost $4/3$. On the other hand we can pack $1/3$ for each vertex in C_4 and it will give a multipacking of size $4/3$. We denote the fractional broadcast

domination number as $\gamma_{b,f}(G)$ and fractional multipacking number as $mp_f(G)$. The duality theorem of linear programming gives the result below.

Proposition 2.1.9. [3] *For a connected graph G,*

$$mp(g) \leq mp_f(G) = \gamma_{b,f}(G) \leq \gamma_b(G).$$

The difference $mp_f(G) - mp(G)$ can be arbitrarily large. The graph shown in Figure 2.2 has fractional multipacking number 4 since we can pack 1/3 of the three inner C_4s. Using k copies of the graph joined in series as before will give us $mp_f(G) - mp(G) = k$.

2.2 Broadcast domination on trees

Broadcasts in trees have a special structure, which is exploited in the thesis *Dominating broadcasts in graphs* [15] by Herke in 2007 and in the paper *Radial trees* [16] by Herke and Mynhardt in 2009.

An important definition is that of a *shadow tree*. Suppose $P = v_0, v_1, v_2, ..., v_d$ is a diametrical path of the tree T. The shadow tree is constructed in several steps. First, consider the forest $F = T - \{v_0, v_1, v_1v_2, v_2v_3, ..., v_{d-1}v_d\}$. For each vertex v_k of P, let Q_k be a longest path in F with origin v_k, and let b_k be its terminus (possibly $v_k = b_k$). Reduce the tree to the subtree induced by the vertices belonging to $V(P) \cup (\bigcup_{k=1}^{d} V(Q_{k-1}))$. For the second step, if there exists $d(v_k, b_k) \geq d(v_k, b'_k)$, remove $Q'_k \ \{v'_k\}$ from the tree. This is the shadow tree of T, which is denoted by S_T. The *shadow* of vertex b_k is the set of vertices $\{v : d(v_k, b_k) \geq d(v_k, v)\}$.

For example, consider the tree T in Figure 2.3.

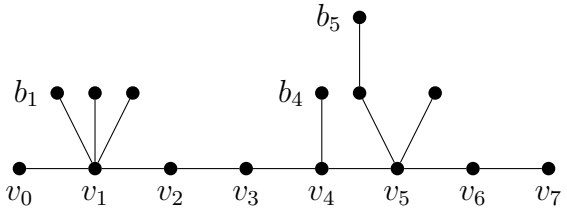


Figure 2.3: Tree T.

One diametrical P is $v_0, v_1, ..., v_7$ and for each vertex v_i we only keep the longest branch starting from it. After the first step we reduce T to the tree in Figure 2.4:

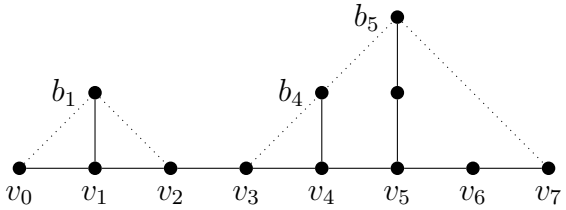


Figure 2.4: First step to construct S_T.

We have $d(b_4, v_5) \leq d(b_5, v_5)$. According to the second step of our construction, we should remove b_4 from the tree. So, finally, S_T is the tree in Figure 2.5:

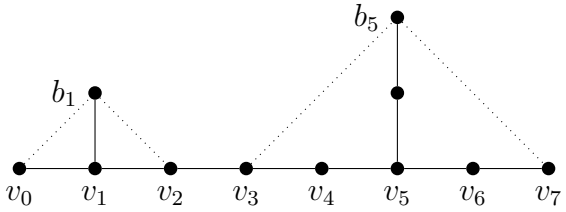


Figure 2.5: Shadow tree S_T.

Shadow trees are extremely helpful in finding the broadcast domination number for trees.

Theorem 2.2.1. [16] *For a tree T and its shadow tree S_T, $\gamma_b(T) = \gamma_b(S_T)$.*

This theorem allows us to restrict attention to the simplified tree knowing it has the same broadcast domination number.

Although the shadow tree is an induced subgraph of the original tree, the analog of Theorem 2.2.10 does not hold for all graphs.

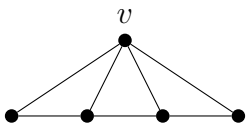


Figure 2.6: A graph G with an induced subgraph having larger broadcast domination number.

Consider the graph G shown in Figure 2.6. Since v is a dominating vertex, $\gamma_b(G) = 1$. However if we remove v, the remaining graph is P_4 and $\gamma_b(P_4) = 2$.

Other important definitions given in this paper are split-set and split-edge. Let T be a tree with diametrical path P. A *split-set* is a set of edges on P whose removal

"splits" T into components such that each component T_i has even positive diameter and $T_i \cap P$ is a diametrical path of T_i. A *split-edge* is an edge that is contained in some split-set. For example, in Figure 2.5, v_2v_3 is a split-edge. On the other hand, the edge v_3v_4 is not a split edge since its removal creates a subtree with diametrical path from b_5 to v_7. In general, all the edges that have two ends in some shadow cannot be a split-edge.

Herke and Mynhardt show that the broadcast domination number is a function of the largest size of a split-set.

Theorem 2.2.2. [16] *If M is split-set with maximum cardinality m of a tree T, then*

$$\gamma_b(T) = \left\lceil \frac{diam(T) - m}{2} \right\rceil.$$

Theorem 2.2.3. [16] *A tree T is radial if and only if it has no nonempty split-set.*

They also bound the broadcast domination number of a tree in terms of its order.

Corollary 2.2.4. [16] *For any tree T of order n, $\gamma_b(T) \leq \lceil \frac{n}{3} \rceil$.*

The proof is by induction on the number of vertices. They show that we can partition the tree into two parts where one part has exactly k vertices, where $k \equiv 0 \pmod 3$. In this part, the bound applies without the ceiling. We adopt a similar strategy in Chapter 5, when we generalize this result. The theorem implies that any graph G of order n will have $\gamma_b(G) \leq \lceil \frac{n}{3} \rceil$ since a dominating broadcast of a spanning tree of a graph is a dominating broadcast of the graph.

Corollary 2.2.5. *For any graph G of order n, $\gamma_b(G) \leq \lceil \frac{n}{3} \rceil$.*

Another nice result appears in the paper *A Linear-Time Algorithm for Broadcast Domination in a Tree* [6] (also see [5]) by Dabney, Dean and Hedetniemi in 2009. They give an linear time algorithm to find an optimal dominating broadcast for trees. The linearity of the algorithm is based on a complex data structure.

Theorem 2.2.6. [5, 6] *The broadcast domination number for trees can be found in $O(n)$ time.*

We give a simpler linear time algorithm to find the broadcast domination number of a tree in Chapter 4. A cubic algorithm based on linear programming that works for all strongly chordal graphs is described in Chapter 3.

In Teshima's Master's Thesis [26], she proved an interesting result shown below.

Theorem 2.2.7. [26] *For any tree T, $\gamma_b(T) = mp(T)$.*

In Chapter 3, we will give an alternative proof of Theorem 2.2.7 using the Duality Theorem of Linear Programming.

There are some results of classifying trees into different categories. In the paper *More trees with equal broadcast and domination numbers* [22] (also see [21]), Lunney and Mynhardt study trees with $\gamma_b(T) = \gamma(T)$. Graphs G with $\gamma_b(G) = \gamma(G)$ are called *1-cap* graphs. They characterize 1-cap trees.

In the paper *Uniquely radial trees* [23], Mynhardt and Wodlinger study trees with $\gamma_b(T) = rad(T)$. Using a complex case analysis, they characterize the trees for which the equality holds for a unique broadcast.

Chapter 3

Broadcast Domination and Multipacking on Strongly Chordal Graphs

Let x_i and y_j be $\{0,1\}$ indicators for the vertices in a graph G with n vertices and $i,j \in \{1,2,...,n\}$. The integer programming formulation of the Minimum Dominating Set Problem is $D(G)$:

$$\text{Minimize } \sum_{i=1}^{n} x_i,$$
$$\text{subject to}$$
$$\sum_{i \in N[v_j]} x_i \geq 1 \text{ for each } j,$$
$$x_i \in \{0,1\}.$$

The integer programming dual of $D(G)$ is the maximum packing problem $P(G)$:

$$\text{Maximize } \sum_{j=1}^{n} y_j,$$
$$\text{subject to}$$
$$\sum_{j \in N[v_i]} y_j \leq 1 \text{ for each } i,$$
$$y_j \in \{0,1\}.$$

Since the dominating set problem is NP-complete for chordal graphs [2], we will not be able to use linear programming methods to solve dominating set problem on chordal graphs unless $P = NP$.

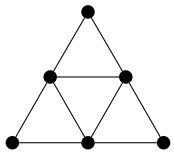


Figure 3.1: A trampoline.

The optimal value to these programs are called the domination number and packing number respectively. For the graph in Figure 3.1, the domination number is 2 and packing number is 1. However, solving the linear programming relaxation of $D(G)$ leads to a fractional solution: assign $\frac{1}{2}$ weight to the middle 3 vertices. On the other hand, the optimal packing is obtained by assigning weight $\frac{1}{2}$ weight to each of the 3 outer vertices.

3.1 Strongly chordal graphs and totally balanced matrices

Chordal graphs are a well-studied class of graphs, which play a prominent role in the study of perfect graphs. A graph is *chordal* if it contains no cycle of length greater than three as an induced subgraph. Rose gave an characterization of chordal graphs.

Definition 1. [24] *A* perfect elimination ordering *of graph $G = (V, E)$ is an ordering $v_1, v_2, ..., v_n$ of V with the property that, for each i, j and k, if $i < j < k$ and $v_iv_j, v_iv_k \in E$, then $v_jv_k \in E$.*

There are several characterizations of chordal graphs. The following is a characterization based on the perfect elimination ordering.

Theorem 3.1.1. [24] *A graph is chordal if and only if it admits a perfect elimination ordering.*

Farber studies the chordal graphs where he regards domination problems and packing problems as linear programming problems. As we saw above, the optimal solutions for domination and packing on chordal graphs given by linear programming may not be integer.

Farber realized that the perfect elimination ordering is not quite strong enough to give an integer solution to domination and packing problems. So he added one more

restriction to the order and gave a new ordering. One can speculate on the origin of this condition based on the discussion following Theorem 3.1.5 below.

Definition 2. [10] *A* strong elimination ordering *of graph $G = (V, E)$ is an ordering $v_1, v_2, ..., v_n$ of V that satisfies the following two conditions for each i, j, k and l:*

(a) if $i < j < k$ and $v_i v_j, v_i v_k \in E$, then $v_j v_k \in E$.

(b) if $i < j < k < l$ and $v_i v_k, v_i v_l, v_j v_k \in E$, then $v_j v_l \in E$.

According to the definition of strong elimination ordering, Farber gives the lemma.

Lemma 3.1.2. [10] *An ordering $v_1, v_2, ..., v_n$ of the vertices of G is a strong elimination ordering of G if and only if for each i, j, k and l with $i \leq j$, $k \leq l$ and $i \in N[v_k], i \in N[v_l]$ and $j \in N[v_k]$, we have $j \in N[v_l]$.*

The *closed neighbourhood adjacency matrix* $A(G)$ of graph G has 1 in position (i, j) if $i \in N[v_j]$ and 0 elsewhere. Consider the submatrix of the closed neighbourhood adjacency matrix of G consisting of rows i and j, and columns k and l. Lemma 3.1.2 says that if the left column and top row of this submatrix have all entries equal to 1, then the bottom right entry must also be 1. That is, the matrix $\Gamma = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ cannot be a submatrix of $A(G)$.

Similar to the characterization of chordal graphs, Farber gives the definition below.

Definition 3. [10] *A graph is* strongly chordal *if it admits a strong elimination ordering.*

Since an ordering that satisfies condition (a) in Definition 2 is a perfect elimination ordering, every strongly chordal graph is chordal.

A related concept that of a is totally balanced matrix. These were studied by Anstee and Farber [1]. A 0,1-matrix is *totally balanced* if it has no incidence matrix of any cycle of at least length 3 as a submatrix. Totally balanced matrices have the following interesting property:

Theorem 3.1.3. [1, 12] *If M is a totally balanced matrix, then the polyhedra $\{x \in \mathbf{R} : Mx \geq 1, x \geq 0\}$ and $\{x \in \mathbf{R} : Mx \leq 1, x \geq 0\}$ have $(0, 1)$-extreme points.*

This theorem suits Farber's needs perfectly because he wants to find integer solutions for the domination problem and the packing problem. Anstee and Farber give the relationship between strongly chordal graphs and totally balanced matrices by an elegant theorem:

Theorem 3.1.4. [1, 10, 11] *A graph is strongly chordal if and only if the closed neighbourhood adjacency matrix $A(G)$ is totally balanced.*

Recall the matrix Γ defined earlier. A Γ-*free matrix* is a matrix with no submatrix equal to Γ and a Γ-*freeable matrix* is a matrix where there exists row and column permutations such that the resulting matrix is Γ-free. Lubiw shows the connection between Γ-free matrices and strongly chordal graphs as below:

Theorem 3.1.5. [19, 20] *A matrix M is totally balanced if and only if it is Γ-freeable.*

Since we know the adjacency matrix of a strongly chordal graph is totally balanced. Thus by Theorem 3.1.5 it is Γ-freeable. Indeed Lemma 3.1.2 tells us the strong elimination ordering gives the row and column orderings to make A(G) Γ-free. By Theorem 3.1.3, there are $(0, 1)$-optimal solutions to the linear programs. These $(0, 1)$ solutions correspond to optimal solutions for $D(G)$ and $P(G)$.

