# Broadcast Domination Algorithms for Interval Graphs, Series-Parallel Graphs, and Trees

Jean R. S. Blair* Pinar Heggernes† Steve Horton* Fredrik Manne†

**Abstract**

Broadcast domination assigns an integer value $f(u) \geq 0$ to each vertex $u$ of a given graph, such that every vertex $u$ with $f(u) = 0$ is within distance $f(v)$ from a vertex $v$ with $f(v) > 0$. We can regard the vertices $v$ with $f(v) > 0$ as broadcast stations, each having a transmission power that might be different from the powers of other stations. The optimal broadcast domination problem seeks to minimize the sum of the costs of the broadcasts assigned to the vertices of the graph. We present dynamic programming algorithms that solve the optimal broadcast domination problem for the first classes of graphs with non-trivial solutions: interval graphs, series-parallel graphs, and trees. We also show that optimal broadcast domination is equivalent to optimal domination on proper interval graphs.

## 1 Introduction and motivation

Domination in graphs is a well known and thoroughly studied subject [15, 16]. A *dominating set $S$* is a subset of the vertices in a graph such that every vertex in the graph either belongs to $S$ or has a neighbor in $S$. The standard optimal domination problem seeks a dominating set of minimum cardinality. Numerous variants of this problem have been studied [2, 23]. For example, optimal $r$-domination seeks a dominating set of minimum cardinality such that every vertex of the graph is within distance $r$ of some vertex of the dominating set [17, 26]. The $(k, r)$-center problem seeks an $r$-dominating set that contains at most $k$ vertices [1, 12].

An application of the standard domination problem was given by Liu in 1968 [22]: a dominating set represents a set of cities having broadcast stations, where every city can hear a broadcast station placed in it or in a neighboring city. Even though this model is limited by the fact that all broadcast stations are assumed to have the same transmission power, it has spawned several variants. For example, $r$-domination is a natural generalization of this model that assumes an ability to hear broadcasts that originate from a distance of at most $r$.

Recently, Erwin introduced the concept of *broadcast domination* [9] in which the broadcast stations (i.e., vertices in the dominating set) are permitted to have different transmission powers. This is a more realistic model of broadcast reachability than the standard domination problem, since transmitters are not, in general, identical. For example, FM

*United States Military Academy, West Point, New York 10996 USA, Jean.Blair@usma.edu, Steve.Horton@usma.edu

†University of Bergen, N-5020 Bergen, Norway, Pinar.Heggernes@ii.uib.no, Fredrik.Manne@ii.uib.no

radio stations are distinguished both by their transmission frequency and by their ERP (Effective Radiated Power). A transmitter with a higher ERP can transmit further, but it is more expensive to build and to operate. Based on this, the broadcast domination problem seeks to compute an integer valued broadcast function $f$ on the vertices, such that every vertex of the graph is distance at most $f(v)$ from some vertex $v$ that has $f(v) > 0$. A broadcast domination is optimal if it minimizes the sum of the costs of the broadcasts across all vertices in the graph. These costs are typically taken to be the $f(v)$ values. Other related broadcast problems are discussed in [8].

In this paper, we present polynomial time algorithms that solve the optimal broadcast domination problem on interval graphs, series-parallel graphs, and trees. The standard optimal domination problem is NP-hard on, for example, planar graphs [12], bipartite graphs [7], and chordal graphs [3], but can be solved in polynomial time on, for example, AT-free graphs [21], permutation graphs [11], and interval graphs [10]. Some variants of the problem, like the ones previously mentioned, have straightforward reductions from the standard domination problem, showing that they are NP-hard on arbitrary graphs. However, the computational complexity of optimal broadcast domination on general graphs is an open problem [8]. Easy polynomial time solutions have been found for paths, cycles, complete graphs, and grid graphs. In each of these cases, there are optimal solutions that either are also solutions to the standard domination problem, or have exactly one non-zero broadcast located at the center of the graph [8]. The classes of graphs we address in this paper do not exhibit this property, thereby lending interest to our algorithms. In addition, we observe in Section 6 that the existence of the series-parallel algorithm is not anticipated by current theory regarding algorithms on recursively constructed graph classes.

This paper is organized as follows. Necessary graph terminology and background, including a formal definition of broadcast domination, is given in the next section. Sections 3, 4, and 5, respectively, describe algorithms for interval graphs, series-parallel graphs, and trees.

## 2 Broadcast domination

All input graphs in this paper are simple and connected. Let $G = (V, E)$ be a graph with $n = |V|$ and $m = |E|$. For any vertex $v \in V$, the *neighborhood* of $v$ is the set $N(v) = \{u \mid uv \in E\}$ and the *closed neighborhood* is the set $N[v] = N(v) \cup \{v\}$. Similarly, for any set $S \subseteq V$, $N(S) = \cup_{v \in S} N(v) - S$ and $N[S] = N(S) \cup S$. A set $S$ is a *dominating set* if $N[S] = V$. The minimum cardinality of a dominating set of $G$ is denoted by $\gamma(G)$.

The *distance*, $d(u, v)$, between two vertices $u$ and $v$ in $G$ is the smallest number of edges on a path between $u$ and $v$ in $G$. The *eccentricity*, $e(v)$, of a vertex $v$ is the largest distance from $v$ to any vertex of $G$. The *radius*, $rad(G)$, of $G$ is the smallest eccentricity in $G$. The *diameter*, $diam(G)$, of $G$ is the largest eccentricity in $G$.

A function $f : V \to \{0, 1, \cdots, diam(G)\}$ is a *broadcast* if for every vertex $v \in V$, $f(v) \leq e(v)$. The set of *broadcast dominators* defined by $f$ is the set $V_f = \{u \mid f(u) > 0\}$. The set of vertices that a vertex $v$ can *hear* is $H_f(v) = \{u \in V_f \mid d(u, v) \leq f(u)\}$. We will omit the subscript $f$ when the broadcast function is clear from the context. The *cost* of a broadcast $f$ incurred by a set $S \subseteq V$ is $f(S) = \sum_{v \in S} f(v)$. Thus, $f(V)$ is the total cost incurred by broadcast function $f$.[1] We say that $G$ has an $f(V)$-*broadcast*.

A broadcast is *dominating* if $|H(v)| \geq 1$ for every vertex $v$ of $G$. The term $\gamma_b(G)$ denotes the minimum cost of a dominating broadcast on $G$. We will refer to a dominating broadcast of cost $\gamma_b(G)$, as an *optimal broadcast*. Although a broadcast function is allowed to assign values from $\{0, 1, ..., diam(G)\}$, we never need to assign values larger than $rad(G)$ to any vertex in order to achieve an optimal broadcast. Choosing a vertex $v$ of minimum eccentricity and assigning $f(v) = rad(G)$ while assigning $f(w) = 0$ to all other vertices $w$, defines a dominating broadcast on $G$. We will call such a broadcast a *radial broadcast*. For some graph classes, a radial broadcast is also an optimal broadcast, and such graphs are called *radial* [8]. However, for interval graphs, series-parallel graphs, and trees, radial broadcasts are not necessarily optimal, as can be seen from the following simple example. Consider a path on 6 vertices: $v_1, v_2, ..., v_6$. A radial broadcast $f$ requires $f(v_3)$ or $f(v_4)$ to be 3, which is the radius of a path on 6 vertices. However a dominating broadcast $f'$ of cost 2 can be achieved by assigning $f'(v_2) = f'(v_5) = 1$.

A broadcast is *efficient* if every vertex hears exactly one broadcast, that is, for every $v$, $|H(v)| = 1$. The following result from [8] is central to several of our results.

**Theorem 2.1** (Dunbar et. al. [8]) *Every graph $G$ has a $\gamma_b(G)$-broadcast that is efficient.*

Before we continue, we also observe that the following more restricted problem is NP-complete, which follows immediately by restriction from dominating set. RESTRICTED BROADCAST DOMINATION (RBD): Given a graph $G = (V, E)$, and positive integers $K, M \leq |V|$, is there a broadcast domination $f$ of $G$ such that $\sum_{v \in V} f(v) \leq K$ and $\max_{v \in V} f(v) \leq M$?

