A linear algorithm for secure domination in trees

AP Burger*, AP de Villiers* & JH van Vuuren*

January 18, 2013

Abstract

A subset \(X\) of the vertex set of a graph \(G\) is a secure dominating set of \(G\) if \(X\) is a dominating set of \(G\) and if, for each vertex \(u\) not in \(X\), there is a neighbouring vertex \(v\) of \(u\) in \(X\) such that the swap set \((X - \{v\}) \cup \{u\}\) is again a dominating set of \(G\). The secure domination number of \(G\), denoted by \(\gamma_s(G)\), is the cardinality of a smallest secure dominating set of \(G\). A linear algorithm is proposed in this paper for finding a minimum secure dominating set and hence the value \(\gamma_s(T)\) for a tree \(T\). The algorithm is based on a strategy of repeatedly replacing pendent spiders with paths in the input tree \(T\).

Keywords: Secure domination, graph protection, linear algorithm, (wounded) spider, tree.

1 Introduction

Let \(G = (V, E)\) be a simple graph of order \(n\). A set \(X \subseteq V\) is a dominating set of \(G\) if every vertex in \(V - X\) is adjacent to some vertex in \(X\). A vertex \(v \in X\) is defended (by itself), while a vertex \(u \notin X\) is defended (by an adjacent vertex \(v \in X\)) if the swap set \((X - \{v\}) \cup \{u\}\) is again a dominating set of \(G\). A set \(X \subseteq V\) is a secure dominating set (SDS) of \(G\) if it is a dominating set of \(G\) and if every vertex of \(G\) is defended. An SDS of \(G\) of minimum cardinality is called a minimum secure dominating set (MSDS) of \(G\) and this minimum cardinality is denoted by \(\gamma_s(G)\), called the secure domination number of \(G\). Various properties of SDSs and bounds on \(\gamma_s(G)\) have been established for various graph classes in the literature [1, 2, 5, 7, 8, 10, 12, 14].

There are numerous applications of the notion of secure domination. If the vertices of \(G\) denote locations on some spatial domain, and the edges model links between these locations along which patrolling guards may move, then a secure dominating set of \(G\) represents a set of locations at which guards may be stationed so that the entire location complex modelled by \(G\) is protected in the sense that if a security concern arises at location \(u\), there will either be a guard stationed at that location who can deal with the problem, or else a guard dealing with the problem from an adjacent location \(v\) will still leave the location complex protected (dominated) after moving from location \(v\) to location \(u\) in order to deal with the problem. In this scenario the secure domination number represents the minimum number of guards required to protect the entire location complex in a bid to save on the total guard remuneration or deployment cost. The above generic algorithm is often realised in the context of surveillance applications, military strategy analysis or the deployment of security guards by private security firms.

The decision problem associated with finding a minimum dominating set of an arbitrary graph is \(\text{NP}\)-complete [11, p. 75]. The decision problem associated with finding an MSDS of an arbitrary graph is therefore also \(\text{NP}\)-complete. Perhaps the fastest exponential-time algorithm for finding

*Department of Logistics, Stellenbosch University, Private Bag X1, Matieland, 7602, South Africa, emails: apburger@sun.ac.za, 14812673@sun.ac.za and vuuren@sun.ac.za
a dominating set of minimal cardinality in an arbitrary graph was proposed by Van Rooij and Bodlaender [15] (in the guise of an algorithm for the celebrated set cover problem), while two exponential-time algorithms (a branch-and-bound algorithm and a branch-and-reduce algorithm) for finding an MSDS in an arbitrary graph were proposed by Burger et al. [4]. Cockayne et al. [9], however, designed a linear-time algorithm for domination in trees which has inspired us to pursue a linear algorithm for finding MSDSs of trees in this paper. The advantage of having a fast algorithm for computing MSDSs of trees is that it may be used to determine an upper bound on the secure domination number \( \gamma_s(G) \) of an arbitrary (connected) graph \( G \), by applying the algorithm to any spanning tree of \( G \).

After reviewing and establishing a number of preliminary results in §2, we describe how an MSDS may be found for a spider (a tree with only one vertex of degree at least 3) in §3. This description serves as foundation on which we base the design of a more general algorithmic approach for finding an MSDS in any tree in §4. Finally, §5 contains a description how the algorithmic approach of §4 may be implemented in linear time and space. In §6 we describe our various approaches towards validating the algorithm presented in §5. The paper closes in §7.

## 2 Preliminary results

Any vertex of degree at least 3 in a tree \( T \) is called a branch vertex of \( T \). If \( T \) contains no branch vertex, then it is a path. The following result, dating from 2005, holds for paths and is due to Cockayne et al. [10].

*Lemma 1 ([10])* If \( T \) is a path \( P_n \) of order \( n \), then \( \gamma_s(P_n) = \lceil \frac{3n}{7} \rceil \).

In the remainder of the paper we therefore assume that \( T \) contains at least one branch vertex. The following result is central in the development for our algorithm.