Farber uses the Γ-free structure of $A(G)$ to develop a primal-dual algorithm for solving $D(G)$ and $P(G)$. In Stage One of the algorithm a packing is constructed. The decision variables y_j code membership in the packing: $y_j = 1$ if v_j is in the packing and $y_j = 0$ otherwise. In Stage Two, a dominating set is constructed. The decision variable $x_i = 1$ if vertex v_i is in the dominating set and $x_i = 0$ otherwise. In fact, it is more helpful to view x_i as coding the decision to include the neighbourhood around v_i as a dominating neighbourhood.

Complementary slackness guides the choices in the algorithm. In particular, if $y_j = 1$, then the corresponding constraint in $D(G)$ must have zero slack. That is y_j must be dominated by exactly one vertex. Conversely, if $x_i = 1$, then the constraint corresponding to neighbourhood $N[v_i]$ in $P(G)$ must have zero slack. That is $N[v_i]$ must contain one vertex from the packing. To this end, we will let $h(i) = 1$ if $N[v_i]$ contains no packing vertices and $h(i) = 0$ if $N[v_i]$ does contain a packing vertex.

Then the algorithm can be illustrated as:

Algorithm 1. *Input: A strongly chordal graph G with strong elimination $v_1, v_2, ..., v_n$.*
Output: A minimum dominating set x_i and maximum packing set y_j.
Initially, $T = \{v_1, v_2, ..., v_n\}$, each $y_j = 0$, each $x_i = 0$ and each $h(k) = 1$.

Stage One*: for $j = 1$ to n*
If $h(k) = 1$ for all k with $k \in N[v_j]$
Then $1 \to y_j$ and $0 \to h(k)$ for all $k \in N[v_j]$.

Stage Two: for $i = n$ to 1

If $h(i) = 0$ and there exists $v_j \in T$ such that $v_j \in N[v_i]$ and $y_j = 1$
Then $1 \to x_i$ and $T - \{v_j\} \to T$.

For the first stage, the algorithm greedily selects the vertices from v_1 to v_n into the packing. Whenever a vertex has no neighbours that are adjacent to a vertex in the packing, it is put into the packing set. For Stage Two, it uses complementary slackness to find the corresponding dominating set. Whenever a vertex has a neighbour in the packing that is not dominated yet, it selects that vertex to go into the dominating set.

As an example, we solve the domination and packing problem for the graph in Figure 3.2.

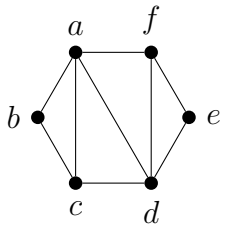


Figure 3.2: A strongly chordal graph.

First we find out that $efdcab$ is the strong elimination ordering $v_1, v_2, v_3, v_4, v_5, v_6$ of the graph above. The corresponding closed neighbourhood adjacency matrix is:

	e	f	d	c	a	b
e	1	1	1	0	0	0
f	1	1	1	0	1	0
d	1	1	1	1	1	0
c	0	0	1	1	1	1
a	0	1	1	1	1	1
b	0	0	0	1	1	1

Initially we have $x_i = 0$, $y_j = 0$ and $h(k) = 1$ for all $i, j, k \in \{1, 2, 3, 4, 5, 6\}$. Also we have $T = \{v_1, v_2, v_3, v_4, v_5, v_6\}$.

Step 1: $j = 1$. Since v_1 has all neighbours with $h(k) = 1$, so $y_1 = 1$. We now have the y-values to be:

$$\begin{pmatrix} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The h-values should now be:

$$\begin{pmatrix} h(1) & h(2) & h(3) & h(4) & h(5) & h(6) \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

T and the x-values remain the same.

Step 2: $j = 2$. Since $h(2) = 0$ and $v_2 \in N[v_2]$, we will not select v_2 into the packing set. No values will change.

Step 3,4,5: Similar to Step 2, each of the vertices v_3, v_4 and v_5 have a neighbour v_k with $h(k) = 0$, we will not select v_3, v_4, v_5 into the packing. No values will change.

Step 6: $j = 6$. $N[v_6] = \{v_4, v_5, v_6\}$, and $h(k) = 1$ for $k \in \{4, 5, 6\}$, so we have $y_6 = 1$. Now the y-values are:

$$\begin{pmatrix} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The h-values are:

$$\begin{pmatrix} h(1) & h(2) & h(3) & h(4) & h(5) & h(6) \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

T and x-values remain the same.

We have gone through Stage One of the algorithm, now we move on to Stage Two.

Step 7: $i = 6$. We have $h(6) = 0$, $y_6 = 1$ and $v_6 \in T$. So we choose v_6 into the dominating set. Now the x-values are:

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The T is now $\{v_1, v_2, v_3, v_4, v_5\}$.
The y-values and the h-values do not change.

Step 8: $i = 5$. v_5 is not adjacent to any vertex v_j that has $y_j = 1$ and $v_j \in T$. So v_5 will not be selected into the dominating set. No values will change.

Step 9: Similar to Step 8. When $i = 4$, v_4 is not adjacent to any vertex with $y_j = 1$ and $v_j \in T$. v_4 is not selected into the dominating set. No values will change.

Step 10: $i = 3$. We have $h(3) = 0$, $y_1 = 1$, $v_1 \in T$ and $v_1 \in N[v_3]$. So we have v_3 in the dominating set. Now the x-values are:

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

The T will be $\{v_2, v_3, v_4, v_5\}$.
The y-values and the h-values do not change.

Step 11,12: v_1 and v_2 no longer have any neighbour v_j with $y_j = 1$ and $v_j \in T$. v_1 and v_2 will not be selected into the dominating set.

Finally we have $y_1 = 1$ and $y_6 = 1$. Also we have $x_3 = 1$ and $x_6 = 1$. So as a result, $\{b, e\}$ will be an optimal packing of the graph and $\{b, d\}$ is a minimum dominating set.

Theorem 3.1.6. *[10] The final values of $x_1, x_2, ..., x_n$ and $y_1, y_2, ..., y_n$ give an optimal solution of $D(G)$ and $P(G)$ and the algorithm halts after $O(|V| + |E|)$ operations.*

This algorithm may fail if the given ordering is not a strong elimination order. Stage One always constructs a feasible packing, but without a Γ-free matrix it might not be optimal. If the first stage does not give an optimal packing, then Stage Two will give the dominating set solution not even feasible. An easy example is P_5, if we order the middle vertex first we will have the matrix as below.

	v_3	v_1	v_2	v_4	v_5
v_3	1	0	1	1	0
v_1	0	1	1	0	0
v_2	1	1	1	0	0
v_4	0	0	1	1	1
v_5	0	0	0	1	1

If we greedily select the packing first, we will have v_3 in the packing set and then every vertex is adjacent to at least one vertex with $h(k) = 0$. So we can no longer put any more vertices into the packing thus the packing is not optimal. And if we check second stage, we will have only v_4 in the dominating set. Obviously the dominating set is not feasible.

3.2 Applying Farber's algorithm to broadcast domination and multipacking

Farber's algorithm works perfectly for the vertex domination problem, but it does not solve the broadcast domination problem, since the constraint matrix of the broadcast domination problem is not simply the adjacency matrix of the graph. To give an algorithm to find the optimal dominating broadcast, first we need to introduce the idea of k-adjacency matrix and ball matrix A^*. The *k-adjacency matrix* A^k has is an $n \times n$ incidence matrix where the rows correspond to vertices and columns correspond to k-closed neighbourhoods. The i, j^{th} entry of A^k is 1 if the vertex v_i is contained in the k-closed neighbourhood of v_j, otherwise is 0. Clearly we can see that the closed neighbourhood adjacency matrix is just A^1. We define the *ball matrix* to be the $\begin{bmatrix} A & A^2 & ... & A^r \end{bmatrix}$ where r is the radius of the graph. Before we apply an algorithm similar to Farber's, we need to clarify that the ball matrix of a strongly chordal graph is totally balanced.

Theorem 3.2.1. [19, 20] *The ball matrix of a strongly chordal graph is totally balanced.*

Before we prove the theorem, we need some other tools.

Definition 4. [19, 20] *Given an $n \times k$, $0-1$ matrix M, the* row intersection matrix *is an $n \times n$, $0-1$ matrix with entry ij equal to 1 if, and only if, there is a column in which both row i and row j have a 1. The row intersection matrix of M is the product $M \cdot M^T$ using the convention that $1 + 1 = 1$.*

Theorem 3.2.2. [19, 20] *The row intersection of a totally balanced matrix is also totally balanced.*

Proof of Theorem 3.2.1: It is clear that $A^{k+1} = A \cdot A^k$ (where $1 + 1 = 1$). The proof is induction on k that the matrix $\begin{bmatrix} I & A & A^2 & ... & A^k \end{bmatrix}$ is totally balanced.

Recall that the closed neighbourhood adjacency matrix A of a strongly chordal graph is totally balanced. Suppose $\begin{bmatrix} I & A & A^2 & ... & A^k \end{bmatrix}$ is totally balanced. By applying the intersection matrix theorem to the transpose we have that

$$\begin{bmatrix} I & A & A^2 & ... & A^k \end{bmatrix}^T \cdot \begin{bmatrix} I & A & A^2 & ... & A^k \end{bmatrix}$$

is totally balanced matrix. The resulting matrix will have $\begin{bmatrix} A & A^2 & ... & A^{k+1} \end{bmatrix}$ as a submatrix. Therefore $\begin{bmatrix} I & A & A^2 & ... & A^{k+1} \end{bmatrix}$ is totally balanced. □

Unfortunately, even if we have the strong elimination ordering of the vertices, the ball matrix A^* may not be Γ-free. Take a P_4 with vertex sequence v_1, v_2, v_3, v_4. It is clear this it is a strong elimination ordering of the vertices. However the ball matrix $\begin{bmatrix} A & A^2 \end{bmatrix}$ is:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & \boxed{1} & \boxed{1} & 1 & 1 & 1 \\ 0 & 0 & 1 & \boxed{1} & \boxed{0} & 1 & 1 & 1 \end{pmatrix}$$

The submatrix consisting of rows 3 and 4 and columns 4 and 5 is a Γ.

In 1985, Hoffman, Kolen and Sakarovitch proved that given an $n \times m$ totally balanced $(0,1)$-matrix A, in $O(nm^2)$ time we can get a Γ-free matrix after row and column permutations [17]. In 1992, Spinrad improved the speed to $O(nm)$ using a new method to produce the ordering [25]. The idea is to push the 1s in the matrix to bottom right since this will give us least chance of making any Γs in the resulting matrix.

Now we can illustrate an algorithm similar to Farber's for broadcast domination and multipacking. First we apply an algorithm to permute A^* into a Γ-free ordering. Let x_{ik} be the indicator variable for the k-closed neighbourhood of vertex v_i, where $x_{ik} = 1$ if vertex v_i is broadcasting with strength k. The decision variables for the multipacking are y_j: $y_j = 1$ if vertex v_j is in the multipacking and $y_j = 0$ otherwise. Let $h(i,k) = k - \sum_{d(i,j) \leq k} y_j$ and $T_{ik} = \{j : d(i,j) \leq k \text{ and } y_j > 0\}$. Here $h(i,k)$ is the slack variable and T_{ik} is the set of vertices that are in the multipacking and hear the broadcast from v_i with strength k. Note that $i \in \{1, 2, ..., n\}$ and $k \in \{1, 2, ..., rad(G)\}$ for the definitions of $h(i,k)$ and T_{ik}. The ball matrix has its columns indexed by ik. As an abuse of notation we will use ik (in Stage Two below) as a column index for defining T_{ik}, and also as a pair of values for defining $h(i,k)$.

As introduced in Chapter 1, here we want to solve the linear programming relaxations for broadcast domination $B(G)$:

Minimize $\sum_{i=1}^{n} \sum_{k=1}^{rad(G)} k x_{ik}$,

subject to

$$\sum_{d(i,j)\leq k} x_{ik} \geq 1 \text{ for each } j,$$
$$x_{ik} \geq 0 \text{ for each } i.$$

and multipacking $M(G)$:

Maximize $\sum_{j=1}^{n} y_j$,

subject to

$$\sum_{d(i,j)\leq k} y_j \leq k \text{ for each } i \text{ and } k,$$
$$y_j \geq 0 \text{ for each } j.$$

Algorithm 2. *Algorithm that solves broadcast domination and multipacking for strongly chordal graph*

Input: A strongly chordal graph G and a Γ-free presentation of the $n \times m$ ball matrix A^*.

