AKCE International Journal of Graphs and Combinatorics

ISSN: 0972-8600 (Print) 2543-3474 (Online) Journal homepage: www.tandfonline.com/journals/uakc20

Unsolved Algorithmic Problems on Trees

Stephen T. Hedetniemi & T.W. Haynes (Communicated by)

To cite this article: Stephen T. Hedetniemi & T.W. Haynes (Communicated by) (2006) Unsolved Algorithmic Problems on Trees, AKCE International Journal of Graphs and Combinatorics, 3:1, 1-37, DOI: 10.1080/09728600.2006.12088803

To link to this article: https://doi.org/10.1080/09728600.2006.12088803

Published online: 10 Mar 2020.

AKCE J. Graphs. Combin., 3, No. 1 (2006), pp. 1-37

Unsolved Algorithmic Problems on Trees

Stephen T. Hedetniemi
Clemson University
Clemson, SC 29634
e-mail: *hedet@cs.clemson.edu*

Communicated by: T.W. Haynes
Received 13 July 2005; accepted 25 April 2006

Abstract

The literature on algorithms and complexity results for domination and domination-related problems is extensive, and deals with a somewhat bewildering variety of concepts and definitions. At the same time, what one observes is that no matter what the definition or concept is, an algorithm problem related to that concept is almost always solvable in linear time when the inputs are restricted to trees. The fact that this is so follows almost directly from the now well-developed theory of algorithms on partial k-trees, or graphs of bounded treewidth. In light of this, it is somewhat surprising that quite a few algorithmic problems on trees remain unsolved. In this paper we offer a list of more than 60 algorithm problems that have yet to be solved for trees, quite a few of which are newly defined here. For about 50 of these problems the associated NP-completeness questions have not yet been settled either.

Keywords:

2000 Mathematics Subject Classification: 05C

1. Introduction

In order to understand the problems to be discussed in this paper, we will need some definitions and terminology.

Let $G = (V, E)$ be a graph of *order* $n = |V|$.

Let v be a vertex in V. The *open neighborhood* $N(v)$ of v is the set of all vertices adjacent to v. The *closed neighborhood* $N[v]$ of v is the set $N(v) \cup \{v\}$. For a set S of vertices, the *open neighborhood* $N(S)$ *of* S is defined by $N(S) = \cup_{v \in S} N(v)$ and the *closed neighborhood* $N[S]$ *of* S is the set $N(S) \cup S$. Given a set $S \subseteq V$, by $G[S]$ we mean the *subgraph of* G *induced by* S, where $G[S] = (S, E \cap (S \times S))$.

A set $S \subseteq V$ is called *independent* if no two vertices in S are adjacent. The *vertex independence number* $\beta_0(G)$ equals the maximum size of an independent set in G, while the *lower independence number*(also called the *independent domination number*) $i(G)$, equals the minimum size of a maximal independent set in G.

A set $S \subseteq V$ is a *dominating set* if every vertex in $V - S$ is adjacent to a vertex in S, that is, $N[S] = V$. The *domination number* $\gamma(G)$ is the minimum size of a dominating set S in G; such a set is called a γ*-set* of G. The *upper domination number* $\Gamma(G)$ equals the maximum size of a minimal dominating set in G.

A set S is a *packing* if for every vertex $v \in V$, $|N[v] \cap S| \leq 1$. The *packing number* $P(G)$ equals the maximum size of a packing in G, while the it lower packing number $p(G)$ equals the minimum size of a maximal packing in G. The following well-known inequality chain relates these invariants:

$$p(G) \leq P(G) \leq \gamma(G) \leq i(G) \leq \beta_0(G) \leq \Gamma(G).$$

Given a set $S \subseteq V$ and a vertex $u \in S$, an *external private neighbor* of u (with respect to S) is a vertex $v \in V - S$ which is adjacent to u but no other vertex in S. A vertex $u \in S$ is its *own private neighbor* if it is not adjacent to any vertex in S. Finally, an *internal private neighbor* of u (with respect to S) is a vertex $v \in S$ that is adjacent to u but is not adjacent to any other vertex in S.

A set $S \subseteq V$ is called *irredundant* if every vertex in S is either its own private neighbor or has an external private neighbor, with respect to S. This can also be defined as follows. Define the *private neighbor set* of a vertex $v \in S$, to equal the set $pn[v, S] = N[v] - N[S - \{v\}]$. A set S is irredundant if for every vertex $v \in S$, $pn[v, S] \neq \emptyset$. The *irredundance number* $ir(G)$ equals the minimum size of a maximal irredundant set in G, while the *upper irredundance number* $IR(G)$ equals the maximum size of an irredundant set in G.

The concepts of independence, domination and irredundance are related in a natural way, as first observed by Cockayne, Hedetniemi and Miller in 1978 [25]:

Theorem 1. *For any graph* G*,*

$$ir(G) \leq \gamma(G) \leq i(G) \leq \beta_0(G) \leq \Gamma(G) \leq IR(G).$$

This inequality chain has become the focus of a substantial amount of research in domination theory, and will be a reference for many of the open problems to follow.

Nearly all of the algorithm problems in the following sections start by considering the class of all sets of vertices (or edges) having some property $\mathcal{P}$ in a graph G, each set in which is called a $\mathcal{P}$*-set*. Particular focus is on the class of all minimal or maximal $\mathcal{P}$-sets, and one attempts to determine the minimum and maximum sizes of a minimal $\mathcal{P}$-set, or the minimum and maximum sizes of a maximal $\mathcal{P}$-set.

In this paper we will occasionally depart from traditional notation. In an effort to provide a more uniform system of notation, we will use subscripts or lower case symbols to refer to minimum size parameters (e.g. $\beta_1(G)$ or $\gamma_t(G)$) and superscripts or upper case symbols to refer to maximum size parameters (e.g. $\beta^1(G)$ or $\Gamma_t(G)$).

A property $\mathcal{P}$ is called *hereditary* if whenever a set S has property $\mathcal{P}$, so does every subset of S. For example, the property of being an independent set is hereditary. For

hereditary properties one considers the class of all maximal $\mathcal{P}$-sets, and one seeks first an algorithm to determine the maximum size of a $\mathcal{P}$-set, and second, an algorithm to determine the minimum size of a **maximal** $\mathcal{P}$-set; we refer to this second problem as the *mini-maximal* problem. One reason for having an interest in hereditary properties is that a hereditary $\mathcal{P}$-set is maximal if and only if it is 1-maximal, meaning that in order to decide if a set S is maximal, all one need do is check every set $S \cup x$, for every x not in S [21].

Similarly, a property $\mathcal{P}$ is called *super-hereditary* if whenever a set S has property $\mathcal{P}$, so does every superset of S. For example, the property of being a vertex cover is super-hereditary. For super-hereditary properties $\mathcal{P}$, one considers the class of all **minimal** $\mathcal{P}$-sets. One then seeks to determine the minimum size of a $\mathcal{P}$-set and the maximum size of a minimal $\mathcal{P}$-set. We refer to this second problem as the *maxi-minimal* problem. Similar to hereditary properties, the property of being a minimal super-hereditary $\mathcal{P}$-set can be determined by considering every set $S - \{x\}$, for every element $x \in S$, that is, a super-hereditary set S is minimal if and only if for every $x \in S$, $S - \{x\}$ is not a $\mathcal{P}$-set [21].

Thus, most of the algorithmic problems that follow come in pairs, either those of finding the maximum and mini-maximal sizes, or of finding the minimum and maxi-minimal sizes of a $\mathcal{P}$-set. In many instances, only the problems of finding either a minimum or a maximum size $\mathcal{P}$-set have been considered, while the maxi-minimal or mini-maximal problems have not been considered. In these cases, we will indicate that the problem invariant is newly mentioned here by saying (NEW HERE). In these cases it would be interesting to undertake a graph theoretical study of the properties and values of these new invariants and their relationships with other related invariants.

The following sections contain a listing of more than 60 algorithm problems for trees that do not appear to have been solved, and an almost equally long listing of associated NP-completeness questions that do not appear to have been settled. For sure, many of these problems are easy to solve and are simple exercises; the fact that they have not been solved is simply because they may have been overlooked. But at the same time, many of these problems are truly difficult, and some may never be solved. Hopefully the reader will find at least some of these problems to be fascinating to ponder.

If the reader knows that any of these problems have been solved, please send an e-mail with this information to: hedet@cs.clemson.edu.

The interested reader is encouraged to refer to two chapters in [64]: the first by Kratsch [78] contains a survey of algorithmic results on domination and related invariants, while the second, by Hedetniemi, McRae and Parks [74] contains a survey of NP-completeness results for many domination and domination-related invariants.

In addition, anyone interested in developing algorithms for solving these types of problems is encouraged to understand the different approaches that can be used to construct polynomial time algorithms for problems that are restricted to partial k-trees for some fixed k, or graphs of bounded treewidth [2, 3, 4, 8, 96].

2. Irredundance Numbers

We have previously defined the irredundance number $ir(G)$ and the upper irredundance number $IR(G)$. There are three other types of irredundance that have received attention in the literature.

We say that a set $S \subset V$ is:

1. *open irredundant* if every vertex in S has an external private neighbor. The *open irredundance number* $oir(G)$ equals the minimum size of a maximal open irredundant set in G, and the *upper open irredundance number* $OIR(G)$ equals the maximum size of an open irredundant set in G.

2. *open-open irredundant* if every vertex in S has either an internal or an external private neighbor. The *open-open irredundance number* $ooir(G)$ equals the minimum size of a maximal open-open irredundant set in G, while the *upper open-open irredundance number* $OOIR(G)$ equals the maximum size of an open-open irredundant set in G.

3. *closed-open irredundant* if every vertex in S has a private neighbor of some kind, either itself, an internal or an external private neighbor. The *closed-open irredundance number* $coir(G)$ equals the minimum size of a maximal closed-open irredundant set in G, while the *upper closed-open irredundance number* $COIR(G)$ equals the maximum size of a closed-open irredundant set in G.

First introduced by Cockayne, Hedetniemi and Miller in 1979 [25], there are now more than 100 papers on irredundance in graphs; the reader is referred to a discussion of this literature in [9, 15, 18, 20, 21, 43, 63, 73]. The complexity of various irredundance numbers is discussed in the paper by Fellows, Fricke, Hedetniemi and Jacobs [42], who first demonstrated the NP-completeness of the decision problems associated with $IR(G)$ and $COIR(G)$, and by Golumbic and Laskar [58]. The decision problem associated with $OIR(G)$ was first settled by Even, Goldreich, Moran and Tong [38], while that for $OOIR(G)$ was settled by Cameron [11].

The complexity of the (lower) irredundance number $ir(G)$ was first settled by Pfaff, Laskar and S.T. Hedetniemi [89] who showed the NP-completeness for bipartite graphs, and by Laskar, Pfaff, Hedetniemi and Hedetniemi [79] who demonstrated the NP-completeness for chordal graphs. The NP-completeness of the decision problems associated with $oir(G)$, $ooir(G)$ and $coir(G)$ for bipartite graphs, chordal graphs, line graphs and line graphs of bipartite graphs was shown by McRae [85].

The values of the lower irredundance numbers of trees have proved to be very difficult to compute. An impressive linear algorithm for computing the irredundance number $ir(T)$ of any tree T was constructed in 1987 by Bern, Lawler, and Wong [6]. This is a dynamic programming algorithm which consists of a 20 by 20 matrix of possible combinations of

subtrees that can combine to form an optimal solution. Although the authors provided little or no proof of the correctness of this algorithm, there is little doubt that it is correct.

It seems reasonable, therefore, that a similar complexity will be required in order to compute the value of the other three lower irredundance numbers.

Algorithm 1. *Design an algorithm for computing the value of* $oir(T)$ *for any tree* T.

Algorithm 2. *Design an algorithm for computing the value of* $ooir(T)$ *for any tree* T.

Algorithm 3. *Design an algorithm for computing the value of* $coir(T)$ *for any tree* T.

In 1990 [77], Jacobson and Peters showed that for any tree T, $\beta_0(T) = \Gamma(T) = IR(T)$. And since a simple greedy algorithm, first designed in 1966 by Daykin and Ng [29], will suffice to compute the value of $\beta_0(T)$, it is easy to compute the value of $IR(T)$ for any tree T.

It is also well known that for any tree T, $OIR(T) = \beta^1(T)$, for example, as noted by Golumbic and Laskar [58], where $\beta^1(T)$ denotes the well known matching number of a tree T (cf. Section). And since a simple greedy algorithm will suffice to compute the value of $\beta^1(T)$ [88], it is easy to compute the value of $IR(T)$ for any tree T.