# 3 Interval graphs

In this section we give a dynamic programming algorithm for computing an optimal efficient broadcast on an interval graph. Moreover, we show that for proper interval graphs our problem reduces to the dominating set problem. First some notation.

A graph $G = (V, E)$ is an *interval graph* if sets of consecutive integers (intervals) can be assigned to every vertex, such that the sets $I(u)$ and $I(v)$ corresponding to vertices $u$ and $v$ intersect if and only if $uv \in E$. Interval graphs can be recognized and the corresponding intervals of the vertices of the graph can be constructed in linear time [4]. It is implicit from [13] and [18] that the integers constituting the intervals can be chosen from the set $\{1, 2, ..., n\}$. The interval graph property is hereditary [14], thus induced subgraphs of interval graphs are also interval. An interval graph is called *proper interval* if no interval is completely contained within another interval. Roberts [25] has shown that an interval graph is proper if and only if it contains to induced copy of $K_{1,3}$.

By the above mentioned results, the vertices of an interval graph can be sorted in non-decreasing order of the left endpoints of their corresponding intervals in linear time. Let $V = \{v_1, v_2, ..., v_n\}$ be the sorted order of the vertices of a given interval graph $G = (V, E)$. In the rest of this section, we will always assume that the vertices of a given interval graph are sorted in this manner. In addition, we will not distinguish between vertices and

[1] The cost function might also be defined differently, reflecting the possibility that the cost of providing a broadcast $f(v)$ is not equal to $f(v)$. Our algorithms can easily be adapted to alternate cost functions.

their corresponding intervals; thus we will use vertex $v_i$ and interval $v_i$ interchangeably. We define $G_{ij}$ to be the subgraph of $G$ induced by vertices/intervals $v_i, v_{i+1}, ..., v_j$, with $1 \leq i \leq j \leq n$. We also denote the left and right endpoints of an interval $v_i$ by $l(v_i)$ and $r(v_i)$ respectively.

A broadcast with $f(v_i) > 0$ is, by definition, heard by exactly those intervals $v_j$ where $d(v_i, v_j) \leq f(v_i)$. Therefore, we commence by establishing a method for constructing shortest paths in interval graphs. Assuming that $i < j$, the shortest path $P = \{v_i = p_0, p_1, ..., p_k = v_j\}$ that we will be using is defined such that $p_{t+1}$ is chosen to be the vertex of $N(p_t)$ with largest right endpoint, for each $t$ between 0 and $k-2$. A formal algorithm for this process, which we call **SP**, is given below:

**Algorithm** Shortest Path (**SP**)
**Input:** $G$, $v_i$, $v_j$.
**Output:** A path $P$ between $v_i$ and $v_j$ containing a minimum number of edges.
$k = 0$; $p_k = v_i$; $P = \{p_k\}$;
**while** $p_k \notin N(v_j)$ **do**
    Choose $p_{k+1}$ to be an interval in $N(p_k)$ with largest right endpoint;
    $P = P \cup \{p_{k+1}\}$; $k = k + 1$;
$P = P \cup \{v_j\}$;    /* *Note that* $v_j = p_k$ */

**Lemma 3.1** *$P$ is a shortest path between $v_i$ and $v_j$.*

**Proof.** Let $S = \{v_i = s_0, s_1, \ldots, s_t = v_j\}$ be a shortest path from $v_i$ to $v_j$ and assume on the contrary that $|S| < |P|$. Let $p_k$ be the first interval such that $p_k \neq s_k$, where $0 < k < t$. We will show that $s_k$ can be replaced by $p_k$ in the shortest path and the result will follow by induction. First, note that the interval $s_{k+1}$ cannot intersect with $s_{k-1}$, otherwise $s_k$ is redundant in $S$. If $r(v_{k+1}) < l(v_{k-1})$, meaning that $s_{k+1}$ is completely to the left of $s_{k-1}$, then some $s_l$ with $l > k+1$, must intersect with $s_{k-1}$ in order to reach $v_j$, and both $s_k$ and $s_{k+1}$ are redundant. Thus $s_{k+1}$ must lie completely to the right of $s_{k-1}$. Since $p_k$ is adjacent to $s_{k-1}$ and stretches at least as far right as $s_k$, $p_k$ will also intersect with $s_{k+1}$ and the result follows. ∎

**Corollary 3.2** *Let $f$ be a dominating broadcast function on $G$ with $v_i \in V_f$. If $v_j$ hears $v_i$, where $i < j$, then every vertex $v_k$ hears $v_i$, for $i < k < j$.*

**Corollary 3.3** *If $d(v_i, v_k) = d(v_i, v_j) = t$ for some $i, k, j$ satisfying $i < k < j$, then $d(v_i, v_q) = t$ for every $q$ satisfying $k < q < j$.*

**Proof.** By Lemma 3.1 every interval $v_q$ with $r(p_{t-2}) < l(v_q) \leq r(p_{t-1})$ is of distance $t$ from $v_i$. Since the intervals are ordered by their left endpoints this includes $v_k$, $v_l$, and any vertex $v_q$ with $k \leq q \leq l$. ∎

A shortest path $P$ from $v_i$ to $v_j$, where $j < i$, can be found in a similar manner as in algorithm **SP**; the only difference is that one selects an interval with the smallest left endpoint in each step. Due to the fact that we have ordered the intervals by their left endpoints, we do not get a similar result as Corollary 3.2 when $j < i$. This is reflected in the following observation:

**Observation 3.4** *Let $d(v_i, v_j) = t$ with $j < i$, and let $P$ be a shortest path between $v_i$ and $v_j$ found as described above. Then for any $k$ with $r(v_k) < l(p_{t-1})$, we have $d(v_i, v_k) > t$.*

**Proof.** The result follows since a shortest path from $v_i$ to $v_k$ requires more than $t$ intervals. ■

It follows from Observation 3.4 that even if $v_j$ hears $v_i$, there might exist a $v_k$ with $j < k < i$, such that $v_k$ does not hear $v_i$. For this situation to happen, we must have $l(v_j) \leq l(v_k)$ and $r(v_k) < l(p_{t-1})$, where $t = d(v_i, v_j)$. Both proper interval graphs and efficient broadcasts in general avoid this situation, as stated in the following corollary.

**Corollary 3.5** *Let $f$ be a dominating broadcast function on $G$ with $v_i \in V_f$. If $G$ is proper interval or $f$ is efficient, then the following is true: If $v_j$ hears $v_i$ with $j < i$, then every vertex $v_k$ hears $v_i$, for $j < k < i$.*

**Proof.** Assume that $G$ is proper interval or that $f$ is efficient, and let there exist a $k$ with $j < k < i$, such that $v_k$ does not hear $v_i$, although $v_j$ hears $v_i$. As discussed above, $v_k$ must satisfy $r(v_k) < l(p_{t-1})$ and $l(v_j) \leq l(v_k)$, and $v_k$ is therefore completely contained within $v_j$. This contradicts the fact that $G$ is proper interval. Let $v_q \neq v_i$ be any vertex that $v_k$ hears. Then $v_j$ also hears $v_q$ contradicting the fact that $f$ is efficient, since $v_j$ then hears both $v_q$ and $v_i$. ■

For proper interval graphs we get a result analogous to Corollary 3.3.

**Corollary 3.6** *If $G$ is proper interval and $d(v_i, v_k) = d(v_i, v_j) = t$ for some $k, j, i$ where $k < j < i$, then $d(v_i, v_q) = t$ for each value $q$ with $k < q < j$.*

**Proof.** By Corollary 3.5 it follows that any $v_q$ with $k < q < i$ has $d(v_i, v_q) \leq t$. If $d(v_i, v_q) < t$ then we must have $r(v_j) < r(v_q)$ and $v_j$ is contained in $v_q$ contradicting the fact that $G$ is proper interval. ■

Now we present a result that resolves one of the mentioned open problems in [8].