Output: A maximum multipacking and a minimum dominating broadcast of G.

Initially, $T = \{1, 2, ..., n\}$, each $y_j = 0$ and each $x_{ik} = 0$. $h(i, k) = k$ for all i and k.

Stage One*:*

For $j = 1$ to n

If $h(i, k) > 0$ for all i and k with $v_j \in N_k[v_i]$

Then $1 \to y_j$ add $h(i, k) - 1 \to h(i, k)$ for all i, k such that $d(i, j) \leq k$.

Stage Two:

For each x_{ik}, $T_{ik} = \{j : d(i, j) \leq k \text{ and } y_j > 0\}$.

For $ik = m$ to 1 (scan the columns from right to left)

If $h(i, k) = 0$ and $T_{ik} \subseteq T$

Then $x_{ik} = 1$ and $T = T - T_{ik}$.

Note that A^* is Γ-free. The constraints for $B(G)$ are essentially $A^*x \geq 1$. If we guarantee this, it means that every vertex is hearing some broadcast. Similarly for $M(G)$, we are using A^{*T} as the constraint matrix. The variable $h(i, k)$ is similar to the $h(k)$ variable in Farber's algorithm for domination, here $h(i, k) > 0$ means that the k-closed neighbourhood around v_i does not contain k packing vertices. We can still put the vertex into the multipacking. For Stage Two, similarly, every time we select a vertex to broadcast with strength k , its k-closed neighbourhood has k vertices in the multipacking and none yet hear the broadcast.

For example, we solve the broadcast domination and multipacking problem for the tree in Figure 3.3:

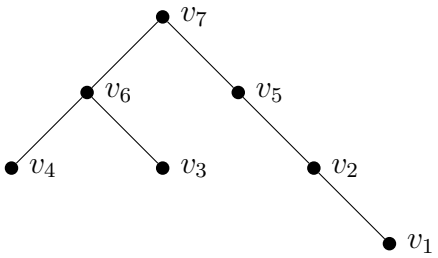


Figure 3.3: Sample tree.

Here, for example, we do not need column x_{11} since the broadcast with strength 1 from v_1 is covered by the broadcast with strength 1 from v_2. There is no reason to select x_{11} in an optimal dominating broadcast. So we can eliminate some columns and initially the matrix A^* is:

	x_{21}	x_{51}	x_{61}	x_{52}	x_{71}	x_{72}	x_{73}	y
v_1	1	0	0	1	0	0	1	0
v_2	1	1	0	1	0	1	1	0
v_3	0	0	1	0	0	1	1	0
v_4	0	0	1	0	0	1	1	0
v_5	1	1	0	1	1	1	1	0
v_6	0	0	1	1	1	1	1	0
v_7	0	1	1	1	1	1	1	0
h	1	1	1	2	1	2	3	

Then we put v_1 into the packing set since there is room for it and next step we have:

	x_{21}	x_{51}	x_{61}	x_{52}	x_{71}	x_{72}	x_{73}	y
v_1	1	0	0	1	0	0	1	1
v_2	1	1	0	1	0	1	1	0
v_3	0	0	1	0	0	1	1	0
v_4	0	0	1	0	0	1	1	0
v_5	1	1	0	1	1	1	1	0
v_6	0	0	1	1	1	1	1	0
v_7	0	1	1	1	1	1	1	0
h	0	1	1	1	1	2	2	

We cannot put v_2 into the packing since $h(2,1) = 0$ and next possible one is v_3. So we put v_3 into the packing and update one more time:

	x_{21}	x_{51}	x_{61}	x_{52}	x_{71}	x_{72}	x_{73}	y
v_1	1	0	0	1	0	0	1	1
v_2	1	1	0	1	0	1	1	0
v_3	0	0	1	0	0	1	1	1
v_4	0	0	1	0	0	1	1	0
v_5	1	1	0	1	1	1	1	0
v_6	0	0	1	1	1	1	1	0
v_7	0	1	1	1	1	1	1	0
h	0	1	0	1	1	1	1	

Now we have no way to choose more vertices into the packing set so an optimal multipacking for the specific tree is v_1 and v_3. For broadcast domination, we check from right to left and we will include both x_{61} and x_{21} into the broadcast since they do not overlap and both have h to be 0.

Theorem 3.2.3. *The algorithm halts after $O(n^3)$ operations and the resulting x_{ik} gives an optimal solution of $B(G)$ and y_j gives an optimal solution of $M(G)$.*

Proof: In Stage One, the algorithm scans through the n rows. For each row, we consider all ik. There are n values for i, and $rad(G)$ values of k. There are $O(n^2)$ values of ik. Thus Stage One takes $O(n^3)$. In Stage Two, we scan through $O(n^2)$ columns. For each column we compute T_{ik} and possibly update T. These sets have $O(n)$ entries, hence the total time is $O(n^3)$. In order to show the algorithm gives the optimal solutions, it is enough to show that these solutions are feasible and they satisfy the conditions of complementary slackness.

(i) Feasibility of dual solution. It is clear that Stage One gives a feasible solution.

(ii) Feasibility of primal solution. Clearly, each x_{ik} is 0 or 1. It suffices to show that for every vertex v_j, there exists $x_{ik} = 1$ for $d(i,j) \leq k$. Take a vertex v_j. By the choice of the multipacking vertices, there is some (i,k) such that $d(i,j) \leq k$, $h(i,k) = 0$ and $\max T_{ik} \leq j$. To see this consider y_j. If $y_j = 1$, then $h(j,1) = 0$ and $T_{j1} = \{j\}$, which means that $(i,k) = (j,1)$ is the desired pair. On the other hand, if $y_j = 0$, then in Stage One when row j is examined, there is some closed neighbourhood to which v_j belongs, say $N_k[v_i]$, for which $h(i,k) = 0$. This closed neighbourhood must have had its kth vertex added before step j(and no further vertices can be added); thus, $\max T_{ik} < j$. If $x_{ik} = 1$ we are done. Otherwise, T_{ik} was not contained in T when scanned

in Stage Two. Hence there is some other $x_{i'k'}$ such that $x_{i'k'} = 1$ and $T_{ik} \cap T_{i'k'} \neq \emptyset$. Since the closed neighbourhoods are scanned from m to 1 we have column $i'k'$ to the right of column ik. Let $j' \in T_{ik} \cap T_{i'k'}$. Then $j' \leq j$ since $\max T_{ik} \leq j$. Since the matrix is Γ-free, we have $d(i', j) \leq k'$ and $x_{i'k'} = 1$. So the primal solution is feasible.

(iii) Complementary slackness. If $x_{ik} = 1$ then $h(i, k) = 0$ by the Stage Two scan. If $y_j = 1$, it is also clear that if $d(i, j) \leq k$ and $x_{ik} = 1$, j will be removed from T and as a result, it will not be dominated twice. □

Theorem 3.2.4. *An optimal dominating broadcast and maximum multipacking for strongly chordal graphs can be found in $O(n^3)$ time.*

Proof: We have n rows and $n \cdot r$ columns in matrix A^*. It takes $O(n^2 \cdot r)$ operations to transform A^* into a Γ-free matrix. Then it takes quadratic time to get the solution of the broadcast domination and multipacking problems. Since $r = O(n)$, overall it takes $O(n^3)$ time to solve the problem. □

Corollary 3.2.5. *If G is strongly chordal, $\gamma_b(G) = mp(G)$.*

Remark: This algorithm shows that for any $n \times m$ Γ-free matrix A, the linear programming problem:

Minimize $\sum cx$

subject to $Ax \geq b, x \geq 0$

and the dual problem

Maximize $\sum by$

subject to $yA^T \leq c, y \geq 0$

can be solved greedily in $O(n + m)$ time.

The following corollary gives a simply method to find the Γ-free ordering for a tree without using Spinrad's algorithm

Corollary 3.2.6. *The ball matrix T^* for a tree T is Γ free when the rows are sorted by depth from the root and the columns are sorted in lexicographic order (reading bottom to top).*

Proof: Given any tree T, root the tree where it has minimum height. We order the vertices by the descending order of the distance from the vertex to the root. We order the columns by ascending lexicographic order reading from the bottom up. Claim T^* is Γ-free.

Suppose we have a Γ in rows a, b and columns ik, jl. By the lexicographic ordering there must be a row c with a 0 in column ik and a 1 in column jl.

	x_{ik}	x_{jl}
a	1	1
b	1	0
c	0	1

Let v be the least common ancestor of a and b.

Case 1: v is on the $a - c$ path. Then $d(a, c) = d(a, v) + d(v, c)$. Since the rows are ordered by descending order of the depth, a should have at least the same depth as b and c. So $d(b, v) \leq d(a, v)$. If the $v_j - a$ path contains v, then

$$d(v_j, b) \leq d(v_j, v) + d(v, b) \leq d(v_j, v) + d(v, a) = l$$

a contraction. Thus v_j and a belong to the same subtree rooted at a child of v. In particular, v is on the $c - v_j$ path and the $b - v_j$ path. Suppose that v is on the $v_j - c$ path. Then

$$d(v_j, b) = d(v, b) + d(v, v_j) \leq d(v, a) + d(v, v_j) \leq l.$$

This is a contradiction. So a and v_j have the least common ancestor to be a child of v. So v is on $c - v_j$ path and $b - v_j$ path. Since we know $d(c, v_j) \leq l < d(b, v_j)$. We have

$$d(c, v) + d(v, v_j) = d(c, v_j) < d(b, v_j) = d(b, v) + d(v, v_j).$$

So $d(v, c) < d(v, b) \leq d(v, a)$.

By a similar argument we can conclude that v is not on $a - v_i$ path. Hence, v does belong to both the $v_i - b$ and $v_i - c$ paths. Thus,

$$d(v_i, c) = d(c, v) + d(v, v_i) < d(b, c) + d(v, v_i) = d(b, v_i) \leq k$$

, a contradiction.

Case 2: v is not on the $a - c$ path. Then we let v' be the least common ancestor of a and c. Now v' is on the $b - a$ path and we apply similar argument like Case 1 to get the contradiction.

The running time of Algorithm 2 is still $O(n^3)$ for a tree as T^* can have $O(n^3)$ ones. This holds even when T is a path. A linear time algorithm for finding the

broadcast and multipacking number of a tree is the focus of the next Chapter. □

Chapter 4

New Algorithms for Broadcast Domination and Multipacking in Trees

In this chapter we describe linear time algorithms for finding broadcast domination number and multipacking number of trees. Since we know that a tree T has $\gamma_b(T) = \gamma_b(S_T)$, and the shadow tree can be found in linear time using the method described in Chapter 2, it is enough to give algorithms for shadow trees.

4.1 Multipacking

We first introduce some new notation to help illustrating the algorithm. For a shadow tree, let a diametrical path be $P = v_0, v_1, ..., v_d$. Let l be the largest subscript such that v_l has degree 3 in S_T. The component of $S_T - \{v_0v_1, v_1v_2, ..., v_{d-1}v_d\}$ containing v_l is a path, which we denote by $v_l, u_1, u_2, .., u_h$. Let $e = l + h$. Note that $e \leq d$ because P is a diametrical path. Similarly, $l - h \geq 0$. The shadow of u_h extends from v_{l-h} to $v_e = v_{l+h}$.

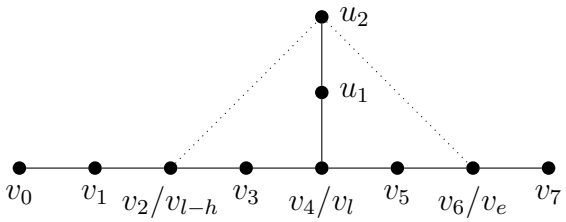


Figure 4.1: An illustration of the notation

The algorithm and its correctness follow from a sequence of five lemmas. Our approach is a simplification of Teshima's [26]. The lemmas that follow are similar to hers.

Lemma 4.1.1. [26] *Let M be a multipacking of the tree T such that $|M \cap \{v_l, v_{l+1}, ..., v_d\}| = i$ and $|M \cap \{u_1, u_2, ..., u_h\}| = j$. Then there is a multipacking M' such that $|M'| = |M|$ where $M' \cap \{v_l, v_{l+1}, ..., v_d\} = \{v_d, v_{d-3}, ..., v_{d-3i+3}\}$, $M' \cap \{u_1, u_2, ..., u_h\} = \{u_h, u_{h-3}, ..., u_{h-3j+3}\}$ and $M \cap (V - \{v_l, v_{l+1}, ..., v_d, u_1, u_2, ..., u_h\}) = M' \cap (V - \{v_l, v_{l+1}, ..., v_d, u_1, u_2, ..., u_h\})$.*

Informally, the lemma states that the vertices of M belonging to the paths $v_l, v_{l+1}, ..., v_d$ and $u_1, u_2, ..., u_h$ can be assumed to be as far as from v_l as possible. This means that the multipacking can be "pushed" towards one end of the tree. We call the multipacking M' in Lemma 4.1.1 a *pushed multipacking*.