In the same paper [58], the authors point out that Fricke, S.M. Hedetniemi and Laskar [47] had shown that for any tree T, $\beta^*(T) = OOIR(T)/2$, where $\beta^*(T)$ is the induced matching number, to be defined later in Section . They then show that the same equality holds for all bipartite graphs. Since it was shown by Cameron [11] that the induced matching number $\beta^*(G)$ can be computed in polynomial time for all chordal graphs, it follows that the value of $OOIR(T)$ can be computed in polynomial time for all trees.

Similarly, in [58] it is shown that for all bipartite and circular arc graphs, $COIR(G) = \beta^{[1]}(G)$, where $\beta^{[1]}(T)$ is the 1-dependence number, to be defined later in Section . Since it is easy to see that the $[1]$-dependence number of a tree can be computed in linear time, using a simple, Wimer-style algorithm, [96] and [30], it follows that $COIR(T)$ can be computed in linear time for any tree T.

2.1 Independent Open Irredundance

In 1999, Cockayne [12] introduced the study of a large class of generalized irredundant sets in graphs. Each type of a generalized irredundant set $S \subset V$ is defined by the types of private neighbors (i.e. self, internal or external) that each vertex in the set must have. For example, if every vertex in S is its own private neighbor then the set S is an independent set, and if every vertex in S either is its own private neighbor or has an external private neighbor, then S is an irredundant set. Among the many possible Boolean functions on three variables that describe a possible type of generalized irredundance, one of the more interesting is the following.

A set S is called an *independent open irredundant set* (or *ioir-set* if S is an independent set and every vertex in S has an external private neighbor. In [12], Cockayne identifies

12 types of generalized irredundant sets the properties of which are hereditary. Perhaps the most interesting of these not previously mentioned are the ioir-sets. One can therefore define *ioir(G)* to equal the minimum size of a maximal ioir-set and *IOIR(G)* to equal the maximum size of an ioir-set.

These generalized irredundant sets are also studied by Finbow in [43] and Cockayne and Finbow in [18].

NP-problem 1. *Does G have a maximal ioir-set of size at most k ?*

Algorithm 4. *Design an algorithm to compute the value of $ioir(T)$ for any tree T .*

NP-problem 2. *Does G have an ioir-set of size at least k ?*

Algorithm 5. *Design an algorithm to compute the value of $IOIR(T)$ for any tree T .*

2.2 Total Irredundance

In 1993, S.M. Hedetniemi, S.T. Hedetniemi and Jacobs [70] generalized the concept of irredundance in a very natural way. A set $S \subset V$ is said to be a *total irredundant set* if every vertex $v \in V$ has a private neighbor with respect to S. This means that every vertex in S either is its own private neighbor, or has an external private neighbor with respect to S, and every vertex in $V - S$ either is not adjacent to any vertex in S, or is adjacent to at least one vertex in $V - S$ which is not adjacent to any vertex in S. This definition gives rise to the two invariants $ir_t(G)$ and $IR_t(G)$, which equal the minimum and maximum sizes of a maximal total irredundant set in G. In [70] it is shown that the value of $IR_t(T)$ can be computed in linear time for any tree T, while the decision problem corresponding to $IR_t(G)$ is NP-complete. No results, algorithm or NP-completeness, were given for $ir_t(G)$.

Two other papers on total irredundance have appeared since then, [62] and [39].

NP-problem 3. *Does G have a maximal total irredundant set of size at most k ?*

Algorithm 6. *Design an algorithm to compute the value of $ir_t(T)$ for any tree T .*

Although it has not been done in the previous papers on total irredundance, one can easily define the other types of total irredundance: *open total irredundance* ($oir_t(G)$ and $OIR_t(G)$), *open-open total irredundance* ($ooir_t(G)$ and $OOIT_t(G)$) and *closed-open total irredundance* ($coir_t(G)$ and $COIR_t(G)$)(NEW HERE).

NP-problem 4. *Does G have a maximal total open irredundant set of size at most k ?*

Algorithm 7. *Design an algorithm to compute the value of $oir_t(T)$ for any tree T .*

NP-problem 5. *Does G have a maximal total open-open irredundant set of size at most k ?*

Algorithm 8. *Design an algorithm to compute the value of* $ooir_t(T)$ *for any tree* T.

NP-problem 6. *Does* G *have a maximal total closed-open irredundant set of size at most* k *?*

Algorithm 9. *Design an algorithm to compute the value of* $coir_t(T)$ *for any tree* T.

NP-problem 7. *Does* G *have a total open irredundant set of size at least* k *?*

Algorithm 10. *Design an algorithm to compute the value of* $OIR_t(T)$ *for any tree* T.

NP-problem 8. *Does* G *have a total open-open irredundant set of size at least* k *?*

Algorithm 11. *Design an algorithm to compute the value of* $OOIR_t(T)$ *for any tree* T.

NP-problem 9. *Does* G *have a total closed-open irredundant set of size at least* k *?*

Algorithm 12. *Design an algorithm to compute the value of* $COIR_t(T)$ *for any tree* T.

2.3 External Redundance

In 1994, McRae [84] introduced the concept of external redundance, which was designed to extend the irredundance, domination, independence inequality chain, and was based on an analysis of the conditions under which an irredundant set is a maximal irredundant set. This was further discussed by Cockayne, Hattingh, Hedetniemi, Hedetniemi and McRae in 1997 [21].

We say that a set $S \subset V$ is *external redundant* if for every vertex $v \in V-S$, there exists a vertex $w \in S \cup \{v\}$, such that $pn[w, S \cup \{v\}] = \emptyset$, and if $w \in S$ then $pn[w, S] \neq \emptyset$. Stated in other words, an external redundant set has the property that any attempt to add a vertex $v \in V - S$ to S results in the creation of a new vertex which does not have a private neighbor; either the vertex v will not have a private neighbor in the set $S \cup \{v\}$, or a vertex in S which has a private neighbor with respect to S no longer has a private neighbor with respect to the larger set $S \cup \{v\}$.

An equivalent definition might be helpful. Given a set $S \subseteq V$, we define the *private neighbor count of* S to equal the value $pnc(S) = |\{v \in S : pn[v, S] \neq \emptyset\}|$. Thus, the private neighbor count of a set S equals the number of vertices in S which have private neighbors. A set S is called external redundant if for every vertex $v \in V - S$, $pnc(S \cup \{v\}) \leq pnc(S)$. Thus, you cannot increase the private neighbor count of S by adding a vertex to S.

The *external redundance number* $er(G)$ equals the minimum size of a maximal external redundant set in G, while the *upper external redundance number* $ER(G)$ equals the maximum size of an external redundant set in G. In [21] no discussion is given of the complexity of the decision problem associated with either $er(G)$ or $ER(G)$, and no algorithms are mentioned.

It is pointed out however that:

Lemma 1. *If $S \subseteq V$ is a maximal irredundant set, then S is a minimal external redundant set.*

Thus, one can include the external redundance numbers in the previous inequality chain, as follows:

Theorem 2. *For any graph G,*

$$er(G) \leq ir(G) \leq \gamma(G) \leq i(G) \leq \beta_0(G) \leq \Gamma(G) \leq IR(G) \leq ER(G).$$

NP-problem 10. *Does G have an external redundant set of size at most k?*

Algorithm 13. *Design an algorithm to compute the value of $er(T)$ for any tree T.*

NP-problem 11. *Does G have a minimal external redundant set of size at least k?*

Algorithm 14. *Design an algorithm to compute the value of $ER(T)$ for any tree T.*

2.4 Perfect Neighborhood Sets

Let $G = (V, E)$ be a graph and let $S \subset V$ be a an arbitrary set of vertices. A vertex $v \in V$ is said to be *perfect* with respect to S if $|N[v] \cap S| = 1$. Thus, if a vertex $v \in V$ is perfect then it is either an element of S that is an isolate in the subgraph $G[S]$ induced by S, or is not in S but is adjacent to exactly one vertex in S. A vertex is said to be *ap* if it is not perfect but is adjacent to a perfect vertex (with respect to S). A set $S \subset V$ is called a *perfect neighborhood set* if every vertex $v \in V$ is either *perfect* or *ap* with respect to S. The *perfect neighborhood number* $\theta(G)$ equals the minimum size of a perfect neighborhood set in G. The *upper perfect neighborhood number* $\Theta(G)$ equals the maximum size of a perfect neighborhood set in G.

Perfect neighborhood sets were introduced by Hedetniemi, Hedetniemi and Henning in 1997 [68], and by Fricke, Haynes, Hedetniemi, Hedetniemi and Henning in 1999 [46]. The problem of deciding if a graph G has a perfect neighborhood set of size at most k was shown in [68] to be NP-complete for bipartite and chordal graphs. Also in [68] a linear algorithm is constructed for computing the value of $\theta(T)$ for any tree T.

The upper perfect neighborhood number was defined and studied in [46], where it was shown, somewhat surprisingly, that for all graphs G, $\Theta(G) = \Gamma(G)$. Thus, the value of $\Theta(T)$ can be computed for all trees T, since for all trees T, $\beta_0(T) = \Gamma(T)$.

A similar definition can be given for open perfect neighborhood sets, in which we say that a vertex $v \in V$ is called *open perfect* if $|N(v) \cap S| = 1$. A set $S \subset V$ is an *open perfect neighborhood set* if every vertex $v \in V$ is either *open perfect* or is adjacent to an *open perfect* vertex. The *open perfect neighborhood number* $\theta_o(G)$ equals the minimum size of an open perfect set in G, and the *upper open perfect neighborhood number* $\Theta_o(G)$ equals the maximum size of an open perfect neighborhood set.

The problem of deciding if a graph has an open perfect neighborhood set of size at most k was shown to be NP-complete for bipartite and chordal graphs by Hedetniemi, Jacobs, Laskar and Pillone in 1996 [72], however no algorithms for trees were constructed for computing the values of $\theta_o(G)$ or $\Theta_o(G)$.

Algorithm 15. *Design an algorithm for computing the value of* $\theta_o(T)$ *for any tree* T.

NP-problem 12. *Does* G *have an open perfect neighborhood set of size at least* k *?*

Algorithm 16. *Design an algorithm for computing the value of* $\Theta_o(T)$ *for any tree* T.

2.5 Annihilation Number

In a series of papers, Cockayne, Favaron, Mynhardt and Puech [26, 16, 17, 24] introduced several new irredundance-related invariants and established an interesting sequence of inequalities among them, especially for trees. But noticeably no NP-completeness results are given nor are algorithms for trees constructed.

The concept of a private neighborhood $pn[u, S]$ of a vertex u in a set S enables one to define a partition $V(G) = \{I, A, EI, EA, M, R\}$, where:

- $I = \{u \in S | u$ is an Isolate in $G[S]\}$,
- $A = S - I$, the vertices in S that are Adjacent to another vertex in S,
- $EI = \{v \in V - S | v$ is an External private neighbor of some vertex in $I\}$,
- $EA = \{v \in V - S | v$ is an External private neighbor of some vertex in $A\}$,
- $M = \{v \in V - S || N(v) \cap S| \geq 2\}$, the vertices in $V - S$ that are Multiply dominated by S,
- $R = V - N[S]$, the Remaining vertices.

With these sets defined, we can say that:

- S is a dominating set if $N[S] = V$,
- S is irredundant if every vertex in A has a private neighbor in EA,
- S is a packing if $M \cup A = \emptyset$,
- S is a perfect neighborhood set (*PN-set*) if the set $I \cup EI \cup EA$ is a dominating set.

A vertex $v \in V - S$ is said to *annihilate* a vertex $u \in S$ if $\emptyset \neq pn[u, S] \subseteq N[v]$. That is, vertex $u \in S$ has a private neighbor with respect to S, but does not have a private neighbor with respect to the set $S \cup \{v\}$. Let $Ann(S) = \{v \in V - S | v$ annihilates some vertex $u \in S\}$. We say that a set S is *U-annihilated* by every subset $U \subseteq Ann(S)$. In [20] Cockayne, Grobler, Hedetniemi and McRae show:

Theorem 3. *A set S is maximal irredundant if and only if S is irredundant and N[R]-annihilated.*

One can observe that the class of $N[R]$-annihilated sets is the same as the class of external redundant sets and is contained in the class of R-annihilated sets (called it *Ra-sets*). Sets that are both R-annihilated and irredundant, hereafter called *Rai-sets*, were first introduced by Favaron and Puech [41], who called them *semi-maximal irredundant sets*.

Define *ra(G)* to equal the smallest size of an Ra-set (R-annihilated set) in G, and *rai(G)* to equal the smallest size of an Rai-set (R-annihilated and irredundant set) in G.