**Theorem 3.7** *For any proper interval graph $G$, $\gamma_b(G) = \gamma(G)$.*

**Proof.** Let $G = (V, E)$ be a proper interval graph, and let $f$ be an optimal efficient broadcast on $G$, where $f(v) > 1$ for some vertex $v$. We will show that there exists a dominating broadcast $f'$ on $G$ where $f'(v) \leq 1$ for every vertex $v$, and where $f'(V) = f(V)$.

Let $v_i$ be a vertex of $G$ with $f(v_i) = t > 1$. We denote the set of vertices at exact distance $d$ from $v_i$ with indices larger than $i$ by $X_r^d$ and those with indices smaller than $i$ by $X_l^d$. From Corollaries 3.3 and 3.6 it follows that the vertices that hear $v_i$ can be partitioned into consecutive subsets: $X_l^t, X_l^{t-1}, \ldots, X_l^1, \{v\}, X_r^1, X_r^2, \ldots, X_r^t$. We will show that there exists a vertex $v_k$ in $X_r^{t-1}$ that is adjacent to every vertex in $X_r^{t-1} \cup X_r^t$. Similarly it can also be shown that there exists a vertex $v_j$ in $X_l^{t-1}$ that is adjacent to every vertex in $X_l^{t-1} \cup X_l^t$. Thus we can set $f'(v_k) = f'(v_j) = 1$ and $f'(v_i) = f(v_i) - 2$, and every vertex that hears $f(v_i)$ will still hear at least one other vertex in $f'$ without increasing the total cost of the broadcast. Now, since $f$ is efficient, no vertex $v_l \neq v_i$, $j < l < k$ has $f(v_l) > 0$. Thus, we can continue in the same way to decrease the power of each vertex $v_i$ with $f(v_i) > 1$ until all vertices have power $\leq 1$.

It remains to prove the existence of $v_k$ (and also $v_j$). Let $v_k$ be the vertex with largest index in $X_r^{t-1}$. Observe first that no vertex $v$ in $X_r^{t-1}$ can have $r(v) > r(v_k)$, because otherwise $v$ would contain $v_k$ as a subinterval, contradicting the fact that $G$ is proper interval. Since every vertex $x$ in $X_r^t$ must be adjacent to some vertex in $X_r^{t-1}$, it follows that $x$ must also be adjacent to $v_k$. To see that $v_k$ must be adjacent to every vertex in $X_r^{t-1}$ note first that $v_k$ must be adjacent to some vertex $v_l$ in $X_r^{t-2}$. It follows that any vertex in $X_r^{t-1}$ not adjacent to $v_k$ will be completely contained in $v_l$. The proof of existence for $v_j$ is similar; here we pick $v_j$ to be the vertex of $X_l^{t-1}$ that has the smallest index. ∎

The following example shows that the above theorem does not apply to interval graphs in general. Let $G = (V, E)$ be the interval graph defined by $V = \{a, b, c, d, e, f\}$ and $E = \{ab, bc, ce, ef, cd\}$. With $f(c) = 2$ an optimal broadcast of total cost 2 is achieved. However, this cannot be achieved by assigning $f(v) = f(w) = 1$ for any pair of vertices $v, w \in V$.

### 3.1 An algorithm for interval graphs

We now present a dynamic programming algorithm for computing an optimal efficient broadcast on an interval graph $G$ in $O(n^3)$ time. This time complexity is achieved with help of an $O(n^2)$ time preprocessing that computes radial broadcasts on all subgraphs $G_{ij}$. The discussion and analysis of this preprocessing is postponed until Section 3.2. Although here we only compute the cost of an optimal solution, extending the results to compute the solution itself is straight forward, and does not add to the time complexity.

It follows from Corollaries 3.2 and 3.5 that an optimal efficient broadcast consists of a set of broadcast dominators that each dominates exactly one subgraph $G_{ij}$ and nothing outside of $G_{ij}$. Since the solution is efficient, there is no overlap between these graphs. Noting that in a connected graph we will never have a broadcast dominator that dominates only itself, we get the following recursion for finding a minimum cost *non-radial* efficient solution for $G_{ij}$:
$\gamma = \min_{i<k<j} Opt(i,k) + Opt(k+1,j)$. The minimum cost of dominating $G_{ij}$ and nothing else is then given by $Opt(i,j) = \min\{MinRad(i,j), \gamma\}$.

Note that it might happen that there does not exist an efficient optimal solution for one or both of $G_{ik}$ and $G_{k+1,j}$ that cannot be heard from outside of these subgraphs. If this is the case we set $Opt(i,j) = \infty$. If we consider $G_{ik}$ then an efficient optimal solution exists if and only if $G_{ik}$ can be decomposed into one or more non overlapping graphs, each dominated by exactly one broadcast dominator that cannot be heard from outside of this subgraph.

Initially we determine for each $i$ and $j$, $i \leq j$, whether there exists a radial broadcast for $G_{ij}$ that cannot be heard from outside of $G_{ij}$. If a radial solution exists, then its cost is stored in $MinRad(i,j)$. If there is no such solution then $MinRad(i,j) = \infty$.

The following $O(n^3)$ time dynamic programming algorithm, which we call Interval Broadcast Domination (**IBD**), progresses by building efficient optimal solutions to all $G_{ij}$ containing $l$ intervals before moving to graphs containing $l+1$ intervals. The correctness and the $O(n^3)$ time complexity of **IBD** follows from the above discussion.

**Algorithm** Interval Broadcast Domination (**IBD**)
**Input:** An interval graph $G$;
**Output:** $\gamma_b(G) = Opt(1, n)$.
Compute all radial solutions and place them in a table $MinRad$;
**for** $i = 1$ **to** $n - 1$ *do*
    $Opt(i, i + 1) = MinRad(i, i + 1)$;
**for** $i = 2$ **to** $n - 1$ **do**
    **for** $j = 1$ **to** $n - i$ **do**
        $\gamma = \min_{k=j+1}^{j+i-2}\{Opt(j, k) + Opt(k + 1, j + i)\}$;
        $Opt(j, j + i) = \min\{MinRad(j, j + i), \gamma\}$;

### 3.2 Computing radial solutions

We now describe how $MinRad(i, j)$, a radial broadcast needed to dominate exactly $G_{ij}$, can be computed efficiently for all $i, j$. If no such solution exists, i.e., if all radial solutions can be heard from outside of $G_{ij}$, then $MinRad(i, j) = \infty$.

We first consider what part of the graph would be dominated if we were to set $f(v_k) = t$, $t > 0$. Let $v_i$ and $v_j$ be the respectively lowest and highest numbered vertices that are dominated by $f(v_k)$. It follows from Corollaries 3.2 and 3.5 that every vertex in $G_{ij}$ must hear $f(v_k)$ if we are going to consider $v_k$ as a candidate for a radial broadcast dominator. We will look at $G_{ik}$ and $G_{kj}$ separately starting with $G_{kj}$.

Since any sub-path of a shortest path is also a shortest path, it follows that if $v_k = p_0, p_1, \ldots, p_t = v_j$, with $t > 1$, is a shortest path from $v_k$ to $v_j$ given by Algorithm **SP**, where $k < j$, then there is a shortest path of length $t - 1$ from $p_1$ to $v_j$. Similarly, if there is a shortest path of length $t - 1$ from $p_1$ to $v_j$ then there is a shortest path from $v_k$ to $v_j$ of length $t - 1$ if $v_k p_1 \in E$; otherwise the shortest length is $t$. In terms of broadcast domination this means that $f(v_k) = t$ will dominate exactly the same set of vertices in $G_{p_1,n}$ as $f(p_1) = t - 1$. Assuming that $k < p_1$, by Corollary 3.2, every vertex in $G_{k+1,p_1-1}$ will also hear $f(v_k) = t$. Thus if $v_j$ is the highest numbered vertex that can hear $f(p_1) = t - 1$ then $f(v_k) = t$ will dominate $G_{k+1,j}$ and no vertex in $G_{j+1,n}$. Setting $f(v_k) = 1$ will dominate $G_{k+1,j}$ where $v_j$ is the highest numbered vertex with $l(v_j) \leq r(v_k)$. This value can be found by searching through the neighbors of $v_k$.