Lemma 4.1.2. [26] *If $e = d$, then $mp(T) = mp(T - u_h)$.*

Proof: After deleting u_h from T, $v_{d-1}v_d$ and $v_{l-h}v_{l-h+1}$ may become new split-edges. We show this does not happen. If $v_{d-1}v_d$ is a split-edge, then v_d will be a singleton part, thus $v_{d-1}v_d$ is not a new split-edge. If $v_{l-h}v_{l-h+1}$ is a new split-edge, then the component containing v_l will have an odd length diametrical path, thus $v_{l-h}v_{l-h+1}$ is not a new split-edge. Hence, after deleting u_h from the tree, there are no new split-edges. By Theorem 2.2.11, since $diam(T - u_h) = diam(T)$ and the two trees have the same largest split sets, we have $\gamma_b(T) = \gamma_b(T - u_h)$. Since $\gamma_b(T) = mp(T)$, so we have $mp(T) = mp(T - u_h)$. □

Lemma 4.1.3. [26] *Suppose $d - e = 1$. If the tree T' is constructed from T by the following rule*

(i): If $deg(v_{l-1}) = 2$, let $T' = T - v_l u_1 + v_{l-1}u_1$,

(ii): If $deg(v_{l-1}) = 3$, let $T' = T - \{u_1, u_2, ..., u_h\}$,

then $mp(T) = mp(T')$. *Further, if* M' *is a maximum pushed multipacking of* T' *and* $v_l \in M'$, *then*

$$M = \begin{cases} M' - v_l + v_{l-1} & \text{if } u_2 \in M', \\ M' - \{v_r, v_l\} + \{v_{l-1}, u_2\} & \text{if } u_2 \notin M', \end{cases}$$

where v_r *be the rightmost vertex among* $v_0, v_1, \ldots, v_{l-3}$ *which is in* M', *is a maximum multipacking of* T.

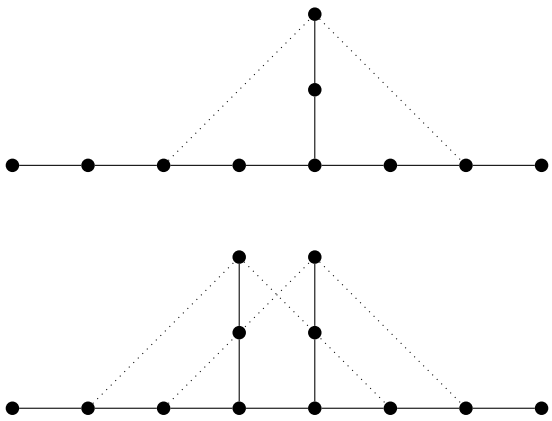


Figure 4.2: Case 1 and Case 2 trees

Proof: Let S be a maximum split-set in T. We first show that S is also a maximum split-set in T'. It then follows that $mp(T) = mp(T')$ since $diam(T) = diam(T')$.

Note that $v_{l-h-1}v_{l-h}$ cannot be a split-edge in T as $d - (l - h) = 2h + 1$ is odd. Thus all split-edges in T are split-edges in T'. The only candidate for a split-edge of T' which is not a split-edge of T is $v_e v_{e+1}$. But $d - e = 1$, so it cannot be a split-edge. Therefore S is a maximum split-set in T'.

Now let M' be a maximum pushed multipacking in T'. Suppose $v_l \in M'$. If $u_2 \in M'$, then the distance 2 constraint at v_{l-1} implies $l-3 < 0$ or $v_{l-3} \notin M'$. It now follows that $M = M' - v_l + v_{l-1}$ is a multipacking of T. Suppose $u_2 \notin M'$. If no vertex among $v_0, v_1, \ldots, v_{l-3}$ is in M', then since no vertex of M' is distance 3 or less from v_{l-1}, the set $M = M' - v_l + v_{l-1}$ is a multipacking of T. Otherwise, let v_r be the rightmost vertex among $v_0, v_1, \ldots, v_{l-3}$ which is in M'. Then $M = M' - \{v_r, v_l\} + \{v_{l-1}, u_2\}$ is a maximum multipacking of T (as in Lemma 4.1.1). □

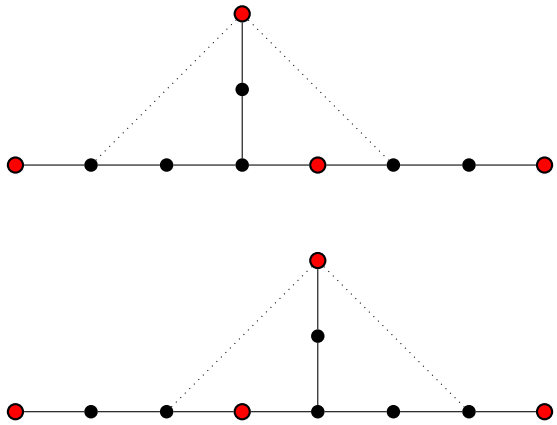


Figure 4.3: The multipacking in T' and T

Lemma 4.1.4. [26] *If $d - e = 2$, then $mp(T) = mp(T - v_d - v_{d-1} - v_{d-2}) + 1$.*

Proof: Take a maximum split-set M in $T - v_d - v_{d-1} - v_{d-2}$. In the tree T, the only possible new split-edges are $v_{d-3}v_{d-2}, v_{d-2}v_{d-1}, v_{d-1}v_d$. The edge $v_{d-3}v_{d-2}$ is not a split-edge because deleting it violates the condition about the diametrical path being preserved. The edges $v_{d-2}v_{d-1}, v_{d-1}v_d$ are not split-edges because deleting either one will leave a part of odd diameter or a singleton. Thus M is also a maximum split-set in T. Since $diam(T) = diam(T - v_d - v_{d-1} - v_{d-2}) + 2$, by Theorem 2.2.2, $\gamma_b(T) = \gamma_b(T - v_d - v_{d-1} - v_{d-2}) + 1$. Thus we have $mp(T) = mp(T - v_d - v_{d-1} - v_{d-2}) + 1$. □

Lemma 4.1.5. [26] *If $d - e > 2$, then $mp(T) = mp(T - v_d - v_{d-1} - v_{d-2}) + 1$.*

Proof: Take a maximum split-set M in $T - v_d - v_{d-1} - v_{d-2}$. Then it is clear that $M + v_{d-3}v_{d-2}$ will be a maximum split-set of T. Since $diam(T) = diam(T - v_d - v_{d-1} - v_{d-2}) + 3$ and the cardinality of the maximum split-set is also increased by 1, by Theorem 2.2.2, we have $\gamma_b(T) = \gamma_b(T - v_d - v_{d-1} - v_{d-2}) + 1$. As a result $mp(T) = mp(T - v_d - v_{d-1} - v_{d-2}) + 1$. □

We now build an algorithm to find a maximum multipacking. The lemmas allow us to deal with smaller and smaller trees until we reach a single vertex, P_2 or P_3.

Algorithm 3. *Algorithm to find a maximum multipacking for shadow trees.*

Input: A shadow tree T.

Output: A maximum multipacking of tree T.

Initially $M = \emptyset$ and $S = \emptyset$.

Step 1
Repeat until $T = \emptyset$.
If T *is a path,*
then
$M \cup \{v_d\} \to M$,
If $d \leq 2$,
then
$\emptyset \to T$.
else
$T - \{v_d\} - \{v_{d-1}\} - \{v_{d-2}\} \to T$.
else
Compute e,
If $d = e$,
then
$T - \{u_h\} \to T$,
If $deg(v_{l-1}) = 3$,
then
$T - \{u_1, u_2, ..., u_{d-1}\} \to T$.
Else if $d - e = 1$,
then
If $deg(v_{l-1}) = 2$,
then
$T - v_l u_1 + v_{l-1} u_1 \to T$,
Else if $deg(v_{l-1}) = 3$,
then
$T - \{u_1, u_2, ..., u_h\} \to T$,
$S \cup \{(v_l, v_{l-1})\} \to S$.
Else if $d - e = 2$,
then
$T - \{v_d\} - \{v_{d-1}\} - \{v_{d-2}\} \to T$ *and* $M \cup \{v_d\} \to M$.
The diametrical path is now $v_0, v_1, ..., v_l, u_1, u_2, ..., u_h$.
Update the shadow tree.
Else if $d - e > 2$,
then
$T - \{v_d\} - \{v_{d-1}\} - \{v_{d-2}\} \to T$ *and* $M \cup \{v_d\} \to M$.

Step 2:

For all $(u, v) \in S$. *If* $u \in M$, $M - \{u\} + \{v\} \to M$.

Theorem 4.1.6. *The set* M *constructed by Algorithm 3 is a maximum multipacking of the shadow tree. The algorithm halts after* $O(n)$ *steps, where* $n = |V(S_T)|$.

Proof: Correctness of the algorithm follows from Lemma 4.1.1 through Lemma 4.1.5. It is sufficient to prove that the algorithm is linear. For each vertex in the tree, we can assign several variables.

- Previous: the previous vertex.
- Next : the next vertex.
- Branch: the branch vertex adjacent to it, if any.
- Bend: the end of its branch, only applies to vertices with degree 3.
- Bcount: the length of the branch.

For the whole tree, we have two variables: Start and End. These two variables indicate v_0 and v_d, respectively.

First we find the shadow tree of the original tree; this takes $O(n)$ time. If the tree is a path, it is easy to see the algorithm halts after $O(n)$ steps.

If $d = e$, we just need to update the Bcount and Bend variable corresponding to v_l.

If $d - e = 1$, we need to update the Bend and Bcount variables for one or both of v_l and v_{l-1}.

If $d - e = 2$, we just need to switch the Next variable and Branch variable of v_l. We also update the Bend and Bcount variables and the End variable for the tree.

If $d - e = 3$, we just need to update the End variable of the tree.

Once the shadow tree is found, each operation takes $O(1)$ time and reduces the number of vertices in the tree. Since it is clear that step 2 of the algorithm runs in linear time, overall the algorithm is linear. □

4.2 Broadcast domination

We first introduce another piece of notation. Let s be the largest integer, if it exists, such that $s < l$ and the edge $v_{s-1}v_s$ is not under a shadow. If such an integer s does not exist, then we define s to be zero.

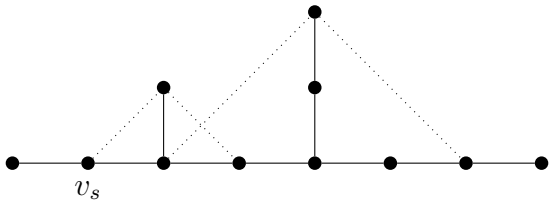


Figure 4.4: An illustration of v_s

Our algorithm for broadcast domination arises from the three lemmas given below.

Lemma 4.2.1. *If $d - e \geq 2$, then $\gamma_b(T) = \gamma_b(T - v_d - v_{d-1} - v_{d-2}) + 1$.*

Proof: Immediate from Lemma 4.1.4 and Lemma 4.1.5. □

Lemma 4.2.2. *If $d - e < 2$ and if $s = 0$ or 1, then T is radial.*

Proof: The only possible split-edges in T are $v_e v_{e+1}$ and v_0, v_1. Removing either of the edge will split one part of the tree to be a singleton vertex. By Theorem 2.2.3, T is radial. □

Lemma 4.2.3. *If $d - e < 2$ and $s > 1$, then*

(i): If $d - s$ is even, split the graph by $v_{s-1}v_s$.

(ii):If $d - s$ is odd, split the graph by $v_{s-2}v_{s-1}$.

Let T_s be the component containing v_s after e_s is deleted. Then T_s is radial and $\gamma_b(T) = \gamma_b(T_s) + \gamma_b(T - T_s)$.

Proof: We first prove that T_s is radial. Suppose T_s is not radial, then T_s must have an nonempty split-set. As we know, all edges under some shadow are not split-edges, so the only possible new split-edges can only be $v_{d-1}v_d$ and $v_{s-1}v_s$. The edge $v_{d-1}v_d$ is not a split-edge since v_d would be left as a singleton vertex. The edge $v_{s-1}v_s \in T_s$ only if Case (ii) happens, and we have $d - s$ is odd. Thus $v_{s-1}v_s$ is not a split-edge. By Theorem 2.2.3, T_s is radial.

Let the radius of T_s be r and M be the maximum split-set in $T - T_s$ (which might be empty). If e_s is a split-edge of T, then the maximum split-set of T is $M \cup \{e_s\}$. By Theorem 2.2.2, $\gamma_b(T - T_s) + \gamma_b(T_s) = \left\lceil \frac{diam(T-T_s)-|M|}{2} \right\rceil + r = \left\lceil \frac{(diam(T-T_s)+2r+1)-(|M|+1)}{2} \right\rceil = \left\lceil \frac{diam(T)-(|M|+1)}{2} \right\rceil = \gamma_b(T)$.