The following impressive inequality chain was proved in [17], where (i) $\gamma_2(G)$ equals the *distance-2 domination number* of G, that is the size of a smallest set S having the property that every vertex in V is distance at most 2 from some vertex in S, (ii) $\theta_i(G)$ equals the smallest cardinality of an independent perfect neighborhood set in G, and (iii) $p(G)$ and $P(G)$ are the two packing numbers.

Theorem 4. *For any tree T,*

$$\gamma_2(T) \leq \theta(T) \leq \theta_i(T) \leq p(T) \leq ra(T) \leq \begin{matrix} rai(T) \\ er(T) \end{matrix} \leq ir(T) \leq P(T) = \gamma(T).$$

The complexity and algorithm questions for these invariants have not been addressed.

NP-problem 13. *Does G have an independent perfect neighborhood set of size at most k?*

Algorithm 17. *Design an algorithm for computing the value of $\theta_i(T)$ for any tree T.*

NP-problem 14. *Does G have an Ra-set set of size at most k?*

Algorithm 18. *Design an algorithm for computing the value of $ra(T)$ for any tree T.*

NP-problem 15. *Does G have an Rai-set set of size at most k?*

Algorithm 19. *Design an algorithm for computing the value of $rai(T)$ for any tree T.*

3. k-dependence, k-dependent domination and k-dependent irredundance

A set $S \subset V$ is called a *k-dependent set* if the maximum degree of a vertex in $G[S]$ is at most k. The *k-dependence number* $\beta_{[k]}(G)$ equals the maximum size of a k-dependent set in G, while the *lower k-dependence number* $i_{[k]}(G)$ equals the minimum size of a maximal k-dependent set in G. The concept of k-dependent sets was first introduced by Fink and Jacobson in 1984 [44].

In 1989, Hare, Hedetniemi and Hedetniemi [60] showed that the problem of deciding if a graph G has a 1-dependent set of size at least k is NP-complete for bipartite and chordal graphs, or any class of graphs, for which DOMINATING SET is NP-complete, that is closed under the corona GoK_1 operation. They also constructed an algorithm for computing the value of $\beta_{[1]}(T)$ for any tree T and the value of $i_{[1]}(T)$ for any tree T.

NP-problem 16. *Does G have a maximal 1-dependent set of size at most k?*

The concept of k-dependence extends quite naturally to both domination and irredundance. In 2002, Favaron, Hedetniemi, Hedetniemi and Rall introduced the concept of k-dependent domination [40]. A *k-dependent dominating set* is a set which is both k-dependent and dominating. The *k-dependent domination number* $\gamma_{[k]}(G)$ equals the minimum size of a k-dependent dominating set and the *upper k-dependent domination number* $\Gamma_{[k]}(G)$ equals the maximum size of a minimal k-dependent dominating set in G.

In the same paper [40], the authors introduce the concept of k-dependent irredundance. The *k-dependent irredundance number* $ir_{[k]}(G)$ equals the minimum order of a maximal k-dependent irredundant set in G, while the *upper k-dependent irredundance number* $IR_{[k]}(G)$ equals the maximum size of a k-dependent irredundant set. The natural relationship between k-dependent dominating sets and k-dependent irredundant sets follows from the following lemma in [40]:

Proposition 1. *Every minimal k-dependent dominating set in a graph G is also a maximal k-dependent irredundant set in G.*

Corollary 1. *For any graph G,*

$$ir_{[k]}(G) \leq \gamma_{[k]}(G) \leq \Gamma_{[k]}(G) \leq IR_{[k]}(G).$$

In [40], the authors establish the NP-completeness of the decision problems associated with each of the invariants $ir_{[k]}(G)$, $\gamma_{[k]}(G)$, $\Gamma_{[k]}(G)$ and $IR_{[k]}(G)$. But no algorithms for computing any of these values were presented. It is a simple matter to construct an algorithm for computing the value of $ir_{[1]}(T)$ and $\gamma_{[1]}(T)$ for any tree T, given algorithms for computing $ir(T)$ and $\gamma(T)$, which already exist. However, it is more of a challenge to compute the upper values.

Algorithm 20. *Design an algorithm for computing the value of $\Gamma_{[1]}(T)$ for any tree T.*

Algorithm 21. *Design an algorithm for computing the value of* $IR_{[1]}(T)$ *for any tree* T.

It is worth noting that in [80], Lawler and Slater present a linear algorithm for finding the minimum number of K_1 and K_2 components which dominate the vertices of a tree.

4. Perfect and Nearly Perfect Sets

A number of graph theoretical invariants are defined in terms of a set $S \subset V$ of vertices and conditions under which a given vertex satisfies the *perfect* equality $|N[v] \cap S| = 1$, or the *nearly perfect* inequality $|N[v] \cap S| \leq 1$.

The concept of nearly perfect sets was first defined in a paper by Cockayne, Hartnell, Hedetniemi and Laskar in 1993 [22], but was first studied by Dunbar, Harris, Hedetniemi, Hedetniemi, McRae and Laskar in 1995 [33].

A set $S \subset V$ is called *nearly perfect* if for every vertex $v \in V - S$, $|N(v) \cap S| \leq 1$. A nearly perfect set S is 1*-minimal* if for every vertex $u \in S$, the set $S - \{u\}$ is not a nearly perfect set, and is 1*-maximal* if for every vertex $u \in V - S$, the set $S \cup \{u\}$ is not a nearly perfect set. The 1*-maximal nearly perfect number* of a graph G, $n_p(G)$, equals the minimum size of a 1-maximal nearly perfect set in G. The 1*-minimal nearly perfect number* $N_p(G)$ equals the maximum size of a 1-minimal nearly perfect set in G.

In [33] a linear algorithm is presented for computing the value of $n_p(T)$ for any tree T. It is also shown that the corresponding decision problem is NP-complete when restricted to either bipartite or chordal graphs. Somewhat surprisingly, it is shown in [33] that the value of $N_p(G)$ can be computed in polynomial time for any graph G.

However, the related concept of a total nearly perfect set was not discussed. As defined in [22], a set S is a *total nearly perfect set* if for every vertex $v \in V$, $|N(u) \cap S| \leq 1$. In the same way that $n_p(G)$ and $N_p(G)$ are defined, one can define $n_{tp}(G)$ and $N_{tp}(G)$.

NP-problem 17. *Does* G *have a* 1*-maximal nearly total perfect set of size at most* k*?*

Algorithm 22. *Design an algorithm to compute the value of* $n_{tp}(T)$ *for any tree* T.

NP-problem 18. *Does* G *have a* 1*-minimal nearly total perfect set of size at least* k*?*

Algorithm 23. *Design an algorithm to compute the value of* $N_{tp}(T)$ *for any tree* T.

5. Types of Domination

The two books on domination in graphs by Haynes, Hedetniemi and Slater [63] and [64] present many different types of dominating and related sets. Since the publication of these two books, perhaps another 20-30 different types of dominating sets have appeared in the literature. Not meaning to be totally comprehensive, we present in the next several subsections a variety of open domination problems for trees.

The most studied types of domination are (i) dominating sets, (ii) independent dominating sets and (iii) total dominating sets.

The first algorithm for computing the value of $\gamma(T)$ for any tree T was presented in 1975 by Cockayne, Goodman and Hedetniemi [19], while the NP-completeness of the associated dominating set problem was first given by Garey and Johnson in [49] (see also [74]).

The first algorithm for computing the value of $\beta_0(T)$ for any tree T was presented in 1966 by Daykin and Ng [29], while the first algorithm for computing the value of $i(T)$ was given by Beyer, Proskurowski, Hedetniemi and Mitchell in 1977 [7].

It was pointed out by Cockayne, Favaron, Payan and Thomason in 1981 [15], that for any bipartite graph, $\beta_0(G) = \Gamma(G)$, and thus the β_0-algorithm for trees suffices to compute the upper domination number of a tree.

A set $S \subset V$ is called a *total dominating set* if for every vertex $v \in V$, $N[v] \cap S \neq \emptyset$. Thus, every vertex in $V - S$ is adjacent to at least one vertex in S and every vertex in S is adjacent to at least one vertex in S other than itself. The *total domination number* $\gamma_t(G)$ equals the minimum size of a total dominating set in G, while the *upper total domination number* $\Gamma_t(G)$ equals the maximum size of a minimal total dominating set in G. First introduced by Cockayne, Dawes and Hedetniemi in 1980 [13], the algorithmic complexity of total domination was first considered by Laskar, Pfaff, Hedetniemi and Hedetniemi in 1984 [79]. They showed that the decision problem associated with the total domination number $\gamma_t(G)$ is NP-complete for undirected path graphs, and constructed the first linear algorithm for computing the value of $\gamma_t(T)$ for any tree T. A discussion of the NP-completeness results associated with $\gamma_t(G)$ can be found in [63]. The NP-completeness of the decision problem associated with the upper total domination number was settled by Jacobson and Peters in 1990 [77] for bipartite graphs, and later by McRae [85] for line graphs and line graphs of bipartite graphs. The first algorithm for computing the upper total domination number of a tree was given by Fricke, Hare, Jacobs and Majumdar in 1990 [45].

5.1 Irredundant Domination

In 1990, Hedetniemi, Hedetniemi and Jacobs [69] introduced the concept of private domination, as follows. A set S is called a *private dominating set* if S is a dominating set and every vertex in S has an external private neighbor. The *private domination number* equals the maximum size of a (minimal) private dominating set, and is denoted $\Gamma_p(G)$.

It is well-known that every graph G has a γ-set that is also a private dominating set (cf. Bollobas and Cockayne [9]). Thus, the minimum size of a private dominating set always equals the domination number.

In [69] the authors show that the following problem is NP-complete.

PRIVATE DOMINATING SET
INSTANCE: A graph G and a positive integer k.
QUESTION: Does G have a minimal private dominating set of size at least k?

The authors also construct a linear algorithm for computing the value of $\Gamma_p(T)$ for any tree T.

The concept of private domination, however, can be placed in the broader context of irredundant domination, as follows.

As defined, a private dominating set is a dominating set which is also open irredundant. Since there are three other kinds of irredundance, it follows that there are three other kinds of irredundant domination.

First, notice that any minimal dominating set is automatically an irredundant set. Thus, if we define $\gamma_{ir}(G)$ to equal the minimum size of an irredundant dominating set, then it would follow that $\gamma(G) = \gamma_{ir}(G)$. Similarly, if $\Gamma_{ir}(G)$ equals the maximum size of a minimal irredundant dominating set, then $\Gamma(G) = \Gamma_{ir}(G)$.

In the interests of uniformity of notation, we could say that $\Gamma_p(G)$, the private domination number, is also the *open irredundant domination number* $\Gamma_{oir}(G)$, and the *lower open irredundant domination number*, $\gamma_{oir}(G)$, equals the minimum size of an open irredundant dominating set. But by the theorem of Bollobas and Cockayne [9], it follows that $\gamma_{oir}(G) = \gamma(G)$.

Next, we can define the *open-open irredundant domination number* $\Gamma_{ooir}(G)$ (NEW HERE) to equal the maximum size of a minimal open-open irredundant dominating set, and $\gamma_{ooir}(G)$ to equal the minimum size of an open-open irredundant dominating set. Again, since every open irredundant set is also an open-open irredundant set, it follows that

$$\gamma(G) = \gamma_{oir}(G) = \gamma_{ooir}(G).$$

Finally, we can define the *closed-open irredundant domination number*, $\Gamma_{coir}(G)$ (NEW HERE), to equal the maximum size of a minimal closed-open irredundant dominating set, and the *lower closed-open irredundant domination number*, $\gamma_{coir}(G)$, to equal the minimum size of a closed-open irredundant dominating set. By much the same reasoning, however, we can conclude that since every γ-set is automatically a closed-open irredundant dominating set, $\gamma(G) = \gamma_{coir}(G)$.

NP-problem 19. *Does G have a minimal open-open irredundant dominating set of size at least k?*

Algorithm 24. *Design an algorithm for computing the value of $\Gamma_{ooir}(T)$, for any tree T.*

NP-problem 20. *Does G have a minimal closed-open irredundant dominating set of size at least k?*

Algorithm 25. *Design an algorithm for computing the value of $\Gamma_{coir}(T)$ for any tree T.*

5.2 Paired Domination

A dominating set $S \subset V$ is said to be a *paired dominating set* if the subgraph $G[S]$ induced by S has a perfect matching. The *paired domination number* $\gamma_{pr}(G)$ equals the minimum size of a paired dominating set, while the *upper paired domination number* $\Gamma_{pr}(G)$ (NEW HERE) equals the maximum size of a minimal paired dominating set.