The complete algorithm, called Maximal Right Domination(**MRD**), is given below. This algorithm returns a table $R(k, f(v_k))$ containing the maximum value $q \in H(v_k)$ for each $v_k \in V$ and $1 \leq f(v_k) \leq e_r(v_k)$ where $e_r(v_k)$ denotes the *right eccentricity* of $v_k$, i.e., the largest distance from $v_k$ to any vertex that has a larger index than $k$. In order to perform these computations efficiently the vertices should be processed in order of decreasing right endpoints. For each vertex $v_k$ we search for the the highest numbered adjacent vertex in order to determine the reach of $f(v_k) = 1$. We then search for its neighbor $v_q$ with the largest right endpoint. From this we copy the values for how far $f(v_q) = t$ will reach in $G_{q,n}$, $1 \leq t \leq e(v_q)$.

**Algorithm** Maximal Right Domination(**MRD**)
**Input:** An interval graph $G$;
**Output:** A table $R(k, f(v_k)) = \max\{q \mid v_k \in H(v_q)\}$ with $1 \leq f(v_k) \leq e_r(v_k)$ for each $v_k \in V$.

**for** $k = n$ **downto** $1$ **do**
    $R(k, 1) = \max\{q \mid l(v_q) \leq r(v_k)\}$;
    Choose $p_1$ to be an interval in $N(v_k)$ with largest right endpoint;
    **for** $i = 1, e_r(p_1)$ **do**
        $R(k, i+1) = R(p_1, i)$;

Determining the value of $R(k, 1)$ for each $v_k$ has an accumulated cost of $O(m) = O(n^2)$. Since at most $n - 1$ values are set for each interval the overall time for **MRD** is $O(n^2)$.

The computation of which lower numbered vertices will be dominated by $f(v_k) = t$ is similar. The main difference is that while **MRD** processes the vertices by decreasing right endpoints, we now process the intervals by increasing left endpoints, and that every $f$-value on a vertex might not yield a broadcast that can be used in an efficient solution. Assume $f(v_k) = 1$ and that $v_q$ is the lowest numbered vertex adjacent to $v_k$, $q < k$. Then from Corollary 3.5 if follows that if there is any vertex in $G_{q+1,k-1}$ not adjacent to $v_k$ then $f(v_k) = 1$ cannot be used in building an efficient solution. To obtain values for $f(v_k) > 1$, it is sufficient to copy these from the same $v_q$. This follows since $v_k$ will dominate the same vertices using strength $t$ as $v_q$ does with strength $t - 1$. And if $f(v_q) = t - 1$ cannot be used in an efficient solution then neither can $f(v_k) = t$. The total computation can be carried out in time $O(n^2)$.

Combining the results for which vertices $f(v_k) = t$ will dominate, we can decide whether this can be used in a radial solution of some $G_{ij}$. If this is the case and $t$ is lower than the previous lowest $f()$ value used for this graph, we set $MinRad(i, j) = t$. For all values of $i$ and $j$ where there does not exist a radial solution that cannot be heard from outside of $G_{ij}$, we set $MinRad(i, j) = \infty$. It follows that the whole process can be carried out in time $O(n^2)$.

# 4 Series-parallel graphs

In this section we describe a dynamic programming algorithm for computing a minimum cost broadcast function that dominates a series-parallel graph. Throughout this section we assume the given graph is series-parallel, and use $r$ to represent the radius of the graph.

Informally, a *recursive graph class* is one in which any sufficiently large member in the class can be formed by successively joining smaller members in the class at specific vertices called *terminals*.[2] Series-parallel graphs are partial 2-trees, a recursive graph class with 2 terminals. We use $t_L(G)$ and $t_R(G)$ to denote the left terminal and the right terminal of $G$, respectively. Every member of the class can then be decomposed into base graphs, typically taken to be a single edge, i.e., the $K_2$. This decomposition is specified by a decomposition tree that identifies two children for each non-leaf node of the tree, and an associated operation that combines the children to create the parent. Series-parallel graphs can be recognized and a corresponding decomposition tree can be constructed in linear time [27].

[2] A more formal description of recursively constructed graphs can be found in [20]. Examples of how this recursive structure can be used to solve graph problems appear in a host of references among which are [24] and [19]. More formal *models* of the methodology appear in [28] and [6].

Let $G = (V, \{t_L(G), t_R(G)\}, E)$ and let $G_j = (V_j, \{t_L(G_j), t_R(G_j)\}, E_j)$ for $j = 1, 2$ be 2-terminal graphs. Define the series operation as $s(G_1, G_2) = G$ if $t_L(G_1) = t_L(G)$, $t_R(G_1) = t_L(G_2)$, and $t_R(G_2) = t_R(G)$. This operation associates $t_R(G_1)$ with $t_L(G_2)$ and then the resulting vertex loses its status as a terminal. The left jackknife operation, $left(G_1, G_2) = G$, is the same as series, except that the associated vertex retains its terminal status, becoming $t_R(G)$, and $t_R(G_2)$ loses its terminal status. The right jackknife operation $right(G_1, G_2) = G$ is analogous. Finally, define the parallel operation as $p(G_1, G_2) = G$ if $t_L(G_1) = t_L(G_2) = t_L(G)$ and $t_R(G_1) = t_R(G_2) = t_R(G)$. Note that in each case, the resulting graph $G$ has two terminals.

### 4.1 Broadcasts in series-parallel graphs

Given a broadcast function $f : V \to \{0, 1, 2, ..., r\}$, we say that dominance condition $(dom_L, dom_R)$ exists if the following holds.

- $f$ dominates $G_i$ except for vertices within distance $-dom_k$ of $t_k(G_i)$ for $k \in \{L, R\}$ whenever $dom_k < 0$ and
- $f$ would dominate hypothetical paths of length $dom_k$ joined to $t_k(G_i)$ whenever $dom_k \geq 0$.

If $dom_k < 0$ we say $t_k(G_i)$ is *underdominated* and if $dom_k \geq 0$ we say $t_k(G_i)$ is *overdominated.* The definition of a dominance condition allows portions of $G_i$ to be undominated at the current node in the decomposition tree only when there is a requirement indicated (the negative $dom_k$ value) that "demands" that this underdominance be corrected at some point closer to the root of the decomposition tree.

Now we can define $P_{G_i}(dom_L, dom_R)$ to be the lowest cost of a broadcast function on $G_i$ given that dominance condition $(dom_L, dom_R)$ exists in $G_i$. $P$ will be either $N$, $L$, $R$, or $B$ to represent *neither*, *left*, *right*, or *both* terminals having a non-zero broadcast originating there. For example, $L_{G_3}(1, -2)$ represents the lowest cost of a broadcast function in $G_3$ that provides an overdominance of 1 at the left terminal, has an underdominance of $-2$ at the right terminal, has a broadcast originating at $t_L(G_3)$, and has no broadcast originating at $t_R(G_3)$.

The following properties of optimal broadcast dominations will be useful in establishing the correctness of our series-parallel algorithm.