If e_s is not a split-edge, there are two cases. If e_s is not in any shadow, the only possibility that makes e_s not a split-edge is that $M = \emptyset$ and $diam(T - T_s)$ is odd. Thus $\gamma_b(T - T_s) + \gamma_b(T_s) = \frac{diam(T-T_s)+1}{2} + r = \frac{diam(T-T_s)+2r+1}{2} = \left\lceil \frac{diam(T)-0}{2} \right\rceil = \gamma_b(T)$.

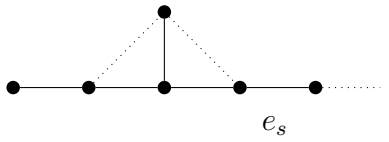


Figure 4.5: Sub-case 1

Suppose e_s is under some shadow, the diametrical path in $T - T_s$ is not a subpath of the diametrical path in T. Thus M is still a maximum split-set of T since we are increasing the diameter by $2r$ which is even. In this case we have $\gamma_b(T-T_s)+\gamma_b(T_s) = \left\lceil \frac{diam(T-T_s)-|M|}{2} \right\rceil + r = \left\lceil \frac{diam(T-T_s)+2r-|M|}{2} \right\rceil = \left\lceil \frac{diam(T)-|M|}{2} \right\rceil = \gamma_b(T)$. □

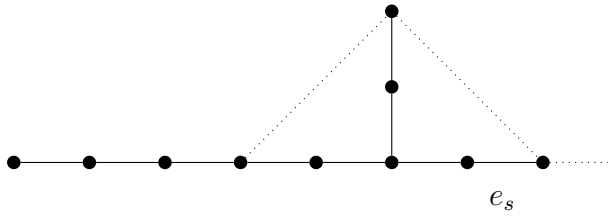


Figure 4.6: Sub-case 2

Similar to the algorithm for multipacking, we give an algorithm to find an optimal dominating broadcast. We will get pairs $((v_i), j)$ such that $v_i \in P$ and $j \in N$ to be the broadcasting vertices. Each pair $((v_i), j)$ means we select vertex v_i as a broadcasting vertex with strength j.

Algorithm 4. *Algorithm to find the minimum broadcast domination for shadow trees.*

Input: A shadow tree T.

Output: A set of pairs (v_i, j) such that broadcasting from v_i with strength j for each pair is an optimal broadcast.

Initially $B = \emptyset$.

Repeat until $T = \emptyset$.

If T is a single vertex,

then

$$B \cup \{(v_1, 1)\} \to B \text{ and } \emptyset \to T.$$

Else if T is P_2 or P_3,

then

$$B \cup \{(v_2, 1)\} \to B \text{ and } \emptyset \to T.$$

Else if $d - e \geq 2$,

then

$T - \{v_d\} - \{v_{d-1}\} - \{v_{d-2}\} \to T$ *and* $B \cup \{(v_{d-1}, 1)\} \to B$.

Else if $d - e < 2$ *and* $s = 0$ *or* 1

then

$B \cup \{(v_{d-r}, r\} \to B$, *where* r *is the radius of* T.

Else if $d - e < 2$ *and* $s > 1$,

then

If $d - s$ *is even,*

then

Delete the edge $v_{s-1}v_s$.

else

Delete the edge $v_{s-2}v_{s-1}$.

$T - T_s \to T$, *where* T_s *is the subtree containing* v_s.

$B \cup \{(v_{d-r}, r)\} \to B$, *where* r *is the radius of* T_s.

Theorem 4.2.4. *The set B gives the minimum dominating broadcast of the shadow tree. The algorithm halts after $O(n)$ steps.*

Proof: For each case, we are reducing the numbers of vertices in the tree by at least 1 and we need $O(1)$ time to process. So the algorithm is linear. The correctness of the algorithm follows from Lemma 4.2.1 through Lemma 4.2.3. □

Note that although every graph has an efficient optimal dominating broadcast, the dominating broadcast given by the algorithm may not be efficient. But we can always obtain an efficient broadcast by "merging" two broadcasts that overlap.

4.3 Illustration of the algorithms

This section focuses on finding a maximum multipacking and a minimum dominating broadcast of a given tree using the algorithms above. Let T be the tree in Figure 4.7.

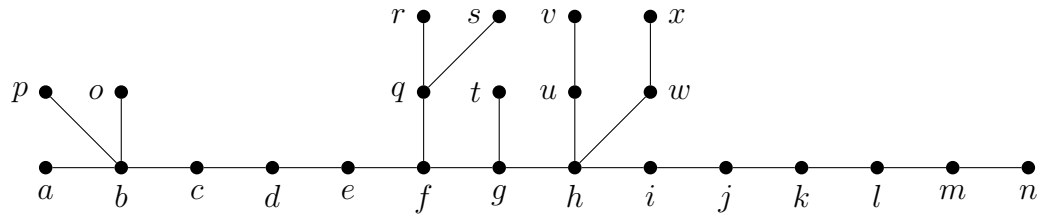


Figure 4.7: Tree T used to illustrate the algorithms

First we find the shadow tree T, S_T.

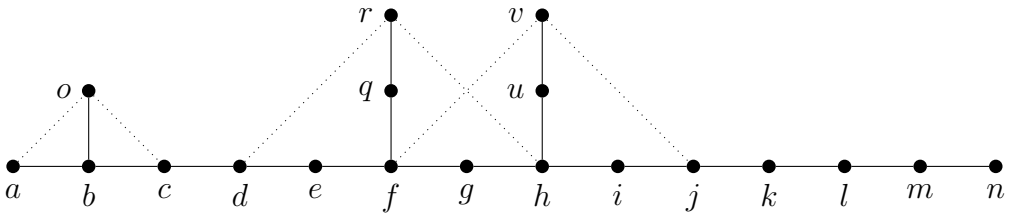


Figure 4.8: The shadow tree, S_T

We illustrate the algorithm to find the maximum multipacking first. Originally M and S are empty. Here we have $v_0 = a, v_1 = b, v_2 = c, v_3 = d, v_4 = e, v_5 = f, v_6 = g, v_7 = h, v_8 = i, v_9 = j, v_{10} = k, v_{11} = l, v_{12} = m, v_{13} = n$ and $u_1 = u, u_2 = v$. Since $d(v_d, v_e) = 13 - 9 = 4 \geq 2$. We insert n into M so that $M = \{n\}$ and $S = \emptyset$. We remove l, m, n from the tree.

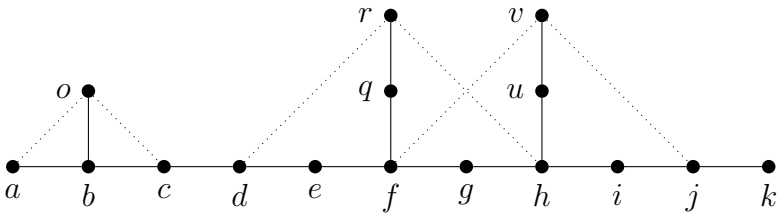


Figure 4.9: The tree after one reduction

Here we have $v_0 = a, v_1 = b, v_2 = c, v_3 = d, v_4 = e, v_5 = f, v_6 = g, v_7 = h, v_8 = i, v_9 = j, v_{10} = k$ and $u_1 = u, u_2 = v$. We have $d(v_d, v_e) = 10 - 9 = 1$. So we delete edge uh and insert the new edge gu. We now have $M = \{n\}$ and $S = \{(h, g)\}$.

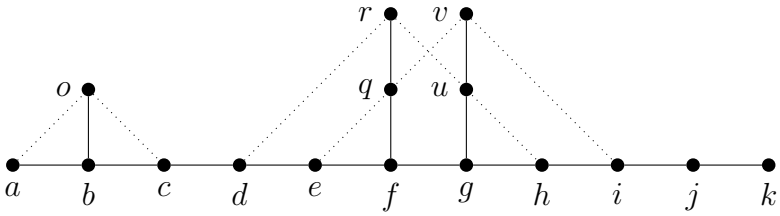


Figure 4.10: The tree after two reductions

Now we have $v_0 = a, v_1 = b, v_2 = c, v_3 = d, v_4 = e, v_5 = f, v_6 = g, v_7 = h, v_8 = i, v_9 = j, v_{10} = k, u_1 = u, u_2 = v$ and $d(v_d, v_e) = 10 - 8 = 2$. We put k into M,

so that now $M = \{n, k\}$ and $S = \{(h, g)\}$. Next, remove i, j, k from the tree. The diametrical path will become $abcdefguv$.

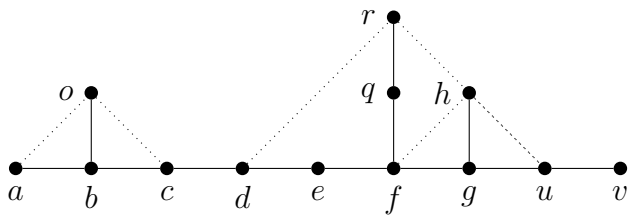


Figure 4.11: The tree after three reductions

To maintain a shadow tree, remove h from the the tree.

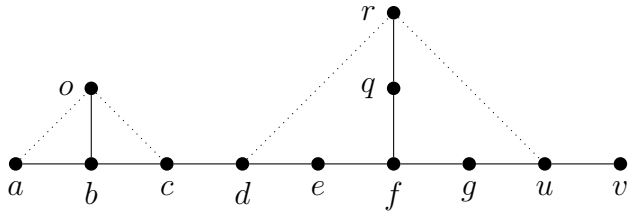


Figure 4.12: The updated shadow tree

Now $v_0 = a, v_1 = b, v_2 = c, v_3 = d, v_4 = e, v_5 = f, v_6 = g, v_7 = u, v_8 = v$ and $u_1 = q, u_2 = r$. Since we have $d(v_d, v_e) = 8 - 7 = 1$, the edge fq is removed and the edge eq is inserted. At this point $M = \{n, k\}$ and $S = \{(h, g), (f, e)\}$.

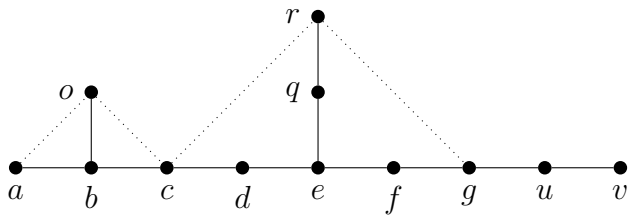


Figure 4.13: The tree after four reductions

We have $v_0 = a, v_1 = b, v_2 = c, v_3 = d, v_4 = e, v_5 = f, v_6 = g, v_7 = u, v_8 = v$ and $u_1 = q, u_2 = r$. Since $d(v_d, v_e) = 8 - 6 = 2$, in this step we delete g, u, v from the tree and insert v into M. Now $M = \{n, k, v\}$ and $S = \{(h, g), (f, e)\}$. The diametrical path becomes $abcdeqr$.

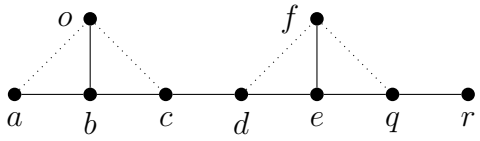


Figure 4.14: The tree after five reductions

Now $v_0 = a, v_1 = b, v_2 = c, v_3 = d, v_4 = e, v_5 = q, v_6 = r$ and $u_1 = f$. Since $d(v_d, v_e) = 6 - 5 = 1$, we remove edge ef from the tree and insert the new edge df. Now $M = \{n, k, v\}$ and $S = \{(h, g), (f, e), (e, d)\}$.

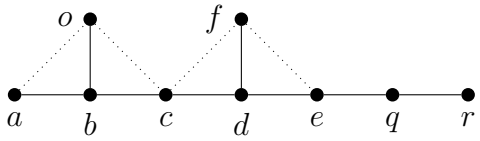


Figure 4.15: The tree after six reductions

We have $v_0 = a, v_1 = b, v_2 = c, v_3 = d, v_4 = e, v_5 = q, v_6 = r$ and $u_1 = f$. Since $d(v_d, v_e) = 6 - 4 = 2$, we delete e, q, r from the tree and insert r into M. Now $M = \{n, k, v, r\}$ and $S = \{(h, g), (f, e), (e, d)\}$. The new diametrical path is $abcdf$.

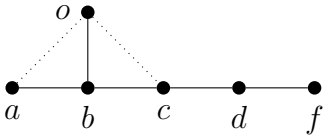


Figure 4.16: The tree after seven reductions

We have $v_0 = a, v_1 = b, v_2 = c, v_3 = d, v_4 = f$ and $u_1 = o$. Since $d(v_d, v_e) = 4 - 2 = 2$, we delete c, d, f from the tree and insert f into M. Now $M = \{n, k, v, r, f\}$ and $S = \{(h, g), (f, e), (e, d)\}$. The new diametrical path is abo.