Paired domination was introduced by Haynes and Slater [66], and studied further in Haynes, Henning and Slater [65] and Profitt, Haynes, and Slater [90], who settled the corresponding NP-completeness question [66]. Goddard [51] has recently constructed a simple algorithm for computing $\gamma_{pr}(T)$ for any tree T.

However, none of these papers discusses the upper paired domination number $\Gamma_{pr}(G)$ defined here.

NP-problem 21. *Does G have a minimal paired dominating set of size at least k ?*

Algorithm 26. *Design an algorithm for computing the value of $\Gamma_{pr}(T)$ for any tree T.*

5.3 Efficient and Perfect Domination

A dominating set S is said to be *efficient* if for every vertex $v \in V$, $|N[v] \cap S| = 1$. Two basic facts about efficient dominating sets (see [63]) are well known in domination theory: (i) not every graph has an efficient dominating set, for example, the five-cycle does not have one, and (ii) if a graph G has an efficient dominating set, then every efficient dominating set in G has the same size, which equals the domination number $\gamma(G)$.

However, every graph does have a perfect dominating set. A dominating set S is said to be *perfect* if for every vertex $v \in V - S$, $|N[v] \cap S| = 1$. Perfect dominating sets were introduced by Weichsel [95] and studied by Yen and Lee [97] and [98] and by Cockayne, Hartnell, Hedetniemi and Laskar in 1993 [22].

In [97], Yen and Lee show that the problem of deciding if a graph G has a perfect dominating set of size at most k is NP-complete for bipartite and chordal graphs. They also construct a linear algorithm for computing the minimum weight of a perfect dominating set in an arbitrary, weighted tree T.

Define the *perfect domination number*, $\gamma_p(G)$, to equal the minimum size of a perfect dominating set in G, and $\Gamma_p(G)$ (NEW HERE) to equal the maximum size of a minimal perfect dominating set in G.

NP-problem 22. *Does G have a minimal perfect dominating set of size at least k ?*

Algorithm 27. *Design an algorithm for computing the value of $\Gamma_p(T)$ for any tree T.*

5.4 Restrained Domination

In 1994, Telle [94] first defined the concept of a restrained dominating set. This was later studied by Domke, Hattingh, Hedetniemi, Laskar and Markus in 1994 [30]. A set $S \subset V$ is a *restrained dominating set* if every vertex in $V - S$ is adjacent to a vertex in S and a vertex in $V - S$ other than itself. The *restrained domination number* $\gamma_r(G)$ equals the minimum size of a restrained dominating set in G, while the *upper restrained domination number* $\Gamma_r(G)$ (NEW HERE) equals the maximum size of a minimal restrained dominating set in G. In [30] the authors showed that the decision problem associated with the restrained domination number $\gamma_r(G)$ is NP-complete for bipartite and chordal graphs. They also proposed a linear algorithm for computing the value of $\gamma_r(T)$ for any tree T. However, the algorithm given in [30] is incorrect. A refined and corrected version of this algorithm has recently been constructed by Goddard [51].

NP-problem 23. *Does G have a minimal restrained dominating set of size at least k?*

Algorithm 28. *Design an algorithm for computing the value of $\Gamma_r(T)$ for any tree T.*

5.5 Signed Domination

In 1995, Dunbar, Hedetniemi, Henning and Slater [35] introduced the concept of signed domination in graphs. A *signed dominating function* is a function of the form $f : V \to \{-1, 1\}$ such that for all vertices $v \in V$, $f[v] \geq 1$. By $f[v]$ we mean the sum of the values of $f(u)$ over all vertices $u \in N[v]$. The *weight* of a signed dominating function is simply the sum $\sum_{v \in V} f(v)$. The *signed domination number* $\gamma_{-11}(G)$ equals the minimum weight of a signed dominating function in G. The *upper signed domination number* $\Gamma_{-11}(G)$ equals the maximum weight of a minimal signed dominating function in G.

The algorithmic complexity of signed domination in graphs is discussed by Hattingh, Henning and Slater in 1995 [61]. They showed that the problem of deciding if a graph G has a signed dominating function of weight at most k is NP-complete for bipartite and chordal graphs, and the similar problem of deciding if a graph has a minimal signed dominating function of weight at least k is NP-complete for bipartite and chordal graphs. They also constructed a linear algorithm for computing the value of $\gamma_{-11}(T)$ for any tree T.

Algorithm 29. *Design an algorithm for computing the value of $\Gamma_{-11}(T)$ for any tree T.*

5.6 Minus Domination

In 1996, Dunbar, Hedetniemi, Henning and McRae [34] introduced the concept of minus domination in graphs (this work was delayed in publication until 1999). A *minus dominating function* is a function of the form $f : V \to \{-1, 0, 1\}$ such that for all vertices $v \in V$, $f[v] \geq 1$. The *weight* of a minus dominating function is simply the

sum $\sum_{v \in V} f(v)$. The *minus domination number* $\gamma_{-101}(G)$ equals the minimum weight of a minus dominating function in G. The *upper minus domination number* $\Gamma_{-101}(G)$ equals the maximum weight of a minimal minus dominating function in G.

In [32], Dunbar, Goddard, Hedetniemi, McRae and Henning showed that decision problems associated with $\gamma_{-101}(G)$ and $\Gamma_{-101}(G)$ are NP-complete for bipartite and chordal graphs. They also presented a linear algorithm for computing the value of $\gamma_{-101}(T)$ for any tree T. However, they did not present an algorithm for the upper minus domination number.

Algorithm 30. *Design an algorithm for computing the value of* $\Gamma_{-101}(T)$ *for any tree* T.

5.7 Secure Domination and Secure Total Domination

In 2003, Cockayne, Favaron and Mynhardt [14] defined a new type of domination, as follows. A set $S \subset V$ is a *secure dominating set* if for every vertex $v \in V - S$, there exists a vertex $u \in N(v) \cap S$ such that $S - \{u\} \cup \{v\}$ is a dominating set. A secure dominating set has the property that for every vertex $v \in V - S$, there is a neighbor of v in S, say u, such that the new set obtained by deleting u from S and adding v is another dominating set. This change of sets corresponds to moving a guard stationed at vertex $u \in S$ to the vertex $v \in V - S$, so that the resulting placement of guards is still a dominating set. The *secure domination number* $\gamma_s(G)$ equals the minimum size of a secure dominating set in G.

NP-problem 24. *Does* G *have a secure dominating set of size at most* k *?*

Algorithm 31. *Design an algorithm for computing the value of* $\gamma_s(T)$ *for any tree* T.

In a very recent paper, Benecke, Cockayne and Mynhardt [5] defined a *secure total dominating set* to be a set $S \subset V$ which is (i) a total dominating set, and (ii) for every vertex $u \in V - S$, there exists a vertex $v \in S \cap N(u)$ such that $S - \{v\} \cup \{u\}$ is a total dominating set. The *secure total domination number* $\gamma_{st}(G)$ equals the minimum size of a secure total dominating set in G.

NP-problem 25. *Does* G *have a secure total dominating set of size at most* k *?*

Algorithm 32. *Design an algorithm for computing the value of* $\gamma_{st}(T)$ *for any tree* T.

5.8 Eternal Security

Recent papers by Burger, Cockayne, Grundlingh, Mynhardt, Winterbach and van Vuuren [10] and Goddard, Hedetniemi and Hedetniemi [52] define a new kind of domination, as follows. Consider placing a guard on each vertex of a dominating set S_0 of a graph G. If for every vertex $v \in V - S_0$ there exists an adjacent vertex $u \in S_0$,

for which the resulting set $S_1 = S_0 - \{u\} \cup \{v\}$ is a dominating set, then we say that the set S_0 is 1-*secure*. A set S_0 is *eternally* 1-*secure* if for any sequence $v_1, v_2, \ldots, v_k$ of vertices, there exists a corresponding sequence of guards at vertices $u_1, u_2, \ldots, u_k$, and a sequence of dominating sets $S_1, S_2, \ldots, S_k$, so that $u_i \in S_{i-1}$ and $S_i = S_{i-1} - \{u_i\} \cup \{v_i\}$ and $v_i \in N[u_i]$.

The *eternal* 1-*security number* $\sigma_1(G)$, (denoted $\gamma_\infty(G)$ in [10]) equals the minimum size of an eternally 1-secure dominating set. It is pointed out in [52] that for any bipartite graph (hence, for any tree), $\sigma_1(G) = \beta_0(G)$.

A more general type of security is defined in [52] in which one permits the movement of any number of guards from vertices in a current dominating set to adjacent vertices in order to form a new dominating set. The *eternal* m-*security number*, $\sigma_m(G)$, is defined to equal the minimum size of a dominating set which can handle an arbitrary sequence of single attacks using multiple guard shifts.

In [52], the complexities of the decision problems associated with the invariants $\sigma_1(G)$ and $\sigma_m(G)$ are left as open problems. Indeed, the problems of deciding if a given dominating set is either an eternally 1-secure dominating set or an eternally m-secure dominating set are left as open problems.

Algorithm 33. *Design an algorithm for deciding if a given set* $S \subset V$ *is eternally* 1*-secure.*

Algorithm 34. *Design an algorithm for deciding if a given set* $S \subset V$ *is eternally* m*-secure.*

NP-problem 26. *Does* G *have an eternally* 1*-secure dominating set of size at most* k*?*

NP-problem 27. *Does* G *have an eternally* m*-secure dominating set of size at most* k*?*

Algorithm 35. *Design an algorithm for computing the value of* $\sigma_m(T)$ *for any tree* T*.*

5.9 Capacitated Domination

In a recent 2006 paper [55], Goddard, Hedetniemi, Huff and McRae consider capacitated domination. An r-*capacitated dominating set* (r CDS) of a graph G is a set $S = \{v_1, v_2, \ldots, v_k\} \subseteq V$ for which there exists a partition $\Pi = \{V_1, V_2, \ldots, V_k\}$ of V satisfying the following conditions for every i, $1 \leq i \leq k$:

- $v_i \in V_i$,
- $|V_i| \leq r + 1$,
- v_i is adjacent to all other vertices in V_i.

The minimum size of an r CDS in G is called the *r-capacitated domination number* and is denoted $\kappa_r(G)$ (Note: the authors actually use the Hebrew letter "daleth"). The authors study the basic properties of capacitated dominating sets, determine exact values for various classes of graphs, and show that the problem of deciding if a graph G has an r-capacitated dominating set of size at most k is NP-hard. They also construct a linear algorithm for computing the value of $\kappa_r(T)$ for any tree T. They mention, but do not discuss, the maxi-minimal version of this problem. One can define the *upper r-capacitated domination number* $\kappa^r(G)$ to equal the maximum size of a minimal r-capacitated dominating set.

NP-problem 28. *Does G have a minimal r-capacitated dominating set of size at least k?*

Algorithm 36. *Design an algorithm for computing the value of $\kappa^r(T)$ for any tree T.*

6. Broadcast Numbers

Recently Erwin [36] and [37], and Dunbar, Erwin, Haynes, Hedetniemi and Hedetniemi [31] introduced the concepts of broadcasts in graphs.

We say that a function $f : V \to \{0, 1, \ldots, diam(G)\}$ is a *broadcast* if for every vertex $v \in V$, $f(v) \leq e(v)$, where $diam(G)$ denotes the *diameter* of G and $e(v)$ denotes the *eccentricity* of vertex v. Also, let $rad(G)$ denote the *radius* of G.

Given a broadcast f, we define $N_f[u] = \{v \mid d(u, v) \leq f(u)\}$ to be the *broadcast neighborhood* of u. Further, we define $V^0 = \{v \mid f(v) = 0\}$ and $V^+ = V - V^0 = \{u \mid f(u) > 0\}$ (if there is some potential for ambiguity, then we shall let $V^+ = V_f^+$, $V^0 = V_f^0$, and so on). We say that every vertex in V^+ is a *broadcast vertex*, and the *broadcast neighborhood* $N_f[V^+] = \bigcup_{v \in V^+} N_f[v]$. If $u \in V^+$ is a broadcast vertex, $v \in V$ and $d(u, v) \leq f(u)$, then vertex v can *hear* a broadcast from vertex u. The set of vertices that a vertex $v \in V$ can hear is defined as $H(v) = \{u \in V^+ \mid d(u, v) \leq f(u)\}$. For a vertex $v \in V^+$, the *private f-neighborhood* $pn_f[v]$ is the set $\{u \in V \mid H(u) = \{v\}\}$. If $v \in pn_f[v]$, then we say that v is its own private f-neighbor. We define the *cost* of a broadcast to be $f(V) = \Sigma_{v \in V} f(v) = \Sigma_{v \in V^+} f(v)$.