**Lemma 4.1** *Let $f$ be an optimal broadcast domination of a series-parallel graph $G$ and consider a graph $G_i = (V_i, \{t_L(G_i), t_R(G_i)\}, E_i)$ in the decomposition of $G$. Let $d_{G_i}(t_L, t_R)$ be the distance in $G_i$ between $t_L(G_i)$ and $t_R(G_i)$.*

- *If $f$ applied to $G_i$ is such that $dom_L = f(t_L(G_i)) > 0$ and $dom_R < 0 = f(t_R(G_i))$, then*
$$-(dom_R + d_{G_i}(t_L, t_R) + 1) < f(t_L(G_i)).$$
- *If $f$ applied to $G_i$ is such that $dom_R = f(t_R(G_i)) > 0$ and $dom_L < 0 = f(t_L(G_i))$, then*
$$-(dom_L + d_{G_i}(t_R, t_L) + 1) < f(t_R(G_i)).$$

**Proof.** We prove only the first bullet, since the proof of the second bullet is analogous. Suppose the first bullet is false for some $G_i$. Then $-(dom_R + d(t_L(G_i), t_R(G_i)) + 1) \geq f(t_L(G_i))$, and the underdominance at $t_R(G_i)$ demands dominance that will dominate every vertex in $G$ that is dominated by the broadcast originating at $t_L(G_i)$. But then $f(t_L(G_i))$ can be set to 0 while maintaining broadcast domination in $G$, contrary to the optimality of $f$. ∎

**Lemma 4.2** *In any optimal broadcast domination $f$ of a graph $G = (V, E)$, if there are two neighbors $v, w \in V$ such that $f(v) > 0$ and $f(w) > 0$, then $f(v) = f(w)$.*

**Proof.** Suppose otherwise. Without loss of generality, assume $f(v) > f(w) > 0$ where $f$ is an optimal dominating broadcast of $G$ with $(v, w) \in E(G)$. Then every vertex dominated by the broadcast originating at $w$ is also dominated by the broadcast originating at $v$. But then $f(w)$ can be set to 0 while maintaining broadcast domination in $G$, contrary to the optimality of $f$. ∎

Given a series-parallel graph $G$, the algorithm computes the values of $N_{G_i}$, $L_{G_i}$, $R_{G_i}$, and $B_{G_i}$ for each graph $G_i$ in a decomposition tree of $G$, in a bottom-up fashion. Since an optimal broadcast function will have $f(v) \leq r$ for all $v \in V$, we restrict the dominance conditions to ranges from $-r$ to $r$. The next subsection describes how to compute the $N$, $L$, $R$, and $B$ arrays for leaves. Subsections 4.3 and 4.4 present recursive relationships that allow the computation of $P_{G_i}(dom_L, dom_R)$ for $P \in \{N, L, R, B\}$ for every non-leaf graph $G_i$ in the decomposition tree. The equations described in the three subsections are summarized in Tables 1, 2, and 3.

## 4.2 Initialization

Recall that each leaf in the decomposition tree of a series-parallel graph corresponds to a $K_2$. The following discussion of how to initialize the four cost arrays $N_{K_2}$, $L_{K_2}$, $R_{K_2}$ and $B_{K_2}$ is summarized with the equations in Table 1.

For the case of $N_{K_2}$ (i.e., $f(t_L(K_2)) = f(t_R(K_2)) = 0$), no broadcast originates from within the $K_2$. Clearly, then, to lead to a valid broadcast domination of the series-parallel graph, the dominance condition of the $K_2$ must indicate a demand that will eventually dominate the $K_2$. The dominance conditions indicating such demand are identified in the following observation.

**Observation 4.3** *Let $f$ be a broadcast domination of a series-parallel graph $G$ and consider an edge $K_2 = (V_{K_2}, \{t_L(K_2), t_R(K_2)\}, E_{K_2})$ corresponding to a leaf in the decomposition of $G$. If $f$ applied to $K_2$ is such that $f(t_L(K_2)) = f(t_R(K_2)) = 0$, then $dom_L + dom_R \leq -2$.*

We use the value $\infty$ to represent invalid dominance conditions. All other cases are set to 0, representing zero cost when no broadcast originates from the $K_2$.

Recall that $L_{K_2}$ represents $f(t_L(K_2)) > 0$ and $f(t_R(K_2)) = 0$, in which case the cost due to the $K_2$ is simply $f(t_L(K_2))$. Thus, we must enter $f(t_L(K_2))$ for all dominance conditions where $dom_L = f(t_L(K_2))$ and $dom_R$ together with $dom_L$ lead to a valid broadcast domination of the final graph $G$.

**Observation 4.4** *Let $f$ be a broadcast domination of a series-parallel graph $G$ and consider an edge $K_2 = (V_{K_2}, \{t_L(K_2), t_R(K_2)\}, E_{K_2})$ corresponding to a leaf in the decomposition of $G$. If $f$ applied to $K_2$ is such that $f(t_L(K_2)) > 0$ and $f(t_R(K_2)) = 0$, then $dom_R < dom_L$.*

This observation gives us the first line in the equation for $L_{K_2}$; the second line is a result of Lemma 4.1. Initialization of $R_{K_2}$ is analogous.

Intuitively, we must set $B_{K_2}(dom_L, dom_R) = \infty$ whenever the dominance condition renders one of the values of $dom_L$ and $dom_R$ superfluous. The conditions when this occurs were formalized in Lemma 4.2.

Table 1: Initializations for Algorithm **SPBD**.

| | | |
|---|---|---|
| $N_{K_2}(dom_L, dom_R)$ | $=$ | $\begin{cases} 0 & \text{if } dom_L + dom_R \leq -2 \\ \infty & \text{if } dom_L + dom_R > -2 \end{cases}$ |
| $L_{K_2}(dom_L, dom_R)$ | $=$ | $\begin{cases} \infty & \text{if } dom_R \geq dom_L \\ \infty & \text{if } dom_R \leq -(dom_L + 2) \\ dom_L & \text{otherwise} \end{cases}$ |
| $R_{K_2}(dom_L, dom_R)$ | $=$ | $\begin{cases} \infty & \text{if } dom_L \geq dom_R \\ \infty & \text{if } dom_L \leq -(dom_R + 2) \\ dom_R & \text{otherwise} \end{cases}$ |
| $B_{K_2}(dom_L, dom_R)$ | $=$ | $\begin{cases} dom_L + dom_R & \text{if } (dom_L = dom_R > 0) \\ \infty & \text{otherwise} \end{cases}$ |

### 4.3 The series operation

Consider a node $G$ in the decomposition tree that is formed by $G = s(G_1, G_2)$, and consider the computation of $L_G$. Note that allowable configurations with an originating broadcast at $t_L(G)$ and no originating broadcast at $t_R(G)$ must either come from $B_{G_1}$ and $L_{G_2}$ or from $L_{G_1}$ and $N_{G_2}$. Other pairs either fail to have the specified originating broadcast, or are incompatible in the sense that $t_R(G_1)$ cannot be associated with $t_L(G_2)$ because one has an originating broadcast and the other does not.

For the case of $B_{G_1}$ and $L_{G_2}$, $t_R(G_1)$ and $t_L(G_2)$ have an identical originating broadcast $i$. The formula simply compares these options for all of the relevant $i$ values, and uses a cheapest one as a candidate value for $L_G(dom_L, dom_R)$. Note that the cost of the

Table 2: Equations for the $G = s(G_1, G_2)$ Operation.