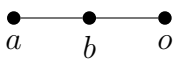


Figure 4.17: The tree after eight reductions

The remaining graph is P_3. We insert o into M.

As a result $M = \{n, k, v, r, f, o\}$ and $S = \{(h, g), (f, e), (e, d)\}$. In Step 2, since $(f, e) \in S$, we need to remove f and add e into M. Thus the algorithm gives a maximum multipacking $M = \{n, k, v, r, e, o\}$.

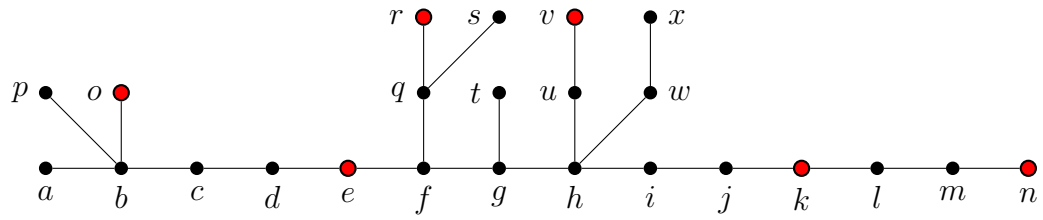


Figure 4.18: The maximum multipacking produced by the algorithm

Now we want to find a minimum dominating broadcast. Starting from S_T with originally $B = \emptyset$, we have $v_0 = a, v_1 = b, v_2 = c, v_3 = d, v_4 = e, v_5 = f, v_6 = g, v_7 = h, v_8 = i, v_9 = j, v_{10} = k, v_{11} = l, v_{12} = m, v_{13} = n$. Since $d(v_d, v_e) = 13 - 9 = 4 \geq 2$, we remove l, m, n from the tree and insert $(m, 1)$ into B. Now $B = \{(m, 1)\}$.

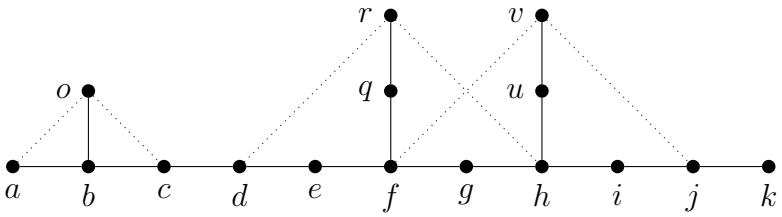


Figure 4.19: The tree remaining after broadcasting from m

Now $v_0 = a, v_1 = b, v_2 = c, v_3 = d, v_4 = e, v_5 = f, v_6 = g, v_7 = h, v_8 = i, v_9 = j, v_{10} = k$. We have $d(v_d, v_e) = 10 - 9 = 1$ and $v_s = v_3$. Since $d(v_s, v_e) = 7$ is odd, we split the graph by deleting the edge $v_{s-2}v_{s-1} = bc$ and insert $(g, 4)$ into B. Now $B = \{(m, 1), (g, 4)\}$.

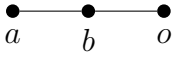


Figure 4.20: The tree remaining after broadcasting from g

The remaining graph is just P_3, so we insert $(b, 1)$ into B. Thus $B = \{(j, 1), (g, 4), (b, 1)\}$.

As a result, the algorithm will give $B = \{(j, 1), (g, 4), (b, 1)\}$. A minimum dominating broadcast will look like:

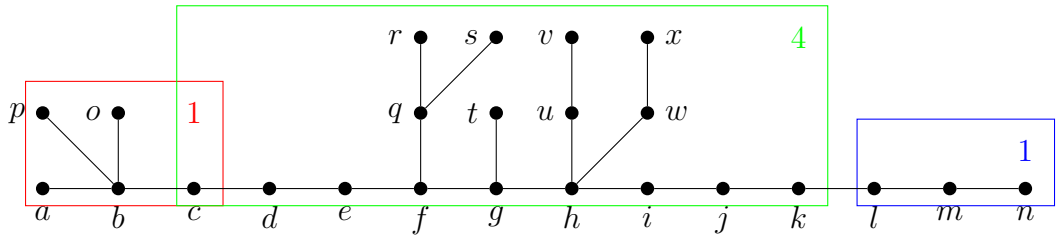


Figure 4.21: The minimum dominating broadcast produced by the algorithm

If we want to find an efficient minimum broadcast, we can merge the red and green broadcast and then broadcast from f with strength 5:

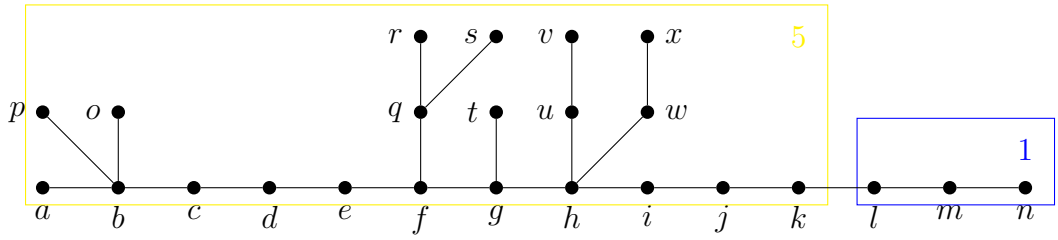


Figure 4.22: An efficient optimal dominating broadcast

Chapter 5

k-Broadcast Domination and k-Multipacking

In this chapter we discuss generalized broadcast domination and multipacking.

5.1 Definition

In the broadcast domination problem, each vertex of the graph G is required to hear at least one broadcast. We now generalize this idea by requiring that each vertex hears some fixed number k of different broadcasts.

Let G be a graph and assign to each vertex $v \in V(G)$ a strength $f(v) \in \{1, 2, ..., diam(G)\}$. Let $V_f^+ = \{v : f(v) > 0\}$, if for each vertex $u \in V(G)$, there exist different vertices $v_1, v_2, ..., v_k \in V_f^+$ such that $d(u, v_i) \leq f(v_i)$ for each i, then f is called a *dominating k-broadcast*. The *cost* of a dominating k-broadcast is the quantity $\sum_{v \in V} f(V)$. The minimum cost of a dominating k-broadcast is the *k-broadcast domination number* of G, and is denoted by $\gamma_{b_k}(G)$. It is easy to see that 1-broadcast domination is the same as broadcast domination.

We can imagine k-broadcast domination to be a model of the following situation. A company wants to build radio towers in some cities, and wants people in every city to be able to hear the radio even if some $(k-1)$ of the towers are not functioning. Radio towers are fairly large, so each city will allow only one tower to be located there. Furthermore, towers that broadcast a stronger signal cost more than those with a weaker signal. The question is where to build the towers in order to guarantee the desired coverage and minimize the overall cost.

The integer programming formulation of k-broadcast domination is different from that for 1-broadcast domination. We need to guarantee that every vertex can broadcast at most once. Hence we need to add constraints that prevent a vertex from broadcasting twice. This is accomplished using the restriction matrix R. It has the same dimension as A^* and each row and column represent the same thing as A^*. In the matrix, the entries (i, x_{ik}) are -1 and all other entries 0. Then we define $A_r^* = \begin{bmatrix} A^* \\ R \end{bmatrix}$. The integer program $B_k(G)$ for finding $\gamma_{b_k}(G)$ is described below:

Minimize $\sum_{i=1}^{n} hx_{ih}$,

subject to

$$A_r^* x \geq \begin{bmatrix} k \\ -1 \end{bmatrix},$$

$x_{ih} \in \{0, 1\}$ for each vertex i and integer $h \in \{0, 1, 2, ..., diam(G)\}$.

The vector $\begin{bmatrix} k \\ -1 \end{bmatrix}$ gives the restriction that every vertex can hear k copies of broadcast and also no vertices are broadcasting twice.

To define k-multipacking we need to take the dual of $B_k(G)$. The integer program of k-multipacking $M_k(G)$ is:

Maximize $\sum_{ji=1}^{n} k \cdot c_j - r'_j$,

subject to

$$\sum_{d(i,j) \leq h} c_j - r_i \leq h \text{ for each vertex } i \text{ and integer } h \in \{0, 1, 2, ..., diam(G)\}$$

$c_j, r_j \in \mathbb{N}$ for each j.

For each vertex we are assigning two different values, c_j and r_j. The value c_j counts how many times a certain vertex is chosen into the k-multipacking; here it is possible to choose a single vertex many times because of the existence of r_j. The variable r_j is the relaxation value on vertex v_j. It allows the restriction on that vertex to be relaxed by r_j.

From the integer program, we see that a *k-multipacking* is a pair (c, r), where $c : V \to N$ and $r : V \to N$ such that for every vertex $i \in V$ we have $\sum_{d(i,j) \leq h} c_j - r_i \leq h$ for each h. The *value* of a k-multipacking (c, r) is the quantity $\sum_{v \in V} k \cdot c(v) - r(v)$. The largest value of a k-multipacking of G is the *k-multipacking number* of G, and is denoted by $mp_k(G)$. If we take a maximum multipacking and consider it as a 1-multipacking, if we want to increase c_j, we must increase r_j by at least the

same amount, otherwise the original multipacking was not maximum. As a result, $mp_1(G) = mp(G)$.

Consider the tree shown in Figure 5.1. A dominating 2-broadcast is obtained by broadcasting with strength 1 from v_2 and strength 2 from v_1. This gives $\gamma_{b_2}(G) \leq 3$. On the other hand, we can describe the pair of vectors (c, r) by ordered pairs $(1, 0)$, $(0, 1)$, $(1, 0)$ and $(0, 0)$ of values at the vertices v_1, v_2, v_3, v_4 respectively. This gives a 2-multipacking with value 3, thus $mp_2(G) \geq 3$. By the Duality Theorem of Linear Programming, we have $\gamma_{b_2}(G) = mp_2(G) = 3$.

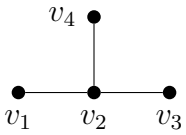


Figure 5.1: A tree with 4 vertices.

The following results are immediate consequences of the definition.

Proposition 5.1.1. *For a graph G, $\gamma_{b_k}(G) \geq mp_k(G)$.*

Proof: This result comes directly from the Duality Theorem of Linear Programming. □

Proposition 5.1.2. *For a graph G, $mp_k(G) \geq k \cdot mp(G)$.*

Proof: By the definition of k-multipacking, we can always assign c_i to be 1 for all the vertices included in a maximum multipacking and r_i to be 0 for all vertices. This gives $k \cdot mp(G)$ as a lower bound on the k-multipacking number. □

These two propositions and Corollary 3.2.5 imply the result below immediately.

Corollary 5.1.3. *If G is strongly chordal graph, then $\gamma_{b_k}(G) \geq k \cdot \gamma_b(G)$.*

Proposition 5.1.2 and Theorem 2.1.7 also leads to the result below.

Corollary 5.1.4. *For a graph G, $mp_k(G) \geq k \cdot \left\lceil \frac{diam(G)+1}{3} \right\rceil$.*

5.2 From 1-broadcast domination to 2-broadcast domination

There seems to be no straightforward relationship between 1-broadcast domination and k-broadcast domination. We will illustrate this by focusing the relationship between 1-broadcast domination and 2-broadcast domination.

Proposition 5.2.1. *For a graph G, $\gamma_b(G) < \gamma_{b_2}(G) \leq 3\gamma_b(G)$.*

Proof: The first inequality is trivial. We prove that $\gamma_{b_2}(G) \leq 3\gamma_b(G)$. Let f be a dominating broadcast with cost $\gamma_b(G)$.

We claim that each vertex in V_f^+ with degree at least 2 has a neighbour w with $f(w) = 0$. Suppose not, and let v be a vertex such that $N[v] \subseteq V_f^+$. The value of $f(v)$ cannot be 1 since all its neighbours are broadcasting. Take a neighbour w of v. If $f(w) > f(v)$, all the vertices that hear v will hear w. In this case, consider a new broadcast f' which is identical to f except that $f'(v) = 0$. The broadcast f' has smaller cost than f and is also a dominating broadcast, a contradiction. If $1 < f(w) \leq f(v)$, we can define a new broadcast f' which is identical to f except that $f'(v) = 0$ and $f'(w) = f(v) + 1$. Again, f' will be a dominating broadcast with smaller cost, a contradiction. If $f(w) = 1$, then since $f(v) > 1$, all of the vertices that hear w also here v. Now, the broadcast f' which is identical to f except that $f'(w) = 0$ is dominating and f' has smaller cost than f, a contradiction. The claim is proved.

By the claim, for each vertex $v \in V_f^+$, there is a vertex $v' \in N(v) \setminus V_f^+$. A 2-broadcast can be obtained by broadcasting from each $v \in V_f^+$ with strength f and, for each such v, broadcasting from some $v' \in N(v) \setminus V_f^+$ with strength $f(v) + 1$. The cost of this broadcast is $2f(v) + |V_f^+| \leq 3\gamma_b(G)$. □

We now find the 2-broadcast domination number for some specific classes of graphs.