A broadcast f of some type is said to be *minimal* (respectively, *maximal*) if there does not exist a broadcast $g \neq f$ of the same type such that for every $u \in V$, $g(u) \leq f(u)$ (respectively, $g(u) \geq f(u)$). Given two distinct broadcasts f and g, we say that $f \leq g$ (respectively, $f \geq g$) if and only if for every vertex $u \in V$, $f(u) \leq g(u)$ (respectively, $f(u) \geq g(u)$), for all $u \in V$.

Let $f_S : V \to \{0, 1\}$ be the characteristic function of a set $S \subseteq V$ of a graph G, that is, $f_S(u) = 1$ if $u \in S$, and $f_S(u) = 0$ otherwise. We will be interested in the characteristic functions of a variety of sets of vertices in a graph.

With this terminology we can define a number of different kinds of broadcasts.

6.1 Dominating broadcasts

As defined in [37], we say that a broadcast f is *dominating* if $N_f[V^+] = V(G)$, or equivalently, if for every $v \in V$, $|H(v)| \geq 1$. The minimum cost $f(V)$ of a dominating broadcast f of a graph G is the *broadcast domination number* $\gamma_b(G)$. We say that a dominating broadcast f of cost equal to $\gamma_b(G)$ is a γ_b*-broadcast*. The *upper broadcast domination number* equals the maximum cost of a minimal dominating broadcast, and is denoted $\Gamma_b(G)$.

In a very recent tour de force, Heggernes and Lokshtanov [75] designed a polynomial algorithm for computing the value of $\gamma_b(G)$ for any graph G. At present no algorithm as been designed for the upper broadcast domination number.

NP-problem 29. *Does G have a minimal dominating broadcast of size at least k ?*

Algorithm 37. *Design an algorithm for computing the value of $\Gamma_b(T)$ for any tree T.*

6.2 Independent broadcasts

A broadcast f is called *independent* if for every vertex $v \in V^+$, $N_f[v] \cap V^+ = \{v\}$, or equivalently, $|H(v)| = 1$. If f is an independent broadcast, then no broadcast vertex can hear a broadcast from any other broadcast vertex. We note that an independent broadcast need not be a dominating broadcast. The maximum cost of an independent broadcast of G is called the *broadcast independence number* and is denoted $\beta_b(G)$. The *lower broadcast independence number* $i_b(G)$ equals the minimum cost of a maximal independent broadcast of G.

The following inequalities illustrate the relationships between all four broadcasting invariants.

$$\begin{array}{ccccccc} \gamma(G) & \leq & i(G) & \leq & \beta_0(G) & \leq & \Gamma(G) \\ \bigvee | & & \diamond & & | \bigwedge & & | \bigwedge \\ \gamma_b(G) & \leq & i_b(G) & \leq & \beta_b(G) & \diamond & \Gamma_b(G) \end{array}$$

NP-problem 30. *Does G have a maximal independent broadcast of size at most k ?*

Algorithm 38. *Design an algorithm for computing the value of $i_b(T)$ for any tree T.*

NP-problem 31. *Does G have an independent broadcast of size at least k ?*

Algorithm 39. *Design an algorithm for computing the value of $\beta_b(T)$ for any tree T.*

6.3 Independent dominating broadcasts

A broadcast f is called *independent dominating* if it is both independent and dominating. The maximum cost of a minimal independent dominating broadcast of G

is called the *upper broadcast independent domination number* and is denoted $\Gamma_{ib}(G)$. Similarly, the *broadcast independent domination number* $\gamma_{ib}(G)$ equals the minimum cost of an independent dominating broadcast of G. Clearly, $\gamma_{ib}(G) \leq i(G)$ and $\Gamma_{ib}(G) \geq \beta_0(G)$, since the characteristic function f_S of any maximal independent set S is a minimal, independent dominating broadcast. Note that if f is a minimal independent dominating broadcast, then for any broadcast $g \neq f$ for which $g \leq f$, we know that g is independent but is not a dominating broadcast.

In domination theory it is well known that a strict inequality $\gamma(G) < i(G)$ often holds. However, for broadcasts we get a different result.

Theorem 5. [37] *If f is a broadcast on a graph G that is dominating but not independent, then there is a broadcast g on G that is dominating, independent, with $g(V) \leq f(V)$, and $V_g^+ \subset V_f^+$.*

Corollary 2. [37] *Every graph G has a γ_b-broadcast which is independent, that is, for any graph G,*

$$\gamma_b(G) = \gamma_{ib}(G).$$

Proposition 2. *For any graph G,*

$$\gamma_{ib}(G) \leq i_b(G) \leq rad(G) \leq diam(G) \leq \Gamma_{ib}(G).$$

NP-problem 32. *Does G have a minimal independent dominating broadcast set of size at least k?*

Algorithm 40. *Design an algorithm for computing the value of $\Gamma_{ib}(T)$ for any tree T.*

6.4 Efficient broadcasts

A broadcast f is called *efficient* if every vertex hears exactly one broadcast, that is, for every vertex $v \in V$, $|H(v)| = 1$. The maximum cost of an efficient broadcast is the *upper broadcast efficiency number* $\Gamma_{eb}(G)$, and the *broadcast efficiency number* $\gamma_{eb}(G)$ equals the minimum cost of an efficient broadcast.

For example, Figure 1 illustrates twelve distinct efficient broadcasts in the path P_7, where $\gamma_{eb}(P_7) = 3$ and $\Gamma_{eb}(P_7) = 6$.

Theorem 6. *Every graph G has a $\gamma_b(G)$-broadcast which is efficient.*

By definition, $\gamma_b(G) \leq \gamma_{eb}(G)$, so Theorem 6 implies that $\gamma_b(G) = \gamma_{eb}(G)$. We know that $\Gamma_{eb}(G) \leq min\{\beta_b(G), \Gamma_b(G), \Gamma_{ib}(G)\}$ for any graph G, because every efficient dominating broadcast is independent and minimal dominating. Also, observe that any diameter broadcast is an efficient dominating broadcast. Therefore we have the following corollary.

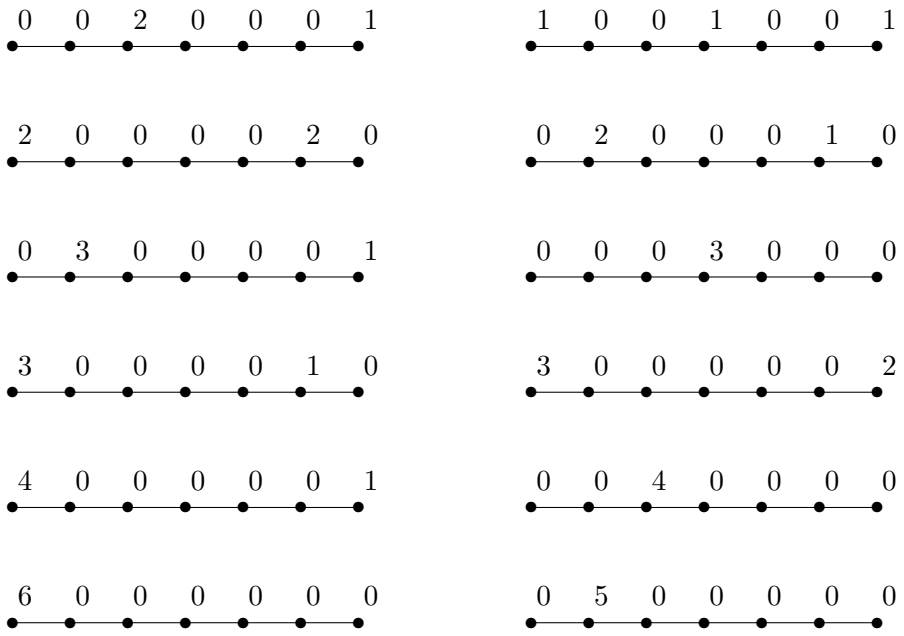

Figure 1: Efficient broadcasts in P_7.

Corollary 3. *For every graph G,*

$$\gamma_b(G) = \gamma_{ib}(G) = \gamma_{eb}(G) \leq i_b(G) \leq rad(G) \leq diam(G) \leq \Gamma_{eb}(G) \leq min\{\beta_b(G), \Gamma_b(G), \Gamma_{ib}(G)\}.$$

NP-problem 33. *Does G have a minimal efficient dominating broadcast of size at least k ?*

Algorithm 41. *Design an algorithm for computing the value of $\Gamma_{eb}(T)$ for any tree T.*

6.5 Packing broadcasts

A broadcast f is called a *packing* if every vertex hears at most one broadcast, that is, for every vertex $v \in V$, $|H(v)| \leq 1$. The maximum cost of a packing broadcast of G is called the *broadcast packing number* and is denoted $P_b(G)$. Similarly, the *lower broadcast packing number* equals the minimum cost of a maximal packing broadcast, and is denoted $p_b(G)$.

Note first that the characteristic function f_S of a maximal packing need not be a maximal packing broadcast. This can be seen by considering the path P_5 with values assigned as indicated in Figure 2(a). This center vertex in Figure 2(a) is a maximal packing, but it is not a maximal packing broadcast, since the value of 1 can be increased to 2, which results in the maximal packing broadcast (see Figure 2(b)).

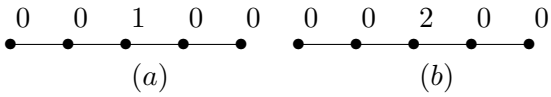

Figure 2: A maximal packing which is not a maximal packing broadcast.

Note that one may have a maximal packing broadcast which is not a dominating broadcast. For example, consider the values 1,0,0,1,0,0 in the graph of P_6. We know that for any graph G,

$$p(G) \leq P(G) \leq \gamma(G).$$

Proposition 3. *Every efficient broadcast is*

(*i*) *a maximal packing broadcast,*
(*ii*) *a minimal dominating broadcast, and*
(*iii*) *a minimal independent dominating broadcast.*

Corollary 4. *For any graph* G,

$p_b(G) \leq \gamma_{eb}(G) = \gamma_{ib}(G) = \gamma_b(G) \leq i_b(G) \leq \Gamma_{eb}(G) \leq min\{P_b(G), \Gamma_b(G), \Gamma_{ib}(G), \beta_b(G)\}.$

NP-problem 34. *Does* G *have a maximal packing broadcast of size at most* k *?*

Algorithm 42. *Design an algorithm for computing the value of* $p_b(T)$ *for any tree* T.

NP-problem 35. *Does* G *have a packing broadcast of size at least* k *?*

Algorithm 43. *Design an algorithm for computing the value of* $P^b(T)$ *for any tree* T.

7. Matchings

A set $M \subseteq E$ of edges is called *independent* if no two edges in M have a vertex in common. A *matching* in a graph G is any independent set M of edges. The *matching number* $\beta^1(G)$ equals the maximum size of a matching in G. The *minimum maximal matching number* $\beta_1(G)$ equals the minimum size of a maximal matching in G. We will also refer to $\beta_1(G)$ as the *lower matching number.*

Perhaps the first algorithm for computing the value of $\beta^1(T)$ for a tree was that of Mitchell [86], which appeared earlier in a 1975 paper by Mitchell, Hedetniemi and Goodman [88]. A $\beta^1(T)$-algorithm was also constructed independently by Savage in 1980 [91]. The first algorithm for computing the value of $\beta_1(T)$ was by Mitchell and Hedetniemi in 1977 [87]. It this paper it was also pointed out for the first time that for any tree T, $\gamma'(T) = \beta_1(T)$, that is, the edge domination number of any tree can always be achieved with a maximal matching of minimum cardinality.

Given a matching M we say that every vertex that is incident to an edge in M is *saturated* by M, and every edge in $E-M$ both of whose vertices are saturated by M is called a *saturated edge*. Let $V(M)$ denoted the set of vertices saturated by M, and let $E(M)$ be the set of edges saturated by M. The subgraph $G[M] = (V(M), E(M) \cup M)$ is the subgraph induced by M.

We say that a graph G has a *perfect matching* if it has a matching M for which $V[M] = V(G)$.

A matching M is called *induced* (or sometimes *strong*) if no two edges in M are joined by an edge. Equivalently, a matching is induced if it does not saturate any edges. Thus, in this case, $G[M] = (V(M), M)$. Induced matchings were introduced by Cameron [11].

The *induced matching number* $\beta^*(G)$ of a graph G equals the maximum size of an induced matching in G, while the *lower induced matching number* $\beta_*(G)$ equals the minimum size of a maximal induced matching in G. Algorithms for computing the value of $\beta^*(T)$ for any tree have been constructed by Cameron [11](chordal graphs), Fricke and Laskar [48] and Zito [99] and [100]. An algorithm for computing the value of $\beta_*(T)$ was recently constructed by Goddard, Hedetniemi, Hedetniemi and Laskar [54].