*Lemma 2* Let \( T \) be a tree and let \( P_{j+1} \) be a path of length \( j \) in \( T \), starting from a leaf of \( T \) but containing no branch vertex of \( T \). Then there is no SDS of \( T \) containing fewer than \( \lceil \frac{3j}{7} \rceil \) vertices of \( P_{j+1} \).

*Proof:* For \( 0 \leq j \leq 2 \) the quantity \( \lceil \frac{3j}{7} \rceil - 1 \) is non-positive. For \( j = 3 \) there is only one possible dominating set for \( P_{j+1} \), and then only in the best-case scenario where the vertex immediately outside \( P_{j+1} \) in \( T \) is included in the dominating set, as shown in Figure 2.1(a). For \( j = 4 \), there is not even a dominating set of cardinality \( \lceil \frac{3j}{7} \rceil - 1 \) for \( P_{j+1} \), let alone an SDS. For \( j = 5 \) there are three possible dominating sets of cardinality \( \lceil \frac{3j}{7} \rceil - 1 \) for \( P_{j+1} \) (in the best-case scenario where the vertex immediately outside \( P_{j+1} \) in \( T \) is included in the dominating set), as shown in Figure 2.1(b). Since none of these dominating sets are SDSs of \( P_{j+1} \), the statement is true for \( j \leq 5 \).

![Figure 2.1](image)

Figure 2.1: Solid vertices form part of the dominating sets of cardinality \( \lceil \frac{3j}{7} \rceil - 1 \) for the subpaths in brackets. Leaves are denoted by square vertices. None of these dominating sets are SDSs.

Furthermore, any stretch of seven consecutive vertices in \( P_{j+1} \) require at least three vertices in any securely dominating set of \( P_{j+1} \) for \( j \geq 6 \). This may be seen by noting that neither of the only
two dominating sets of cardinality \( \lceil \frac{3(6)}{7} \rceil - 1 = 2 \) for the subpath of length 6 in Figure 2.1(c) securely dominates the subpath, even in the best-case scenario where the two vertices immediately outside the subpath are both included in the dominating set.

Now suppose \( j = 7s + t \) for some integer \( 0 \leq t < 7 \). Then it follows that at least

\[
\left\lceil \frac{3t}{7} \right\rceil + 3s \geq \left\lceil \frac{3(7s + t)}{7} \right\rceil - 1
\]

vertices are required to dominate \( P_{j+1} \) securely.

\[\blacksquare\]

3 Spiders

The notion of a spider \(^1\) plays an important role in our algorithmic approach towards determining an MSDS of an arbitrary tree. A spider is a tree with exactly one branch vertex \( b \) and is denoted by \( S(a_1, \ldots, a_r) \), where \( a_i \geq 1 \) is the length of the path, denoted by \( L_i \), between the \( i \)-th leaf \( z_i \) of the spider and \( b \), for all \( 1 \leq i \leq r \). We call the path \( L_i - b \) the \( i \)-th leg, the reduced path \( L'_i = L_i - (N[b] \cap V(L_i)) \) the \( i \)-th reduced leg and the full path \( L_i \) the \( i \)-th extended leg of the spider. We use the notation \( N[b] \) to denote the closed neighbourhood of the branch vertex \( b \), which is the set of all neighbouring vertices of \( b \) together with \( b \) itself. These notions are illustrated in Figure 3.1 for the spider \( S(3, 3, 1) \).

![Figure 3.1: The spider S(3, 3, 1).](image)

We associate a characteristic vector \( a^* = (a_1^*, \ldots, a_r^*) \) with the spider \( S = S(a_1, \ldots, a_r) \), where \( a_i^* \equiv a_i \pmod{7} \), for all \( 1 \leq i \leq r \) and define \( \Lambda_i(S) \) as the unique subset of the vertex set of the \( i \)-th extended leg \( L_i \) of \( S \) with the property that the vertex set of every subpath \( P_{j+1}^i \) of length \( j \) within \( L_i \) which contains the leaf \( z_i \) shares exactly \( \lceil \frac{3j}{7} \rceil \) vertices with \( \Lambda_i(S) \), for all \( 1 \leq i \leq r \). Then the set \( \Lambda_i(S) \) contains the solid vertices shown in Figures 3.2 and 3.3. Furthermore, let

\[
\Lambda(S) = \bigcup_{i=1}^{r} \Lambda_i(S).
\]  

(3.1)

**Corollary 1** Let \( S = S(a_1, \ldots, a_r) \) be a spider. Then \( \gamma_s(S) \geq |\Lambda(S)| \), where \( \Lambda(S) \) is defined in (3.1).

\(^1\)This special kind of tree is sometimes also called a wounded spider.
Figure 3.2: The unique subset Λ_i(S) of the vertex set of an extended leg L_i of a spider S with the property that the vertex set of every subpath of L_i of length j starting at z_i shares exactly \[ \lceil 3j/7 \rceil \] vertices with Λ_i(S).

Proof: By contradiction. Suppose there exists an SDS X of cardinality |X| = |Λ(S)| − 1 for the spider S = S(a_1, …, a_r) with branch vertex b.