| | | |
|---|---|---|
| $N_G(dom_L, dom_R)$ | $=$ | $\min \begin{cases} \min\limits_{1 \le i \le r} R_{G_1}(dom_L, i) + L_{G_2}(i, dom_R) - i \\ \min\limits_{0 \le i \le j \le r-1} N_{G_1}(dom_L, j) + N_{G_2}(-i-1, dom_R) \\ \min\limits_{0 \le i \le j \le r-1} N_{G_1}(dom_L, -i-1) + N_{G_2}(j, dom_R) \end{cases}$ |
| $L_G(dom_L, dom_R)$ | $=$ | $\min \begin{cases} \min\limits_{1 \le i \le r} B_{G_1}(dom_L, i) + L_{G_2}(i, dom_R) - i \\ \min\limits_{0 \le i \le j \le r-1} L_{G_1}(dom_L, j) + N_{G_2}(-i-1, dom_R) \\ \min\limits_{0 \le i \le j \le r-1} L_{G_1}(dom_L, -i-1) + N_{G_2}(j, dom_R) \end{cases}$ |
| $R_G(dom_L, dom_R)$ | $=$ | $\min \begin{cases} \min\limits_{1 \le i \le r} R_{G_1}(dom_L, i) + B_{G_2}(i, dom_R) - i \\ \min\limits_{0 \le i \le r-1} N_{G_1}(dom_L, j) + R_{G_2}(-i-1, dom_R) \\ \min\limits_{0 \le i \le r-1} N_{G_1}(dom_L, -i-1) + R_{G_2}(j, dom_R) \end{cases}$ |
| $B_G(dom_L, dom_R)$ | $=$ | $\min \begin{cases} \min\limits_{1 \le i \le r} B_{G_1}(dom_L, i) + B_{G_2}(i, dom_R) - i \\ \min\limits_{0 \le i \le j \le r-1} L_{G_1}(dom_L, j) + R_{G_2}(-i-1, dom_R) \\ \min\limits_{0 \le i \le j \le r-1} L_{G_1}(dom_L, -i-1) + R_{G_2}(j, dom_R) \end{cases}$ |

originating broadcast at $t_R(G_1) = t_L(G_2)$ is subtracted since its cost is represented in both the cost of dominating $G_1$ and the cost of dominating $G_2$, but it only belongs once in the cost of dominating $G$.

For the case of $L_{G_1}$ and $N_{G_2}$, no subtraction is needed since $t_R(G_1)$ and $t_L(G_2)$ have no originating broadcast. Now either $G_1$ provides overdomination that must at least cover any underdomination in $G_2$, or vice-versa, as represented by the last two lines in the formula for $L_G(dom_L, dom_R)$. When all of these cases are considered, $L_G(dom_L, dom_R)$ is assigned lowest cost of a broadcast function in $G$ with dominance condition $(dom_L, dom_R)$ that has a broadcast originating at $t_L(G)$ but has no broadcast originating at $t_R(G)$.

Similar arguments justify the recursions given for $N_G$, $R_G$, and $B_G$.

### 4.4 The parallel operation

Consider the computation of $L_G(dom_L, dom_R)$ when $G = p(G_1, G_2)$. When $dom_L \ge 0$, we need to compare configurations in $G_1$ and $G_2$ where both graphs have the appropriate originating broadcast at $t_L$ and where one graph provides $dom_R$ at $t_R$. The other graph can then provide anything between $-(dom_R + 1)$ and $dom_R$ at $t_R$, thereby guaranteeing

Table 3: Equations for the $G = p(G_1, G_2)$ Operation.

$$B_G(dom_L, dom_R) = B_{G_1}(dom_L, dom_R) + B_{G_2}(dom_L, dom_R) - dom_L - dom_R$$

$$L_G(dom_L, dom_R) = \begin{cases} \min\limits_{-(dom_R+1) \le i \le dom_R} \left\{ \begin{array}{l} L_{G_1}(dom_L, dom_R) + L_{G_2}(dom_L, i) - dom_L \\ L_{G_1}(dom_L, i) + L_{G_2}(dom_L, dom_R) - dom_L \end{array} \right\} & \text{if } dom_R \ge 0 \\ \min\limits_{dom_R \le i \le -(dom_R+1)} \left\{ \begin{array}{l} L_{G_1}(dom_L, dom_R) + L_{G_2}(dom_L, i) - dom_L \\ L_{G_1}(dom_L, i) + L_{G_2}(dom_L, dom_R) - dom_L \end{array} \right\} & \text{if } dom_R < 0 \end{cases}$$

$$R_G(dom_L, dom_R) = \begin{cases} \min\limits_{-(dom_L+1) \le i \le dom_L} \left\{ \begin{array}{l} R_{G_1}(dom_L, dom_R) + R_{G_2}(i, dom_R) - dom_R \\ R_{G_1}(i, dom_R) + R_{G_2}(dom_L, dom_R) - dom_R \end{array} \right\} & \text{if } dom_L \ge 0 \\ \min\limits_{dom_L \le i \le -(dom_L+1)} \left\{ \begin{array}{l} R_{G_1}(dom_L, dom_R) + R_{G_2}(i, dom_R) - dom_R \\ R_{G_1}(i, dom_R) + R_{G_2}(dom_L, dom_R) - dom_R \end{array} \right\} & \text{if } dom_L < 0 \end{cases}$$

$$N_G(dom_L, dom_R) = \begin{cases} \min\limits_{\substack{-(dom_L+1) \le i \le dom_L \\ -(dom_R+1) \le j \le dom_R}} \left\{ \begin{array}{l} N_{G_1}(dom_L, dom_R) + N_{G_2}(i, j) \\ N_{G_1}(dom_L, j) + N_{G_2}(i, dom_R) \\ N_{G_1}(i, dom_R) + N_{G_2}(dom_L, j) \\ N_{G_1}(i, j) + N_{G_2}(dom_L, dom_R) \end{array} \right\} & \text{if } dom_L, dom_R \ge 0 \\ \min\limits_{\substack{-(dom_L+1) \le i \le dom_L \\ dom_R \le j \le -(dom_R+1)}} \left\{ \begin{array}{l} N_{G_1}(dom_L, dom_R) + N_{G_2}(i, j) \\ N_{G_1}(dom_L, j) + N_{G_2}(i, dom_R) \\ N_{G_1}(i, dom_R) + N_{G_2}(dom_L, j) \\ N_{G_1}(i, j) + N_{G_2}(dom_L, dom_R) \end{array} \right\} & \text{if } dom_L \ge 0 \text{ and } dom_R < 0 \\ \min\limits_{\substack{dom_L \le i \le -(dom_L+1) \\ -(dom_R+1) \le j \le dom_R}} \left\{ \begin{array}{l} N_{G_1}(dom_L, dom_R) + N_{G_2}(i, j) \\ N_{G_1}(dom_L, j) + N_{G_2}(i, dom_R) \\ N_{G_1}(i, dom_R) + N_{G_2}(dom_L, j) \\ N_{G_1}(i, j) + N_{G_2}(dom_L, dom_R) \end{array} \right\} & \text{if } dom_L < 0 \text{ and } dom_R \ge 0 \\ \min\limits_{\substack{dom_L \le i \le -(dom_L+1) \\ dom_R \le j \le -(dom_R+1)}} \left\{ \begin{array}{l} N_{G_1}(dom_L, dom_R) + N_{G_2}(i, j) \\ N_{G_1}(dom_L, j) + N_{G_2}(i, dom_R) \\ N_{G_1}(i, dom_R) + N_{G_2}(dom_L, j) \\ N_{G_1}(i, j) + N_{G_2}(dom_L, dom_R) \end{array} \right\} & \text{if } dom_L, dom_R < 0 \end{cases}$$

that the resulting graph will provide $dom_R$ at $t_R$. The other half of $L_G(dom_L, dom_R)$ (i.e., when $dom_L < 0$) is computed analogously.

Similar arguments justify the recursions given for $N_G$, $R_G$, and $B_G$.

### 4.5 The algorithm

We have defined initializations for leaves in a decomposition tree for a series-parallel graph, and recursive equations for computing the cost of an optimal broadcast domination for the series and parallel operations. The formulas for jackknife operations are a straight-forward adaption of those for the series operation. In the interest of space, they are omitted from this paper. An algorithm to compute the cost of an optimal solution for an entire graph $G$ would simply find the lowest cost for which there is no underdominance in the $\{N, L, R, B\}$ arrays at the root of the decomposition tree. The details of this algorithm, which we call **SPBD**, are given below. It is straightforward, then, to work back down the tree to find the actual broadcast domination function $f$ that corresponds to that optimal cost.