The next several sections will discuss open problems related to a variety of different kinds of matchings.

7.1 Uniquely Restricted Matchings

A matching M is said to be *uniquely restricted* if there is no other matching of the same size on the vertices saturated by M. Equivalently, a matching is uniquely restricted if and only if the subgraph $G[M]$ contains no (alternating) cycles. Thus, in a tree, every matching is automatically uniquely restricted.

The *uniquely restricted matching number* $\beta^{ur}(G)$ equals the maximum size of a uniquely restricted matching in G, while the *lower uniquely restricted matching number* $\beta_{ur}(G)$ (NEW HERE) equals the minimum size of a maximal uniquely restricted matching in G.

Uniquely restricted matchings were introduced by Golumbic, Hirst and Lewenstein [57], and have been studied by Levit and Mandrescu [81], [82] and [83].

Notice that every induced matching is a uniquely restricted matching. Therefore:

$$\beta_{ur}(G) \leq \beta_*(G) \leq \beta^*(G) \leq \beta^{ur}(G).$$

UNIQUELY RESTRICTED MATCHING
INSTANCE: graph G and a positive integer k.
QUESTION: Does G have a uniquely restricted matching of size at least k?

Golumbic, Hirst and Lewenstein [57] show that UNIQUELY RESTRICTED MATCHING is NP-complete for bipartite and split graphs, but can be solved in polynomial time for proper interval graphs, trees, cacti and threshold graphs. They leave as unsolved the complexity of UNIQUELY RESTRICTED MATCHING for interval graphs.

It is interesting to note that one may have to spend some time, given an arbitrary matching M in a given class of graphs, deciding whether M is a uniquely restricted matching. The authors show that this question can usually be answered in time $O(|V| + |E|)$.

Apparently the authors did not consider the mini-maximal version of uniquely restricted matchings. Trivially, the lower uniquely restricted matching number of any tree equals the lower matching matching number of a tree, and therefore an algorithm exists for this problem [87]. However the NP-completeness question has not been settled.

NP-problem 36. *Does G have a maximal uniquely restricted matching of size at most k ?*

7.2 Connected, Isolatefree, Acyclic and Disconnected Matchings

A recent paper by Goddard, Hedetniemi, Hedetniemi and Laskar [54] introduces four kinds of matchings

A matching M is said to be *connected* if the induced subgraph $G[M]$ is connected. The *connected matching number* $\beta^c(G)$ equals the maximum size of a connected matching in G. The *lower connected matching number* is denoted $\beta_c(G)$.

A matching is said to be *isolate free* if either $|M| = 1$ or $G[M]$ has no K_2 component. The corresponding invariants are denoted $\beta^{if}(G)$ and $\beta_{if}(G)$.

A matching M is said to be *acyclic* if the induced subgraph $G[M]$ contains no cycles. The *acyclic matching number* $\beta^a(G)$ equals the maximum size of an acyclic matching in G, while the *lower acyclic matching number* is denoted $\beta_a(G)$

Finally a matching is called *disconnected* if $G[M]$ is a disconnected graph. The corresponding two invariants are denoted $\beta^{dc}(G)$ and $\beta_{dc}(G)$.

The following results are given in [54].

1. $\beta^*(G) \leq \beta^a(G) \leq \beta^{ur}(G) \leq \beta^1(G)$.
2. $\beta^*(G) \leq \beta^{dc}(G) \leq \beta^1(G)$.
3. $\beta^c(G) \leq \beta^{if}(G) \leq \beta^1(G)$.
4. For every connected graph G, $\beta^c(G) = \beta^{if}(G) = \beta^1(G)$.
5. For any tree T, $\beta^1(T) - 1 \leq \beta^{dc}(T) \leq \beta^1(T)$.
6. For any connected graph G, $\beta_1(G) \leq \beta_{if}(G) \leq \beta_c(G)$.

It is also pointed out in [54] that the connected and isolate free matching numbers can be computed in polynomial time, while the ACYCLIC MATCHING Problem is NP-hard. In

addition it is mentioned that LOWER ACYCLIC MATCHING, LOWER CONNECTED MATCHING AND LOWER DISCONNECTED MATCHING are all NP-hard.

An interesting open problem is that of determining the complexity of the following problem.

NP-problem 37. *DISCONNECTED MATCHING*
INSTANCE: A graph G and a positive integer k.
QUESTION: Does G have a disconnected matching of size at least k?

Algorithm 44. *Design an algorithm to compute the value of $\beta_{if}(T)$ for any tree T.*

Algorithm 45. *Design an algorithm to compute the value of $\beta_c(T)$ for any tree T.*

Algorithm 46. *Design an algorithm to compute the value of $\beta_{dc}(T)$ for any tree T.*

Algorithm 47. *Design an algorithm to compute the value of $\beta^{dc}(T)$ for any tree T.*

7.3 b-Matchings

For each vertex $v \in V$ specify a value $b(v)$, where $1 \leq b(v) \leq d(v)$, where $d(v)$ is the degree of v. In this way we define a vector called a *b-vector*. A *b-matching* is a set of edges M having the property that no vertex v is incident with more than $b(v)$ edges of M. The *b-matching number* $\beta^{(b)}(G)$ equals the maximum number of edges in a b-matching in G. The *lower b-matching number* $\beta_{(b)}(G)$ (NEW HERE) equals the minimum number of edges in a maximal b-matching in G.

One of the earliest papers on b-matchings in graphs was that of Goodman, Hedetniemi and Tarjan in 1976 [59]. This paper contains a linear time, greedy algorithm for computing the value of $\beta^{(b)}(T)$ for any tree T.

However, the authors did not define or consider the mini-maximal version of this problem.

NP-problem 38. *Does G have a maximal b-matching of size at most k?*

Algorithm 48. *Design an algorithm for computing the value of $\beta_{(b)}(T)$ for any tree T.*

8. Influence Numbers

In [93], Stege and Van Rooij introduced the concept of the influence number of a set of vertices and the influence number of a graph (see also [1]). Let $S \subset V$ be a set of vertices. For any vertex $u \in V$ the value $d(u, S)$ equals the minimum distance $d(u, v)$ between vertex u and a vertex $v \in S$.

The *influence* of S is defined to equal

$$\eta(S) = \sum_{u \in S} \frac{1}{2^{d(u,S)}}.$$

The *influence number* $\eta(G)$ of a graph G equals the maximum influence of a set S in G. In [27] Daugherty settled the NP-completeness of the decision problem associated with $\eta(G)$, while McRae [27] has constructed a linear algorithm for computing the value of $\eta(T)$ for any tree T.

Similarly, Hartnell, Hedetniemi, Hedetniemi and Rall (cf. [27]) defined the *total influence* of a set S to equal

$$\sum_{u \in S} \sum_{v \in V-S} \frac{1}{2^{d(u,v)}}.$$

The *total influence number* $\eta_t(G)$ equals the maximum total influence of a set $S \subset V$. Daugherty, Lyle and Laskar have recently published a paper on the total influence number of a graph [28]. However, the complexity of the decision problem associated with the total influence number has not been settled, nor has an algorithm been constructed for trees.

NP-problem 39. *Does G have a total influence set of size at least k ?*

Algorithm 49. *Design an algorithm for computing the value of $\eta_t(T)$ for any tree T.*

The mini-maximal problems for the influence and total influence numbers have not yet been defined and studied. We say that a set S is a *maximal (total) influence set* if no proper superset of S has greater (total) influence. While it is difficult to define a symbol for these two invariants, we will use a subscript of *mm*, for mini-maximal, to denote these two values. Let $\eta_{mm}(G)$ (NEW HERE) equal the mini-maximal influence number of G, and $\eta_{mmt}(G)$ (NEW HERE) equal the mini-maximal total influence number of G.

NP-problem 40. *Does G have a maximal influence set of size at most k ?*

Algorithm 50. *Design an algorithm for computing the value of $\eta_{mm}(T)$ for any tree T.*

NP-problem 41. *Does G have a maximal total influence set of size at most k ?*

Algorithm 51. *Design an algorithm for computing the value of $\eta_{mmt}(T)$ for any tree T.*

9. Vertex Partition and Coloring Problems

Several problems involved with partitioning the vertex set of a graph into sets, each of which has a given property $\mathcal{P}$, have recently appeared, whose solutions for trees have not been found. We present several examples.

9.1 Broadcast Chromatic Number

As introduced by Goddard, Hedetniemi, Hedetniemi, Harris and Rall [53], a function $\pi : V \to \{0, 1, \ldots, k\}$ is called a *broadcast coloring of order k* if $\pi(u) = \pi(v)$ implies that

$d(u,v) > \pi(u)$. The minimum order of a broadcast coloring of a graph G is called the *broadcast chromatic number*, and is denoted by $\chi_b(G)$. Equivalently, a broadcast coloring is a partition $\Pi = \{V_1, V_2, \ldots, V_k\}$ of V such that each color class V_i is an i*-packing*, that is, the distance between any two vertices in V_i is greater than i.

In [53] the authors show that one can determine in polynomial time if an arbitrary graph has a broadcast coloring of order 1, 2, or 3. However, they show that the problem of deciding if a graph has a broadcast coloring of order 4 is NP-hard. The computation of the value of $\chi_b(T)$ for any tree has proved to be quite difficult.

Algorithm 52. *Design an algorithm for computing the value of* $\chi_b(T)$ *for any tree* T.

We note that a paper similar to [53] has been written by Sloper [92], who uses a slightly different definition of a broadcast coloring. We will give these colorings a different name and call them *broadcast e-colorings* ("e" for eccentricity). A *broadcast e-coloring* of a graph G is a function $color : V \to \mathcal{N}$ such that (i) $color(u) = color(v)$ implies that $d(u,v) > color(u)$ and (ii) for every $v \in V$, $color(v) \leq e(v)$, where $e(v)$ is the eccentricity of v. According to this definition, many graphs will not have a broadcast e-coloring. In [92] Sloper presents several classes of trees which have broadcast e-colorings and several classes of trees which are not broadcast e-colorable. This raises an interesting problem:

NP-problem 42. *Given an arbitrary tree* T*, is it broadcast e-colorable?*

9.2 Dominator Partitions

In a very recent paper, Cockayne, Hedetniemi, Hedetniemi, Laskar, McRae and Wallis [23] introduced the concept of a dominator partition. We say that a vertex $v \in V$ is a *dominator* of a set $S \subset V$ if v dominates (is adjacent to) every vertex in S. A partition $\pi = \{V_1, V_2, \ldots, V_k\}$ is called a *dominator partition* if every vertex $v \in V$ is a dominator of at least one block V_i of π. A dominator partition is said to be *minimal* if any partition π' obtained from π by forming the union of any two classes $V_i \cup V_j$, $i \neq j$, is no longer a dominator partition. The *dominator partition number* $\pi_d(G)$ equals the minimum k such that G has a dominator partition of order k. The *upper dominator partition number* $\Pi_d(G)$ equals the maximum order of a minimal dominator partition of G. In something of a surprise, the authors prove in [23]:

Theorem 7. *For any graph* G, $\gamma(G) \leq \pi_d(G) \leq \gamma(G) + 1$.

They then show that the problem of deciding if a graph G has a dominator partition of order at most k is NP-complete, even for bipartite, planar or chordal graphs. In addition they show that the value of $\pi_d(T)$ can be computed in linear time for any tree T. Also in polynomial time, one can decide if $\pi_d(G) \leq 2$. However, they were not able to settle the complexity of the following interesting question:

NP-problem 43. *Does G have a dominator partition of order at most 3 ?*

Very recently, Goddard, Jacob and Laskar [56] presented several results on the upper dominator partition number. However, they have not solved the following two problems.

NP-problem 44. *Does G have a minimal dominator partition of order at least k ?*

Algorithm 53. *Design an algorithm for computing the value of $\Pi_d(T)$ for any tree T .*

9.3 Dominator Colorings

In a very recent paper Hedetniemi, Hedetniemi, McRae and Blair [71] introduce the concept of dominator colorings of graphs. A subsequent paper on dominator colorings by Gera, Rasmussen and Horton [50] was recently presented at the 37^{th} Southeastern International Conference on Graph Theory, Combinatorics and Computing. A *dominator coloring* of a graph G is a dominator partition $\pi = \{V_1, V_2, \ldots, V_k\}$ in which each class V_i is an independent set. Thus, dominator partitions are proper colorings, in which each vertex $v \in V$ is a dominator of at least one color class. As with dominator colorings, one can say that a dominator coloring π is *minimal* if any partition π' obtained from π by forming the union of any two classes $V_i \cup V_j$, $i \neq j$, is no longer a dominator coloring. We define the *dominator chromatic number* $\chi_d(G)$ to equal the minimum order of a dominator coloring of G , while the *dominator achromatic number* $\psi_d(G)$ equals the maximum order of a minimal dominator coloring of G .