Proposition 5.2.2. *For an integer k, $\gamma_{b_2}(P_{3k}) = 2k + 1$, $\gamma_{b_2}(P_{3k+1}) = 2k + 2$ and $\gamma_{b_2}(P_{3k+2}) = 2k + 2$.*

Proof: For P_{3k} it is easy to see that the 1-broadcast domination number is k and it can only be achieved by broadcasting with strength 1 from the middle vertex of every 3 vertices. Since this broadcast is unique, there cannot be a 2-broadcast with strength $2k$. A 2-broadcast with strength $2k+1$ can be obtained by broadcasting with strength 1 from the first vertex, and the second and third vertices of each consecutive triple.

For P_{3k+1}, broadcast with strength 1 from the first vertex, and the middle vertex of every 3 consecutive vertices following it. This will give us a 1-broadcast domination. Now do the same from the other side. By parity, no vertex broadcasts twice. This gives us a dominating 2-broadcast with cost $2k + 2$.

For P_{3k+2}, apply a similar strategy as for P_{3k+1}. Starting from the left and broadcast on the left end with strength 1. Then starting from the third vertex, broadcast with strength 1 from the middle vertex of every 3 vertices. Now do the same from right to left. This gives a dominating 2-broadcast with cost $2k+2$. □

The question arises naturally: is $\gamma_{b_2}(G) \geq 2 \cdot \gamma_b(G)$? While it is true that $\gamma_{b_2}(G) \geq 2\gamma_b(G)$ for strongly chordal graphs, this bound does not hold for all graphs. Here we give some examples that have $\gamma_{b_2}(G) = 2 \cdot \gamma_b(G) - 1$. It is easy to see that $\gamma_b(K_{2,n}) = 2$ and $\gamma_b(K_{n_1,n_2,\ldots,n_k}) = 2$ while $\gamma_{b_2}(K_{2,n}) = 3$ and $\gamma_{b_2}(K_{n_1,n_2,\ldots,n_k}) = 3$ ($k \geq 3$). And the graph G in Figure 5.2 has $\gamma_b(G) = 3$ and $\gamma_{b_2}(G) = 5$.

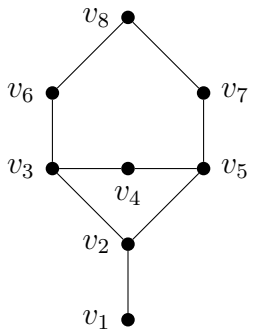


Figure 5.2: A graph G which has $\gamma_{b_2}(G) = 2 \cdot \gamma_b(G) - 1$.

It is easy to see that $\gamma_b(G) = 3$. A 2-broadcast is obtained by broadcasting from v_3 and v_5 with strength 2 and from v_8 with strength 1.

Proposition 5.2.3. *For each integer k, $\gamma_{b_2}(C_{3k+1}) = 2k+1$.*

Proof: Let the cycle be $v_1, v_2, ..., v_{3k+1}, v_1$. The broadcast vertices will be v_1, v_2, v_3, v_{3i-1}, v_{3i} where $1 < i \leq k$ and each is broadcasting with strength 1. This is a dominating 2-broadcast of the required strength. It remains to show there is no dominating 2-broadcast with smaller strength.

Each vertex on a cycle has 2 neighbours, and each vertex can broadcast to $2m+1$ vertices if broadcasting with strength m. Suppose we have a dominating 2-broadcast with total cost $2k$ which is broadcasting from vertices $v_1, v_2, ..., v_x$ with strength $m_1, m_2, ..., m_x$, respectively. Thus $\sum_{1 \leq i \leq x} m_i = 2k$ and $\sum_{1 \leq i \leq x} (2m_i + 1) \geq 6k+2$. But $\sum_{1 \leq i \leq x} (2m_i+1) = 2 \cdot \sum_{1 \leq i \leq x} m_i + |V_f^+| = 4k + |V_f^+|$. Thus we must have $4k + |V_f^+| \geq 6k+2$ or $|V_f^+| \geq 2k+2$. However $|V_f^+| \leq \sum_{1 \leq i \leq x} m_i = 2k$, a contradiction.

It now follows that $\gamma_{b_2}(C_{3k+1}) = 2k+1$. □

Theorem 5.2.4. *Given any integer k, there exists a connected graph with $\gamma_{b_2}(G) \leq 2 \cdot \gamma_b(G) - k$.*

Proof: In the proof we will call the graph in Figure 5.3 a *block*, and denote it by B. It is easy to see that the radius of B is 2 and $2 \cdot \gamma_b(B) = \gamma_{b_2}(B) = 2$. Another graph we need in our construction is C_4. It has radius 2 and $2 \cdot \gamma_b(C_4) = \gamma_{b_2}(C_4) - 1 = 3$.

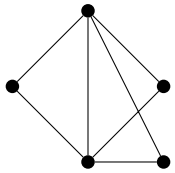


Figure 5.3: The block B.

Let k be an integer. Let H_k be the graph constructed from C_4 and k disjoint copies $B_1, B_2, ..., B_k$ of B as shown in Figure 5.4.

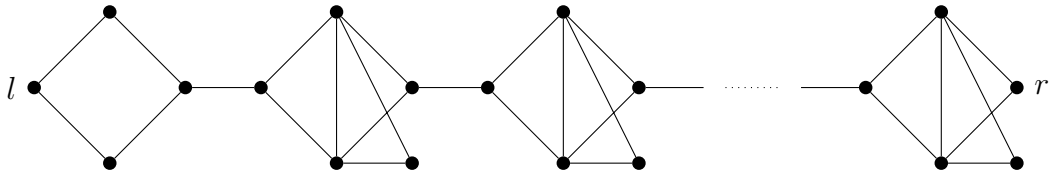


Figure 5.4: H_k.

Let G_k be constructed from $k-1$ disjoint copies of H_k and one copy of C_4 as shown in Figure 5.5.

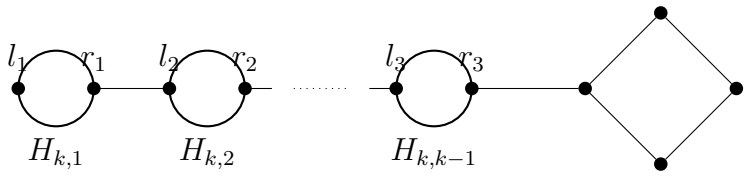


Figure 5.5: G_k

In G_k we have a total of $k \cdot (k-1)$ copies of B and k copies of C_4, so $\gamma_{b_2}(G_k) \leq 2k \cdot (k-1) + 3k = 2k^2 + k$. Thus it is sufficient to prove that $\gamma_b(G_k) = k^2 + k$. Suppose there exists a dominating broadcast f with cost less than $k^2 + k$ and there is a vertex v which broadcasts with strength $f(v) > 2$. Suppose the broadcast from v is heard by all vertices in x copies of C_4. Then by the definition it will be heard by all vertices of at least $k \cdot (x-1) + n$ copies of B where $n \in \{0, 1, 2, ..., 2x-2\}$. So

$f(v) \geq \frac{3}{2}(x + k \cdot (x-1) + n - 1) + 1$. On the other hand, we can broadcast to the vertices which hear v using a total cost of $2x + k \cdot (x-1) + n \leq f(v)$, since $k \geq 2$. This contradicts the minimality of f, so no vertex broadcasts with strength more than 2 in an optimal broadcast in G_k. So the optimal dominating broadcast has cost 2 on each C_4 and cost 1 on each B. This gives a total cost of $\gamma_b(G) = k^2 + k$.

As a result, the graph G_k has $\gamma_{b_2}(G_k) \leq 2 \cdot \gamma_b(G) - k$. This completes the proof.□

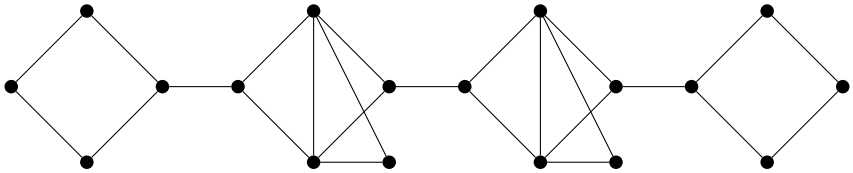


Figure 5.6: Example when k is 2.

There is also no additive upper bound with respect to any constant.

Theorem 5.2.5. *Given any integer k, there exists a connected graph G with $\gamma_{b_2}(G) = 2 \cdot \gamma_b(G) + k$.*

Proof: In the proof we will use the graph $3T_4$ shown in Figure 5.7.

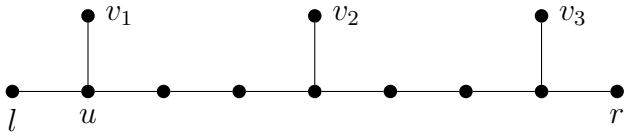


Figure 5.7: $3T_4$.

The graph G_k made by connecting k copies of $3T_4$ in series has $\gamma_{b_2}(G_k) = 2 \cdot \gamma_b(G_k) + k$. It is easy to see that $2 \cdot \gamma_b(3T_4) = \gamma_{b_2}(3T_4) - 1 = 6$ and $\gamma_b(G_k) = 3k$. It is enough to show that $\gamma_{b_2}(G_k) \geq 7k$. A 2-multipacking of G_k with total value $7k$ is obtained by choosing l, v_1, v_2, v_3 and relaxing by 1 at u in each copy of $3T_4$. By Duality Theorem of Linear Programming, $\gamma_{b_2}(G_k) \geq mp_2(G + k) \geq 7k$. This completes the proof. □

5.3 2-Broadcast domination on trees

Unlike broadcast domination, shadow trees are no longer helpful for 2-broadcast domination. For the tree T given in Figure 5.8, we have $\gamma_{b_2}(T) = 7$. However if we remove

u and v to construct the shadow tree, $\gamma_{b_2}(T-u,v)=6$. We can broadcast with strength 3 from the center and 1 from u' and 1 each for the two copies of P_3 on the left and right of the central vertex. Without the help of the shadow tree, it is much harder to work on 2-broadcast domination than on broadcast domination.

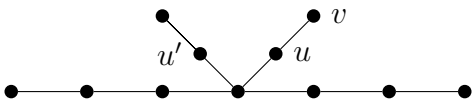


Figure 5.8: 2-Broadcast shadow tree example.

Herke and Mynhardt showed that $\gamma_b(G) \leq \lceil \frac{n}{3} \rceil$ [16]. We conclude the thesis with a similar result for 2-broadcast domination.

Let T be a tree and P be a diametrical path of T. Let x be a leaf of the shadow tree of T which is not on P. Let v_x be the closest vertex to x which is on P. Set $k_x = d(x, v_x)$. The *shadow of* x is the subtree of T induced by $N_{k_x}[v_x]$. By definition of the shadow tree, every vertex not on P belongs to some shadow.

We use shadows to define the *s-decomposition* of T. Let F be the forest induced by the set of edges of T which belong to some shadow. Each component of F is called a *tree part*. A *path part* is a component of the collection of paths induced by the vertices in $V(T) \setminus V(F)$.

Note that tree parts are defined in terms of edge induced subgraphs, and path parts are defined in terms of vertex induced subgraphs. Path parts can only be adjacent to tree parts, but tree parts can be adjacent to other tree parts.

We use Figure 5.9 to illustrate the definitions. The diametrical path P is v_0, v_1, v_2, $..., v_8$. There are two tree parts after forming the s-decomposition. The subtree induced by $\{v_0, v_1, v_2, v_9, v_{10}\}$ will become a tree part and the subtree induced by $\{v_3, v_4, v_5, v_6, v_{11}, v_{12}\}$ is another tree part. The subtree induced by $\{v_7, v_8\}$ will be the only path part after the s-composition is formed.

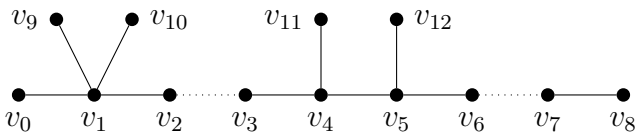


Figure 5.9: s-decomposition.

Theorem 5.3.1. *For any tree T of order $n > 1$, $\gamma_{b_2}(T) \leq \lceil \frac{4n}{5} \rceil$.*

Proof: The proof is by induction on the number of vertices.

By Proposition 5.2.2, the statement holds if T is a path, P_n. Note that the ceiling is not needed when $n \geq 5$.

Suppose the s-decomposition of T consists of a single tree part. There are two cases depending on whether T has a unique center. If it does, then $n \geq 3 \cdot rad(T) + 1$ while $\gamma_{b_2}(T) \leq rad(T) + (rad(T) + 1)$ since we can broadcast with strength $rad(T)$ from the central vertex and strength $rad(T) + 1$ from a neighbour of it. In this case we have

$$\gamma_{b_2}(T) \leq 2 \cdot rad(T) + 1 \leq \frac{4}{5}(3 \cdot rad(T) + 1) \leq \frac{4n}{5}.$$

Otherwise, T is bicentral. We have $rad(T) \geq 2$ and $n \geq 3 \cdot rad(T) \geq 6$. By broadcasting with strength $rad(T)$ from each center we obtain

$$\gamma_{b_2}(T) \leq 2 \cdot rad(T) \leq \frac{12 \cdot rad(T)}{5} \leq \frac{4n}{5}.$$

Hence the statement holds when the s-decomposition consists of a single tree part. In particular, the statement is true for all trees with four or fewer vertices. Suppose it holds for all trees with k or fewer vertices, for some $k \geq 4$. Consider a tree T with $k + 1$ vertices.