**Algorithm** Series-Parallel Broadcast Domination (**SPBD**)
**Input:** A series-parallel graph $G$ and a decomposition tree $T_d$ for $G$.
**Output:** $\gamma_b(G)$.
$r = rad(G)$;
**for** each leaf $G_i$ in $T_d$ **do**
    Use the formulas in Table 4.2 to compute $N_{G_i}$, $L_{G_i}$, $R_{G_i}$, and $B_{G_i}$;
**repeat**
    $G_i$ = an unprocessed node in $T_d$ whose children have been processed;
    **if** $G_i = s(G_1, G_2)$ **then**
        Use the formulas in Table 4.3 to compute $N_{G_i}$, $L_{G_i}$, $R_{G_i}$, and $B_{G_i}$;
    **else if** $G_i = p(G_1, G_2)$ **then**
        Use the formulas in Table 4.4 to compute $N_{G_i}$, $L_{G_i}$, $R_{G_i}$, and $B_{G_i}$;
**until** the root of $T_d$ has been processed;
$x = \infty$;
**for** $i = 0$ **to** $r$ **do**
    **for** $j = 0$ **to** $r$ **do**
        $x = \min\{x, N_G(i,j), L_G(i,j), R_G(i,j), B_G(i,j)\}$;
$\gamma_b(G) = x$;

The following results verify that Algorithm **SPBD** is correct and give an upper bound on the running time of the algorithm.

**Lemma 4.5** *Every possible minimal broadcast function on a series-parallel graph $G$ is considered during execution of Algorithm* **SPBD**.

**Proof.** By Observation 4.4, Observation 4.3, Lemma 4.1, and Lemma 4.2, the only possible broadcast functions on the leaves that are not included in the initialization are ones that cannot lead to an optimal solution $G$. Thus, at the initial stage of the algorithm every possible minimal broadcast function is represented. At each non-leaf node of the decomposition tree, every combined configuration of $G_1$ and $G_2$ is considered for retention.

A combined configuration is eliminated only when another combined configuration with no worse cost is represented in the same cell $P_G(dom_L, dom_R)$ for $P \in \{N, L, R, B\}$. ∎

**Theorem 4.6** *Given a series-parallel graph $G$, Algorithm* **SPBD** *computes an optimal broadcast domination function of $G$ in $O(nr^4)$ time.*

**Proof.** The correctness follows from Lemma 4.5. For the time complexity, note that there are no more than $n$ nodes in the decomposition tree of a series-parallel graph. For each node we fill in four $(2r+1) \times (2r+1)$ arrays. Filling in the $N_G$ arrays for a parallel operation takes the most time, requiring a minimization over $O(r^2)$ values. Thus, the time required by SP is $O(nr^4)$. ∎

## 5 Trees

In this section we describe and prove correct an $O(nh)$ time algorithm for computing an optimal efficient broadcast function of a rooted tree $T$ of height $h$. In order to make $h$ small, the root of $T$ can be chosen to be a vertex that has the smallest eccentricity, also called a *center vertex* of $T$. This results in a total time of $O(nr)$ for the presented algorithm, where $r = rad(T)$.

Let $T$ be a tree with root $v_r$. We denote the subtree rooted at vertex $v$ by $T_v$, and hence $T = T_{v_r}$. For a subtree $T_v$ and efficient broadcast domination $f$ of $T$, we have the following three possibilities for $T_v$'s root $v$: 1) $f(v) > 0$, which will cover $v$ and vertices both in $T_v$, and in other parts of $T$ through $v$'s parent edge; 2) $f(v) = 0$ and $v$ hears some broadcast originating in $T_v$; and 3) $f(v) = 0$ and $v$ hears some broadcast originating outside of $T_v$ through $v$'s parent edge.

Recall from Section 4.1 the notions of underdomination ($dom_k < 0$) and overdomination ($dom_k \geq 0$). In this section, we use the same descriptions, except we partition overdomination into *proper overdomination* ($dom_k > 0$) and *exact domination* ($dom_k = 0$). Let $cost_v$ be an array associated with vertex $v$ and indexed from $-h$ to $h$. The value of $cost_v[i]$ will be the minimum cost for an efficient broadcast domination of $T_v$ with domination condition $i$. The algorithm will compute the $cost_v$ values in a bottom-up fashion. Analogous to the case of series-parallel graphs, elements of $cost_{v_r}[i]$ for $i < 0$ are not considered when reporting the solution since these represent configurations where $T_{v_r} = T$ contains vertices which hear no broadcast.

Suppose $v$ is a leaf. Then the $cost_v[i] = 0$ for $i < 0$, since an efficient broadcast domination $f$ that covers $v$ through $v$'s parent will not have a broadcast originating at $v$. Note that there is no efficient broadcast function on a single node that has an exact domination. Hence, we set $cost_v[0] = \infty$. Finally, the only way we can achieve a proper overdomination of value $i$ for leaf $v$ is to set $f(v) = i$. A complete formula for $cost_v$ when $v$ is a leaf is given below:

$$cost_v[i] = \begin{cases} 0 & \text{if } i < 0 \\ \infty & \text{if } i = 0 \\ i & \text{if } i > 1 \end{cases}$$

Now consider computation of the cost vector $cost_v[i]$ for $T_v$ where $v$ has children $v_1, v_2, \ldots, v_c$. When $i < 0$, an underdomination of $i$ in $T_v$ is equivalent to an underdomination of $i+1$ in each $T_{v_k}$, $1 \le k \le c$ (exact domination if $i+1=0$). Thus, the cost incurred for $T_v$ is the sum of $cost_{v_k}[i+1]$ over all children $v_k$ of $v$. When $i=0$, there cannot be a broadcast originating at $v$, and since we require $f$ to be efficient, exactly one child subtree of $v$ must have a proper overdomination of 1, and all other child subtrees must have exact domination. If $i > 0$, either $f(v) = i$, or $f(v) = 0$ and exactly one child subtree of $v$ has proper overdomination $i+1$. If $f(v) = i$, then each of the child subtrees must have an underdomination of $-i$. If, on the other hand, exactly one child subtree has proper overdomination $i+1$, then all other child subtrees must have underdomination $-i$. These relationships are formalized in the equations given below when $v$ is not a leaf.

$$BestChild_v(i) = \min_{1\le k\le c} \left\{ cost_{v_k}[i+1] + \sum_{1\le j\le c,\, j\ne k} cost_{v_j}[-i] \right\}$$

$$cost_v[i] = \begin{cases} \displaystyle\sum_{1\le k\le c} cost_{v_k}[i+1] & \text{if } i < 0 \\ \displaystyle\min_{1\le k\le c} \left\{ cost_{v_k}[1] + \sum_{1\le j\le c,\, j\ne k} cost_{v_j}[0] \right\} & \text{if } i = 0 \\ \min \left\{ (i + \displaystyle\sum_{1\le k\le c} cost_{v_k}[-i]),\ BestChild_v(i) \right\} & \text{if } i > 0 \end{cases}$$

To show that the time complexity of the described algorithm is $O(nh)$, we will argue that each $cost_v[i]$ needs to be written or read a constant number of times. Since there are $2h+1$ values in each of the $n$ $cost_v$ arrays, the result follows. Clearly computation of the $cost_v[i]$ values for all leaves $v$ fits within the desired time complexity. Thus, we focus on the time required to compute all the non-leaf values. Observe that when a non-leaf vertex $v$ is processed, it must use the values in the cost arrays of each of its children, say $v_1, v_2, \ldots, v_c$. But after the values of $cost_v$ are filled in, the child cost arrays are never accessed again. From the equation for calculating $cost_v$ for non-leaf nodes (see Table 4), it is clear that each entry of the cost tables of the children of $v$ is accessed only a constant number of times, establishing the desired result.

With an algorithm called **TBD**, we propose an implementation of the dynamic programming approach described above. **TBD** computes the $cost_v$ array of each vertex $v$ in $T$, starting from the leaves, and processing a vertex only after the cost arrays of its children have been computed.