Let $\chi(G)$ denote the well known chromatic number of a graph G .

Theorem 8. *[71] For any graph G , $max\{\gamma(G), \chi(G)\} \leq \chi_d(G) \leq \gamma(G) + \chi(G)$.*

It is also shown that the problem of deciding if a graph G has a dominator coloring of order at most k is NP-complete. The following is also proved:

Proposition 4. *[71] For any tree T , $\gamma(T) \leq \chi_d(T) \leq \gamma(T) + 1$.*

However the authors have not been able to construct an algorithm for trees, and have not settled the decision problem associated with $\psi_d(G)$.

Algorithm 54. *Design an algorithm for computing the value of $\chi_d(T)$ for any tree T .*

NP-problem 45. *Does G have a minimal dominator coloring of order at least k ?*

Algorithm 55. *Design an algorithm for computing the value of $\psi_d(T)$ for any tree T .*

10. Problems Involving Independent Sets

This concluding section is something of a postscript to most of the preceding sections. Most of the these problems ask you to find the minimum or maximum size of a set having

one or more specified properties. What if, in addition, you add the requirement (if it isn't already stipulated) that the set be independent? Immediately a number of problems take on added interest. We conclude by listing several of these that we think are of some interest.

Algorithm 56. *Design an algorithm for computing the minimum size of a* 1*-maximal independent nearly perfect set in a tree* T*.*

Algorithm 57. *Design algorithms for computing the minimum and maximum size of a maximal independent total irredundant set in a tree* T*.*

Algorithm 58. *Design algorithms for computing the minimum and maximum size of an independent open perfect neighborhood set in a tree* T*.*

Algorithm 59. *Design an algorithm for computing the minimum size of an independent* k*-capacitated dominating set in a tree* T*.*

Algorithm 60. *Design an algorithm for computing the maximum influence of an independent set in a tree* T*.*

Algorithm 61. *Design an algorithm for computing the maximum total influence of an independent set in a tree* T*.*

11. Summary

The tables below provide a quick summary of these many open problems.

NP	*Tree Algorithm*	*Lower*	*Number*	*Upper*	*Tree Algorithm*	NP
[89]		oir	*open irredundance*	OIR	[77]	[38]
[85]		$ooir$	*open-open irredundance*	$OOIR$	[47]	[11]
[85]		$coir$	*closed-open irredundance*	$COIR$	[58]	[42]
		$ioir$	*independent open irredundance*	$IOIR$		
		ir_t	*total irredundance*	IR_t	[70]	[70]
		oir_t	*total open irredundance*	OIR_t		
		$ooir_t$	*total open-open irredundance*	$OOIR_t$		
		$coir_t$	*total closed-open irredundance*	$COIR_t$		
		er	*external redundant*	ER		
[72]		θ_o	*open perfect neighborhood*	Θ_o		
		θ_i	*independent perfect neighborhood*	Θ_i		
		ra	*R-annihilated*	Ra		
		rai	*R-annihilated irredundance*	Rai		
	[60]	$i_{[1]}$	[1]*-dependence*	$\beta_{[1]}$	[60]	[60]
[40]	Simple	$\gamma_{[1]}$	[1]*-dependent domination*	$\Gamma_{[1]}$		[40]
[40]	Simple	$ir_{[1]}$	[1]*-dependent irredundance*	$IR_{[1]}$		[40]
		n_{tp}	*total nearly perfect*	N_{tp}		

NP	*Tree Algorithm*	*Lower*	*Number*	*Upper*	*Tree Algorithm*	*NP*
[49]	[19]	γ_{ooir}	*open open irredundant domination*	Γ_{ooir}		
[49]	[19]	γ_{coir}	*closed-open irredundant domination*	Γ_{coir}		
[66]	[51]	γ_{pr}	*paired domination*	Γ_{pr}		
[97]	[97]	γ_{p}	*perfect domination*	Γ_{p}		
[30]	[30]	γ_{r}	*restrained domination*	Γ_{r}		
[61]	[61]	γ_{-11}	*signed domination*	Γ_{-11}		[61]
[32]	[32]	γ_{-101}	*minus domination*	Γ_{-101}		[32]
		γ_{s}	*secure domination*	Γ_{s}		
		γ_{st}	*secure total domination*	Γ_{st}		
		σ_1	*eternal 1-security*	Σ_1		
		σ_m	*eternal m-security*	Σ_m		
[55]	[55]	κ_{r}	*r-capacitated domination*	κ^{r}		
[75]	[75]	γ_{b}	*broadcast domination*	Γ_{b}		
		i_{b}	*independent broadcast*	β_{b}		
[75]	[75]	γ_{ib}	*independent dominating broadcast*	Γ_{ib}		
[75]	[75]	γ_{eb}	*efficient dominating broadcast*	Γ_{eb}		
		p_{b}	*packing broadcast*	P_{b}		
	[87]	β_{ur}	*uniquely restricted matching*	β^{ur}	[57]	[57]
[54]		β_{if}	*isolate-free matching*	β^{if}	[54]	[54]
[54]		β_{c}	*connected matching*	β^{c}	[54]	[54]
[54]		β_{dc}	*disconnected matching*	β^{dc}		
		$\beta_{(b)}$	*(b)-matching*	$\beta^{(b)}$	[59]	
		η_{mm}	*influence*	η	[27]	[27]
		η_{mmt}	*total influence*	η_{t}		
[53]		χ_{b}	*broadcast chromatic*			
[23]	[23]	π_{d}	*dominator partition*	Π_{d}		
[71]		χ_{d}	*dominator chromatic*	ψ_{d}		
		n_{ip}	*independent nearly perfect*	N_{ip}		
		κ_{ik}	*independent k-capacitated domination*	κ^{ik}		
		η_{i}	*independent influence*	η^{i}		
		η_{it}	*independent total influence*	η^{it}		

Acknowledgments

The author gratefully acknowledges Professors Sandra Hedetniemi, Wayne Goddard, Renu Laskar, Brian Dean and David Jacobs at Clemson University, Professor Alice McRae at Appalachian State University, Professor Doug Rall at Furman University, Professor Jean Dunbar at Converse College, Professor E. J. Cockayne at the University of Victoria, Professor Johan Hattingh at Georgia State University, Professor Peter J. Slater at The University of Alabama in Huntsville, and Professor Teresa Haynes at East Tennessee State University, for their generous help in listing these open problems and providing appropriate bibliographic entries. Any omissions or incorrect citations, should they be present, are entirely the fault of the author, for which apologies are offered.

References

[1] P. Agah, A. Hertel, P. Hertel, A. Scott, U. Stege and I. van Rooij, Profit problems in graphs: Classical graph problems reconsidered, Manuscript, 2005.

[2] S. Arnborg, Efficient algorithms for combinatorial problems on graphs with bounded decomposability - a survey, *BIT,* **25** (1985),2-23.

[3] S. Arnborg, J. Lagergren and D. Seese, Easy problems for tree-decomposable graphs, *J. Algorithms,* **12**(1991), 308-340.

[4] S. Arnborg and A. Proskurowski, Linear time algorithms for NP-hard problems restricted to partial k-trees, *Discrete Appl. Math.,* **23** (1989),11-24.

[5] S. Benecke, E.J. Cockayne and C.M. Mynhardt, Secure total domination in graphs, submitted for publication.

[6] M.W. Bern, E.L. Lawler and A.L. Wong, Linear-time computation of optimal subgraphs of decomposable graphs, *J. Algorithms,* **8**(1987), 216-235.

[7] T. Beyer, A. Proskurowski, S. Hedetniemi and S. Mitchell, Independent domination in trees, *Proc. Eighth Southeastern Conf. on Combinatorics, Graph Theory and Computing*, Util. Math., Winnipeg, 321-328, 1977.

[8] H.L. Bodlaender, A tourist guide through treewidth, *Acta Cybernet.,* **11**(1993),1-23.

[9] B. Bollobas and E.J. Cockayne, Graph theoretic parameters concerning domination, independence and irredundance, *J. Graph Theory,* **3**(1979),241-249.

[10] A.P. Burger, E.J. Cockayne, W.R. Grundlingh, C.M. Mynhardt, W. Winterbach and J.H. van Vuuren, Infinite order domination in graphs, *J. Combin. Math. Combin. Comput.,* **50** (2004),179-194.

[11] K. Cameron. Induced matchings, *Discrete Appl. Math.,* **24**(1989),97-102.

[12] E.J. Cockayne, Generalized irredundance in graphs: hereditary properties and Ramsey numbers, *J. Combin. Math. Combin. Comput.,* **31**(1999), 15-31.

[13] E.J. Cockayne, R.M. Dawes and S.T. Hedetniemi, Total domination in graphs, *Networks,* **10**(1980), 211-219.

[14] E.J. Cockayne, O. Favaron and C.M. Mynhardt, Secure domination, weak Roman domination and forbidden subgraphs, *Bull. Inst. Combin. Appl.,* **39** (2003), 87-100.

[15] E.J. Cockayne, O. Favaron, C. Payan and A.G. Thomason, Contributions to the theory of domination, independence and irredundance in graphs, *Discrete Math.,* **33**(1981), 249-258.

[16] E.J. Cockayne, O. Favaron, J. Puech and C.M. Mynhardt, Packing, perfect neighbourhood, irredundant and R-annihilated sets in graphs, *Australas. J. Combin.,* **18**(1998), 253-262.

[17] E.J. Cockayne, O. Favaron, J. Puech and C.M. Mynhardt, An inequality chain of domination parameters for trees, *Discuss. Math. Graph Theory,* **18**(1)(1998), 127-142.

[18] E.J. Cockayne and S. Finbow, Generalised irredundance in graphs: Nordhaus-Gaddum bounds, *Discuss. Math. Graph Theory*, (To appear).

[19] E.J. Cockayne, S.E. Goodman and S.T. Hedetniemi, A linear algorithm for the domination number of a tree, *Inform. Process. Lett.,* **4**(1975),41-44.

[20] E.J. Cockayne, P.J.P. Grobler, S.T. Hedetniemi and A.A. McRae, What makes an irredundant set maximal? *J. Combin. Math. Combin. Comput.,* **25**(1997),213-223.

[21] E.J. Cockayne, J.H. Hattingh, S.M. Hedetniemi, S.T. Hedetniemi and A. A. McRae, Using maximality and minimality conditions to construct inequality chains, *Discrete Math.,* **176**(1997),43-61.

[22] E.J. Cockayne, B.L. Hartnell, S.T. Hedetniemi and R. Laskar, Perfect domination in graphs, *J. Combin. Inform. Sys. Sci.,* **18**(1-2)(1993), 136-148.

[23] E.J. Cockayne, S.M. Hedetniemi, S.T. Hedetniemi, R. Laskar and C.K. Wallis, Dominator partitions of graphs, Submitted for publication, 2006.

[24] E.J. Cockayne, S.M. Hedetniemi, S.T. Hedetniemi and C.M. Mynhardt, Irredundant and perfect neighbourhood sets in trees, *Discrete Math.,* **188**(1-3)(1998), 253-260.

[25] E.J. Cockayne, S.T. Hedetniemi and D.J. Miller, Properties of hereditary hypergraphs and middle graphs, *Canad. Math. Bull.,* **21**(1978),461-468 .

[26] E.J. Cockayne and C.M. Mynhardt, On a conjecture concerning irredundant and perfect neighbourhood sets in graphs, Papers in honour of Stephen T. Hedetniemi, *J. Combin. Math. Combin. Comput.,* **31**(1999),241-253.

[27] S.M. Daugherty, The influence and total influence numbers of a graph, MS paper, Dept. Mathematical Sciences, Clemson University, May 2005.

[28] S. Daugherty, J. Lyle and R. Laskar, On the total influence number of a graph, *Congr. Numer.,* **174**(2005),107-121.

[29] D.E. Daykin and C.P. Ng, Algorithms for generalized stability number of tree graphs, *J. Austral. Math. Soc.,* **6**(1966),89-100.

[30] G.S. Domke, J.H. Hattingh, S.T. Hedetniemi, R.C. Laskar and L.R. Markus, Restrained domination in graphs, *Discrete Math.,* **203**(1999),61-69.

[31] J.E. Dunbar, D.J. Erwin, T.W. Haynes, S.M. Hedetniemi and S.T. Hedetniemi, Broadcasts in graphs, *Discrete Appl. Math.,* **154**(1)(2006),59-75.