By our work above, we can assume the s-decomposition of T has at least two parts. By definition of the s-decomposition, some part X is adjacent to only one other part.

First suppose that X is a tree part with n_x vertices. Then, by our argument above, $\gamma_{b_2}(X) \leq \frac{4n_x}{5}$. By the induction hypothesis, for thee tree $T - X$ we have $\gamma_{b_2}(T - X) \geq \left\lceil \frac{4(n-n_x)}{5} \right\rceil$. Therefore,

$$\gamma_{b_2}(T) \leq \frac{4n_x}{5} + \left\lceil \frac{4(n - n_x)}{5} \right\rceil \leq \left\lceil \frac{4n_x}{5} + \frac{4(n - n_x)}{5} \right\rceil = \left\lceil \frac{4n}{5} \right\rceil.$$

The same argument as above works when X is a path part with at least 5 vertices.

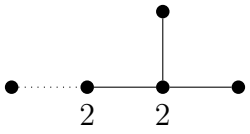
Suppose that X is a path part with four vertices. The part adjacent to X is a tree part, and the vertex v in the tree part which is adjacent to X has degree 2 in T (if it had degree 3 or more, its neighbour in X would be under a shadow). Since $X \cup \{v\}$ is a path on 5 vertices, applying the induction hypothesis to the subtree induced by $V(T) - (V(X) \cup \{v\})$ gives

$$\gamma_{b_2}(T) \leq 4 + \left\lceil \frac{4}{5}(n - 5) \right\rceil \leq \left\lceil \frac{4n}{5} \right\rceil.$$

It remains to consider the cases where X is a path on at most three vertices. Our strategy is to combine X with the adjacent tree part Y and show $\gamma_{b_2}(T) \leq \frac{4}{5}|V(X \cup Y)|$. The result then follows as above.

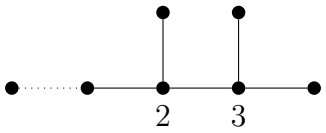
Case 1: X is a path with one vertex and Y has a unique center. By our earlier arguments, we know that $V(Y) \geq 3 \cdot rad(Y) + 1$. We can broadcast $rad(Y) + 1$ on the two new centers after combining X and Y and we have

$$\gamma_{b_2}(X \cup Y) \leq 2(rad(Y) + 1) \leq \frac{12 \cdot rad(Y) + 8}{5} \leq \frac{4}{5}|V(X \cup Y)|.$$



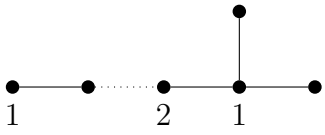
Case 2: X is a path with one vertex and Y is bicentral. Then $X \cup Y$ has a unique center and at least $3 \cdot rad(Y) + 1 \geq 7$ vertices. As before, we have $rad(X \cup Y) = rad(Y)$ and we can broadcast with cost $2 \cdot rad(Y) + 1$. We have

$$\gamma_{b_2}(X \cup Y) \leq 2 \cdot rad(Y) + 1 \leq \frac{4}{5}(3 \cdot rad(Y) + 1) \leq \frac{4}{5}|V(X \cup Y)|.$$



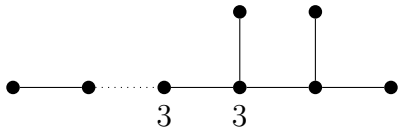
Case 3: X is a path with two vertices and Y has a unique center. Then $X \cup Y$ has at least $3 \cdot rad(Y) + 3$ vertices. By broadcasting strength $rad(Y)$ from the center of Y and strength $rad(Y) + 1$ from its neighbour closest to X and strength 1 from a vertex of X, we obtain

$$\gamma_{b_2}(X \cup Y) \leq 2 \cdot rad(Y) + 2 \leq \frac{4}{5}(3 \cdot rad(Y) + 3) \leq \frac{4}{5}|V(X \cup Y)|.$$



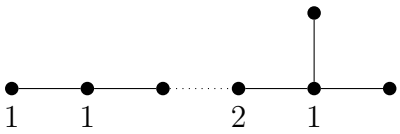
Case 4: X is a path with two vertices and Y is bicentral. Then $X \cup Y$ is bicentral and has at least $3 \cdot rad(Y) + 2$ vertices. Since a dominating 2-broadcast can be obtained by broadcasting with strength $rad(Y) + 1$ from the two centers of $X \cup Y$, and noting $rad(Y) \geq 2$, we have

$$\gamma_{b_2}(X \cup Y) \leq 2 \cdot rad(Y) + 2 \leq \frac{4}{5}(3 \cdot rad(Y) + 2) \leq \frac{4}{5}|V(X \cup Y)|.$$



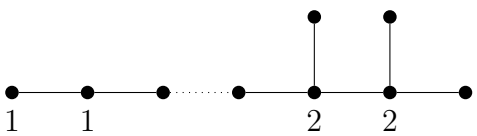
Case 5: X is a path with three vertices and Y has a unique center. Then $X \cup Y$ has at least $3 \cdot rad(Y) + 4$ vertices. By broadcasting with strength $rad(Y)$ from the central vertex of Y, strength $rad(Y)+1$ from its neighbour closest to X and strength 1 from each of the two vertices of X farthest from Y, we obtain

$$\gamma_{b_2}(X \cup Y) \leq 2 \cdot rad(Y) + 3 \leq \frac{4}{5}(3 \cdot rad(Y) + 4) \leq \frac{4}{5}|V(X \cup Y)|.$$



Case 6: X is a path with three vertices and Y is bicentral. Then $X \cup Y$ has at least $3 \cdot rad(Y) + 3$ vertices. By broadcasting with strength $rad(Y)$ from each of the two centers of Y, and strength 1 from each of the two vertices of X farthest from Y, we have

$$\gamma_{b_2}(X \cup Y) \leq 2 \cdot rad(Y) + 2 \leq \frac{4}{5}(3 \cdot rad(Y) + 3) \leq \frac{4}{5}|V(X \cup Y)|.$$



All cases have been considered, and it now follows by induction that $\gamma_{b_2}(T) \leq \lceil \frac{4n}{5} \rceil$ for any tree of order $n \geq 1$. □

It seems like the bound in the above theorem could possibly be improved using a more detailed analysis. We conjecture that

$$\gamma_{b_2}(T) \leq \frac{2n+4}{3}$$

for any tree of order $n \geq 1$.

Corollary 5.3.2. *For any graph G of order $n > 1$, $\gamma_{b_2}(G) \leq \lceil \frac{4n}{5} \rceil$.*

Proof: It is easy to see that a dominating 2-broadcast in a spanning tree of G is a dominating 2-broadcast in G. □

It would also be interesting to find a lower bound on the size of a 2-multipacking in a tree of order n.

Chapter 6

Conclusion and Open Problems

In Chapter 3, we applied Farber's algorithm and extended it to broadcast domination and multipacking in strongly chordal graphs. In Chapter 4 we gave two linear algorithms for finding maximum multipacking and minimum dominating broadcast for trees, respectively. In Chapter 5, we investigated k-broadcast domination and k-multipacking. Many problems remain unsolved.

1. We know that broadcast domination is in P. What is the complexity of finding a maximum multipacking in a general graph?

2. Classify the graphs that have $\gamma_b(G) = mp(G)$.

   In this paper, we proved that $\gamma_b(G) = mp(G)$ for strongly chordal graphs. We also know that $\gamma_b(C_{3k}) = mp(C_{3k})$. Are there other classes of graphs that have $\gamma_b(G) = mp(G)$?

3. Can the bound $\gamma_b/mp < 3$ be improved?

   By the Duality Theorem of Linear Programming, we know $\gamma_b/mp \geq 1$.

4. Give an efficient algorithm to find a minimum dominating k-broadcast or maximum k-multipacking for a certain class of graphs.

   It seems that even to find the 2-broadcast domination for trees is a hard problem. Since we cannot use same vertex to broadcast twice, it is not so straight forward to get a 2-broadcast from a minimum dominating broadcast. Also, we cannot use shadow trees $\gamma_{b_2}(T) = \gamma_{b_2}(S_T)$ does not hold.

5. Is it true that, for any tree T of order $n > 1$, $\gamma_{b_2}(T) \leq \frac{2n+4}{3}$?

Theorem 5.3.1 gives a bound of the 2-broadcast domination number but the bound seems to be loose. We conjecture that $\frac{2n+4}{3}$ is the upper bound for the 2-broadcast domination number for a tree of order n.

6. Give a general lower bound on the 2-multipacking number of trees, in terms of the order.

Bibliography

[1] R.P. Anstee and M. Farber, *Characterizations of Totally Balanced Matrices*, Journal of Algorithms, **5** (1984), 215–230.

[2] K.S. Booth and J.H. Johnson, *Dominating sets in chordal graphs*, SIAM Journal on Computing, **11** (1982), 191–199.

[3] R. Brewster, C.M. Mynhardt and L.Teshima, *New bounds for the broadcast domination number of a graph*, Central European Journal of Mathematics, **11** (2013), 1334–1343.

[4] G. Chartrand and L. Lesniak, *Graphs and Digraphs*, Third edition, Prindle, Weber & Schmidt, Boston, MA, 1996.

[5] J. Dabney, *A Linear-Time Algorithm for Broadcast Domination in a Tree*, Master's Thesis, Department of Computer Science, Clemson University, 2007.

[6] J. Dabney, B.C. Dean and S.T. Hedetniemi, *A Linear-Time Algorithm for Broadcast Domination in a Tree*, Networks, **53** (2009), 160–169.

[7] J.E. Dunbar, D.J. Erwin, T.W. Haynes, S.M. Hedetniemi and S.T. Hedetniemi, *Broadcasts in graphs*, Discrete Applied Maths, **154** (2006), 59–75.

[8] D.J. Erwin, *Cost domination in graphs*, PhD Thesis, Department of Mathematics, Western Michigan University, 2001.

[9] D.J. Erwin, *Dominating broadcasts in graphs*, Bulletin of the ICA, **42** (2004), 89–105.

[10] M. Farber, *Characterizations of Strongly Chordal Graphs*, Discrete Mathematics, **43** (1983), 173–189.

[11] M. Farber, *Domination, Independent Domination, and Duality in Strongly Chordal Graphs*, Discrete Applied Mathematics, **7** (1984), 115–130.

[12] D.R. Fulkerson, A.J. Hoffman and R. Oppenheim, *On balanced matrices*, Math. Programming Study, **1** (1974), 120–132.

[13] P. Heggernes and D. Lokshtanov, *Optimal broadcast domination in polynomial time*, Discrete Mathematics, **306** (2006), 3267–3280.

[14] B.L. Hartnell and C.M. Mynhardt, *On the difference between broadcast and multipacking number of graphs*, Utilitas Mathematica, **94** (2014), 19–29.

[15] S. Herke, *Dominating broadcasts in graphs*, Master's Thesis, Department of Mathematics and Statisitcs, University of Victoria, 2007.

[16] S. Herke and C.M. Mynhardt, *Radial trees*, Discrete Mathematics, **309** (2009), 5950–5962.

[17] A.J. Hoffman and A.W.J. Kolen and M.Sakarovitch, *Totally-balanced and greedy matrices*, SIAM Journal on Algebraic and Discrete Methods, **6** (1985), 721–730.

[18] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co Ltd, New York, NY, 1979.

[19] A. Lubiw, *Doubly Lexical Orderings of Matrices*, SIAM Journal on Computing, **16** (1987), 854–879.

[20] A. Lubiw, Γ-*free Matrices*, Master's Thesis, Department of Combinatorics and Optimization, University of Waterloo, 1982.

[21] S. Lunney, *Trees with equal broadcast and domination numbers*, Master's Thesis, Department of Mathematics and Statistics, University of Victoria, 2011.

[22] S. Lunney and C.M. Mynhardt, *More trees with equal broadcast and domination numbers*, Australian Journal of Combinatorics, **61** (2015) 251–272.

[23] C.M. Mynhardt and J.Wodlinger, *Uniquely radial trees*, submitted, (2010).

[24] D.J. Rose, *Triangulated graphs and the elimination process*, Journal of Mathematical Analysis and Applications, **32** (1970) 597–609.

[25] J.P. Spinard, *Doubly Lexical Ordering of Dense* $0-1$ *Matrices*, Information Processing Letters, **45** (1993) 229–235.

[26] L. Teshima, *Broadcasts and multipackings in graphs*, Master's Thesis, Department of Mathematics and Statistics, University of Victoria, 2012.