**Algorithm** Tree Broadcast Domination (**TBD**)
**Input:** A tree $T$ rooted at a center vertex $v_r$.
**Output:** $\gamma_b(T)$.
**for** every vertex $v$ in $T$ (traversed in post order) **do**
  **for** $i = -h$ **to** $h$ **do**
    $Sum(i) = 0$;
    $InfinityCount(i) = 0$;

**for** each child $v_k$ of $v$ **do**
    **if** $cost_{v_k}[i] = \infty$ **then**
        $InfinityCount(i) = InfinityCount(i) + 1;$
    **else**
        $Sum(i) = Sum(i) + cost_{v_k}[i];$
**for** $i = -h$ **to** $-1$ **do**
    $cost_v[i] = Sum(i+1);$
$cost_v[0] = \infty;$
**if** $InfinityCount(0) \leq 1$ **then**
    **for** each child $v_k$ of $v$ **do**
        **if** $cost_{v_k}[0] = \infty$ **then**
            $cost_v[0] = \min\{cost_v[0],\ cost_{v_k}[1] + Sum(0)\};$
        **else if** $InfinityCount(0) = 0$ **then**
            $cost_v[0] = \min\{cost_v[0],\ cost_{v_k}[1] + Sum(0) - cost_{v_k}[0]\};$
**for** $i = 1$ **to** $h-1$ **do**
    $BestChild(i) = \infty;$
    **if** $InfinityCount(-i) \leq 1$ **then**
        **for** each child $v_k$ of $v$ **do**
            **if** $cost_{v_k}[-i] = \infty$ **then**
                $BestChild(i) = \min\{BestChild(i),\ cost_{v_k}[i+1] + Sum(-i)\};$
            **else if** $InfinityCount(i) = 0$ **then**
                $BestChild(i) = \min\{BestChild(i),\ cost_{v_k}[i+1] + Sum(-i) - cost_{v_k}[-i]\};$
        $cost_v[i] = \min\{(i + Sum(-i)),\ BestChild(i)\};$
    $cost_v[h] = h + Sum(-h);$
$\gamma_b(G) = \min_{0 \leq i \leq h}\{cost_{v_k}[i]\};$

## 6 Summary

The primary results in this paper are dynamic programming algorithms for finding optimal broadcast domination functions in classes of graphs that require non-trivial broadcast functions for optimality. These include:

- An $O(n^3)$ time algorithm that finds $\gamma_b(G)$ for any interval graph $G$. This algorithm can easily be adapted to provide an efficient optimal broadcast domination function $f$ of $G$.

- An $O(nr^4)$ time algorithm that finds $\gamma_b(G)$ for any series-parallel graph $G$. This algorithm can easily be adapted to provide an optimal broadcast domination function even under the restriction of bounded maximum broadcast power, i.e., when $f(v) \leq k$ for all $v \in V$ for some fixed constant $1 \leq k \leq rad(G)$.

- An $O(nr)$ time algorithm that finds $\gamma_b(T)$ for any tree $T$. This algorithm can easily be adapted to provide an efficient optimal broadcast domination function $f$ of $T$.

It is worth noting that the series-parallel algorithm is not "anticipated" in the same sense that other fast algorithms are, for problems restricted to recursively constructible instances. That is, it is not evident that the problem of computing $\gamma_b$ is expressible in any of the formal contexts that have been developed for graph problems on recursive structures [5]. Among these is the predicate calculus developed in [6] where if a given problem is shown to be expressible in the calculus, then a polynomial-time algorithm for its solution is guaranteed for the problem on *any* recursive graph. Thus, if a legal expression for the given problem can be formed, it is generally straightforward to create a practical algorithm. On the other hand, the more interesting outcome is to find a polynomial time algorithm for a problem whose formal expressibility status is, if not explicitly known to be impossible, at least ambiguous. This is the case with the computation of $\gamma_b$, thus lending interest to the creation of the series-parallel algorithm.

## References


[1] J. Bar-Ilan, G. Kortsarz, and D. Peleg, *How to allocate network centers*, J. Algorithms, 15 (1993), pp. 385–415.

[2] C. Berge, *Theory of Graphs and its Applications*, no. 2 in Collection Universitaire de Mathematiques, Dunod, Paris, 1958.

[3] K. S. Booth and J. H. Johnson, *Dominating sets in chordal graphs*, SIAM J. Comput., 11 (1982), pp. 191–199.

[4] K. S. Booth and G. S. Lueker, *Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms*, J. Comput. System Sci., 13 (1976), pp. 335–379.

[5] R. B. Borie. personal communication, 2002.

[6] R. B. Borie, R. G. Parker, and C. A. Tovey, *Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families*, Algorithmica, 7 (1992), pp. 555–581.

[7] A. K. Dewdney, *Fast Turing reductions between problems in NP*, Tech. Rep. 71, Dept. of Computer Science, University of West Ontario, 1981.

[8] J. E. Dunbar, D. J. Erwin, T. W. Haynes, S. M. Hedetniemi, and S. T. Hedetniemi, *Broadcasts in graphs*, (2002). Submitted.

[9] D. J. Erwin, *Dominating broadcasts in graphs*, (2002). Submitted.

[10] M. Farber, *Domination, independent domination, and duality in strongly chordal graphs*, Disc. Appl. Math., 7 (1984), pp. 115–130.

[11] M. Farber and J. M. Keil, *Domination in permutation graphs*, J. Algorithms, 6 (1985), pp. 309–321.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability*, W. H. Freeman and Co., 1978.

[13] P. C. Gilmore and A. J. Hoffman, *A characterization of comparability graphs and of interval graphs*, Canadian Journal of Mathematics, 16 (1964), pp. 539–548.

[14] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, 1980.

[15] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater, *Domination in Graphs: Advanced Topics*, Marcel Dekker, New York, 1998.

[16] ——, *Fundamentals of Domination in Graphs*, Marcel Dekker, New York, 1998.

[17] M. A. Henning, *Distance domination in graphs*, in Domination in Graphs: Advanced Topics, T. W. Haynes, S. T. Hedetniemi, and P. J. Slater, eds., Marcel Dekker, New York, 1998, pp. 321–349.

[18] C. W. Ho and R. C. T. Lee, *Counting clique trees and computing perfect elimination schemes in parallel*, Information Processing Letters, 31 (1989), pp. 61–68.

[19] S. B. Horton, *The Optimal Linear Arrangement Problem: Algorithms and Approximation*, PhD thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, 1997.

[20] S. B. Horton, R. G. Parker, and R. B. Borie, *On minimum cuts and the linear arrangement problem*, Disc. Appl. Math., 103 (2000), pp. 127–139.

[21] D. Kratsch, *Domination and total domination in asteroidal triple-free graphs*, Tech. Rep. Math/Inf/96/25, F.-Schiller-Universität, Jena, 1996.

[22] C. L. Liu, *Introduction to Combinatorial Mathematics*, McGraw-Hill, New York, 1968.

[23] O. Ore, *Theory of Graphs*, no. 38 in American Mathematical Society Publications, AMS, Providence, 1962.

[24] M. Richey and R. Parker, *On finding spanning eulerian subgraphs*, Naval Research Logistics Quarterly, 32 (1985), pp. 443–455.

[25] F. S. Roberts, *Indifference graphs*, in Proof Techniques in Graph Theory, F. Harary, ed., Academic Press, 1969, pp. 139 – 146.

[26] P. J. Slater, *R-domination in graphs*, J. Assoc. Comput. Mach., 23 (1976), pp. 446–450.

[27] J. Valdes, R. E. Tarjan, and E. L. Lawler, *The recognition of series parallel digraphs*, SIAM J. Comput., 11 (1982), pp. 298–313.

[28] T. Wimer and S. Hedetniemi, *K-terminal recursive families of graphs*, in Proceedings of the 250th Anniversary Conference on Graph Theory, Fort Wayne, IN, 1986.