[32] J. Dunbar, W. Goddard, S. Hedetniemi, A. McRae and M.A. Henning, The algorithmic complexity of minus domination in graphs, *Discrete Appl. Math.,* **68**(1996), 73-84.

[33] J.E. Dunbar, F.C. Harris, S.M. Hedetniemi, S.T. Hedetniemi, A.A. McRae and R.C. Laskar, Nearly perfect sets in graphs, *Discrete Math.,* **138**(1995),229-246.

[34] J. Dunbar, S. Hedetniemi, M.A. Henning and A. McRae, Minus domination in graphs, *Discrete Math.,* **199**(1999), 35-47.

[35] J. Dunbar, S.T. Hedetniemi, M.A. Henning and P.J. Slater, Signed domination in graphs, In *Proc. Seventh Quadrennial Internat. Conf. on the Theory and Applications of Graphs*, Y. Alavi and A. Schwenk, Eds., *Graph Theory, Combinatorics and Algorithms* **1**(1995),311-322.

[36] D. Erwin, Cost domination in graphs, PhD Thesis, Western Michigan University, 2001.

[37] D. Erwin, Dominating broadcasts in graphs, *Bull. Inst. Combin. Appl.,* **42**(2004),89-105.

[38] S. Even, O. Goldreich, S. Moran and P. Tong, On the NP-completeness of certain network testing problems, *Networks,* **14**(1984),1-24.

[39] O. Favaron, T. Haynes, S.T. Hedetniemi, M.A. Henning and D. Knisley, Total irredundance in graphs, *Discrete Math.,* **256**(2002),115-127.

[40] O. Favaron, S.M. Hedetniemi, S.T. Hedetniemi and D.F. Rall, On k-dependent domination, *Discrete Math.,* **249**(2002),83-94.

[41] O. Favaron and J. Puech, Irredundant and perfect neighborhood sets in graphs and claw-free graphs, 16th British Combinatorial Conference (London, 1997), *Discrete Math.,* **197/198**(1999),269-284.

[42] M. Fellows, G. Fricke, S. Hedetniemi and D. Jacobs, The private neighbor cube, *SIAM J. Discrete Math.,* **7**(1994), 41-47.

[43] S. Finbow, Generalisations of irredundance in graphs, PhD Thesis, University of Victoria, 2003.

[44] J.F. Fink and M.S. Jacobson, n-domination in graphs, In *Graph Theory with Applications to Algorithms and Computer Science*, Y. Alavi and A.J. Schwenk, Eds., (Kalamazoo, MI 1984), Wiley, 283-300, 1985.

[45] G.H. Fricke, E.O. Hare, D.P. Jacobs and A. Majumdar. On integral and fractional total domination, *Congr. Numer.,* **77**(1990), 87-95.

[46] G.H. Fricke, T.W. Haynes, S.M. Hedetniemi, S.T. Hedetniemi and M.A. Henning, On perfect neighborhood sets in graphs, *Discrete Math.,* **199** (1999), 221-225.

[47] G. Fricke, S.M. Hedetniemi and R. Laskar, Open-irredundance in trees, Unpublished manuscript, 1989.

[48] G. Fricke and R. Laskar, Strong matchings in trees, *Congr. Numer.,* **89** (1992), 239-243.

[49] M.R. Garey and J.S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, Freeman, 1979.

[50] R. Gera, C. Rasmussen and S. Horton, Dominator colorings and safe clique partitions, Presented at 37^{th} *Southeastern Internat. Conf. on Graph Theory, Combinatorics and Computing*, (Florida Atlantic University, March 6-10, 2006).

[51] W. Goddard, Private communication, April 2006.

[52] W. Goddard, S.M. Hedetniemi and S.T. Hedetniemi, Eternal security in graphs, *J. Combin. Math. Combin. Comput.,* **52**(2005), 169-180.

[53] W. Goddard, S.M. Hedetniemi, S.T. Hedetniemi, J.M. Harris and D.F. Rall, Broadcast chromatic numbers of graphs, *Ars Combin.,* (To appear).

[54] W. Goddard, S.M. Hedetniemi, S.T. Hedetniemi and R. Laskar, Generalized subgraph-restricted matchings in graphs, *Discrete Math.,* **293** (2005), 129-138.

[55] W. Goddard, S.T. Hedetniemi, J.L. Huff and A.A. McRae, Capacitated domination, Submitted for publication, January 2006.

[56] W. Goddard, J. Jacob and R. Laskar, On the upper dominator partition problem of certain graphs. Presented at the *Thirty Seventh Southeastern Internat. Conf. on Combinatorics, Graph Theory and Computing*, (Florida Atlantic University, March 6-10, 2006).

[57] M.C. Golumbic, T. Hirst and M. Lewenstein, Uniquely restricted matchings, *Algorithmica,* **31** (2001), 139-154.

[58] M.C. Golumbic and R.C. Laskar, Irredundancy in circular arc graphs, *Discrete Appl. Math.,* **44** (1993), 79-89.

[59] S. Goodman, S. Hedetniemi and R.E. Tarjan, b-matchings in trees, *SIAM J. Comput.,* **5**(1) (1976), 104-108.

[60] E.O. Hare, S.M. Hedetniemi and S.T. Hedetniemi, 2-dependence numbers of trees, Unpublished manuscript, dated September 11, 1989.

[61] J.H. Hattingh, M.A. Henning and P.J. Slater, The algorithmic complexity of signed domination in graphs, *Australas. J. Combin.,* **12**(1995), 101-112.

[62] T.W. Haynes, S.T. Hedetniemi, M.A. Henning and D.J. Knisley, Stable and unstable graphs with total irredundance number zero, *Ars Combin.,* **61** (2001), 33-46.

[63] T.W. Haynes, S.T. Hedetniemi and P.J. Slater, *Fundamentals of Domination in Graphs*, Marcel Dekker, 1998, 446 pp.

[64] T.W. Haynes, S.T. Hedetniemi and P.J. Slater, Eds. *Domination in Graphs: Advanced Topics*, Marcel Dekker, 1998, 497 pp.

[65] T.W. Haynes, M.A. Henning and P.J. Slater, Trees with equal domination and paired-domination numbers, *Ars Combin.,***76** (2005),169-175.

[66] T.W. Haynes and P.J. Slater, Paired-domination in graphs, *Networks,* **32**(3)(1998), 199-206.

[67] T.W. Haynes and P.J. Slater, Paired-domination and the paired domatic number, *Congr. Numer.,* **109**(1995), 65-72.

[68] S.M. Hedetniemi, S.T. Hedetniemi and M.A. Henning, The algorithmic complexity of perfect neighborhoods in graphs, *J. Combin. Math. Combin. Comput.,* **25** (1997), 183-192.

[69] S.M. Hedetniemi, S.T. Hedetniemi and D.P. Jacobs, Private domination: theory and algorithms, *Congr. Numer.,* **79** (1990), 147-157.

[70] S.M. Hedetniemi, S.T. Hedetniemi and D.P. Jacobs, Total irredundance in graphs: theory and algorithms, *Ars Combin.,* **35A** (1993), 271-284.

[71] S.M. Hedetniemi, S.T. Hedetniemi, A.A. McRae and J.R.S. Blair, Dominator colorings of graphs, Preprint, 2006.

[72] S.T. Hedetniemi, D.P. Jacobs, R. Laskar and D. Pillone, Open perfect neighborhood sets in graphs, Unpublished manuscript, dated August 23, 1996.

[73] S. Hedetniemi, R. Laskar and J. Pfaff, Irredundance in graphs: a survey, *Congr. Numer.,* **48**(1985), 183-194.

[74] S.T. Hedetniemi, A.A. McRae and D.A. Parks, Complexity Results, Chapter 9 in *Domination in Graphs: Advanced Topics*, T.W. Haynes, S.T. Hedetniemi and P.J. Slater, Eds., Marcel Dekker, 233-269,1998.

[75] P. Heggernes and D. Lokshtanov, Optimal broadcast domination of arbitrary graphs in polynomial time, Rept. No. 290, Dept. Informatics, University of Bergen, February 2005.

[76] D. Hershkowitz and H. Schneider, Ranks of zero patterns and sign patterns, *Linear and Multilinear Algebra,* **34**(1993),3-19.

[77] M.S. Jacobson and K. Peters, Chordal graphs and upper irredundance, upper domination and independence, *Discrete Math.,* **86**(1990), 59-69.

[78] D. Kratsch, Algorithms, Chapter 8 in *Domination in Graphs: Advanced Topics*, T.W. Haynes, S.T. Hedetniemi and P.J. Slater, Eds., Marcel Dekker, 191-231,1998.

[79] R. Laskar, J. Pfaff, S.M. Hedetniemi and S.T. Hedetniemi, On the algorithmic complexity of total domination, *SIAM J. Alg. Disc. Meth.,* **5** (1984),420-425.

[80] E.L. Lawler and P.J. Slater, A linear time algorithm for finding an optimal dominating subforest of a tree, In *Graph Theory with Applications to Algorithms and Computer Science*, (Kalamazoo, MI, 1984), Wiley, 501-506, 1985.

[81] V.E. Levit and E. Mandrescu, Unicycle bipartite graphs with only uniquely restricted maximum matchings, *Combinatorics, computability and logic (Constanţa, 2001)*, *Springer Ser. Discrete Math. Theor. Comput. Sci.,* Springer, 151-157, 2001.

[82] V.E. Levit and E. Mandrescu, Bipartite graphs with uniquely restricted maximum matchings and their corresponding greedoids.

[83] V.E. Levit and E. Mandrescu, Local maximum stable sets in bipartite graphs with uniquely restricted maximum matchings, Stability in graphs and related topics. *Discrete Appl. Math.,* **132**(2003), 163-174.

[84] A.A. McRae. External redundant set in graphs, Presented at 25 *th Southeastern. Conf. on Combinatorics, Graph Theory and Computing*, (Boca Raton, FL, 1994).

[85] A.A. McRae. Generalizing NP-completeness proofs for bipartite graphs and chordal graphs, PhD Thesis, Clemson University, 1994.

[86] S.L. Mitchell, Linear algorithms on trees and maximal outerplanar graphs: design, complexity analysis and data structures study, PhD Thesis, University of Virginia, 1977.

[87] S.L. Mitchell and S.T. Hedetniemi, Edge domination in trees, *Congr. Numer.,* **19** (1977), 489-509.

[88] S. Mitchell, S. Hedetniemi and S. Goodman, Some linear algorithms on trees, *Proc. Sixth Southeastern Conf. on Combinatorics, Graph Theory and Computing*, Util. Math., Winnipeg, 467-483, 1975.

[89] J. Pfaff, R. Laskar and S.T. Hedetniemi, NP-completeness of total and connected domination and irredundance for bipartite graphs, Tech. Rept. 428, Dept. Mathematical Sciences, Clemson University, July 1983.

[90] K.E. Proffitt, T.W. Haynes and P.J. Slater, Paired-domination in grid graphs, *Congr. Numer.,* **150**(2001),161-172.

[91] C. Savage, Maximum matchings and trees, *Inform. Process. Lett.,* **10**(4-5)(1980), 202-205.

[92] C. Sloper, Broadcast-coloring of trees, *Rept. No.,* 233 September 2002, Dept. of Informatics, University of Bergen.

[93] U. Stege and I. van Rooij, Profit domination in graphs, Preprint, 2005.

[94] J.A. Telle, Vertex partitioning problems: characterization, complexity and algorithms on partial k-trees, PhD Thesis, University of Oregon, 1994.

[95] P.M. Weichsel, Large subgraphs of hypercubes, In *Steiner Systems and Codes*, preprint.

[96] T.V. Wimer, S.T. Hedetniemi and R. Laskar, A methodology for constructing linear graph algorithms, *Proc. Sundance Conf.*, (Sundance, UT, 1985), *Congr. Numer.* **50**(1985), 43-60.

[97] C.C. Yen and R.C.T. Lee, The weighted perfect domination problem, *Inform. Process. Lett.,* **35**(1990),295-299.

[98] C.C. Yen and R.C.T. Lee, A linear algorithm to solve the weighted perfect domination problem in series-parallel graphs, undated manuscript.

[99] M. Zito, Induced matchings in regular graphs and trees, *WG* 1999:89-100. *Graph-theoretic concepts in computer science* (Ascona, 1999), *Lecture Notes in Comput. Sci.,* 1665:89-100, Springer, Berlin, 1999.

[100] M. Zito, Linear time maximum induced matchings algorithm for trees, *Nordic J. Comput.,* **7**(1)(2000),58.