We first show that if b ∈ Λ(S), then Λ(S) \ {b} is not an SDS of S. Suppose b ∈ Λ(S) and let X = Λ(S) \ {b}. Then a^*_i is odd for some i ∈ {1, …, r}. For each a^*_i = 1, 3 or 5 it is easy to see that there exists at least one vertex of the leg L_i − b that is not defended by any vertex of X.

We conclude, therefore, that there is at least one leg, L_i − b (say), which is a path of length a_i − 1, which does not contain the branch vertex b and which contains at most |Λ_i(S)| − 1 < \[ \lceil 3(a_i − 1)/7 \rceil \] vertices from X. The spider S is therefore not securely dominated by X as a result of Lemma 2.

The following result shows that the set Λ_i(S) may be augmented very slightly to form an MSDS of a spider S = S(a_1, …, a_r).

Lemma 3 The set Λ_i(S) in (3.1) securely dominates all the reduced legs L'_1, …, L'_r of the spider S with branch vertex b. Moreover, there exists a subset N^* ⊆ N[b] such that Λ_i(S) ∪ N^* is an MSDS for S.

Proof: Note that the branch vertex b is not necessarily defended by any vertex in Λ(S) — see Figure 3.4(a) for a special case in point. Furthermore, when viewed in the context of the entire spider, the vertex of the leg L_i − b that is adjacent to the branch vertex may not be defended, even if it is defended when viewing the extended leg L_i as a path in isolation (because the branch vertex may be required to defend a vertex in another leg of the spider — see the spider in Figure 3.4(b) for an example of this phenomenon). Every vertex of the reduced leg L'_i is, however, dominated securely by Λ_i(S) even when viewed in the context of the entire spider, and it follows by Lemma 2 that Λ_i(S) is a smallest set with this property.

Suppose X is an MSDS for the entire spider. Then it follows by Lemma 2 that another SDS X' of cardinality at most |X| may be formed by “shifting” the vertices of X towards the branch vertex along each reduced leg of the spider, until these vertices conform to the optimum pattern in
Figure 3.2. If, during this shifting process, vertices are shifted “out of” a reduced leg into the closed neighbourhood $N[b]$ of the branch vertex $b$, then these vertices are merely superimposed on top of the pattern $\Lambda(S) \cap N[b]$. If, however, vertices are shifted “out of” an extended leg of the spider, then these vertices are discarded (disregarded). Clearly, $\Lambda(S)$ and $X'$ differ only within the set $N[b]$, and, moreover, $X'$ is an MSDS of the spider with the property that $\Lambda(S) \cap N[b] \subseteq X' \cap N[b]$. Therefore the statement of the lemma holds with $N^* = X' \setminus \Lambda(S)$.

Figure 3.4: Solid vertices are included in $\Lambda(S)$. (a) A spider $S(7,7,7)$ with characteristic vector $a^* = (0,0,0)$ for which the branch vertex is not defended by any vertex in $\Lambda(S)$. (b) The spider $S(3,3,1)$ with characteristic vector $a^* = (3,3,1)$ in which some vertices in $N[b]$ are not defended.

The following result is a characterisation of when the set $\Lambda(S)$ in (3.1) securely dominates the entire spider $S$, i.e. when $N^* = \emptyset$ in Lemma 3.

**Lemma 4 (Characterisation of when $\Lambda(S)$ is an SDS)** Let $S$ be a spider with characteristic vector $a^* = (a_1^*, \ldots, a_r^*)$. Then the set $\Lambda(S)$ in (3.1) is an MSDS of $S$ if and only if:

(a) $3 \notin a^*$ and $a^*$ contains at most one unit entry and at least one odd entry, or
(b) $3 \in a^*$ and $1 \notin a^*$, or
(c) $1, 3, 5 \notin a^*$ and $6 \in a^*$, or
(d) $1, 3, 5, 6 \notin a^*$ and $4 \in a^*$ and $a^*$ contains at most $r - 2$ zero entries.

**Proof:** It follows by Lemma 3 that we need only verify that $\Lambda(S)$ securely dominates the closed neighbourhood $N[b]$ of the branch vertex $b$ of $S$ in, and only in, the above four cases. Note that if $\Lambda(S)$ securely dominates $S$, then $\Lambda(S)$ is necessarily an MSDS of $S$ by Corollary 1. Denote the neighbour of $b$ in the extended leg $L_i$ of $S$ by $v_i$ for all $1 \leq i \leq r$ throughout the proof of this lemma. We consider two mutually exclusive cases, namely where the characteristic vector $a^*$ contains some odd entries, and where it contains only even entries.

**Case 1: $a^*$ contains some odd entries.** In this case $b \in \Lambda(S)$ and hence we must verify that the open neighbourhood $N(b)$ is securely dominated by $\Lambda(S)$ in, and only in, the above cases. We distinguish between two subcases (corresponding to parts (a) and (b) of the lemma):

**Subcase 1(a):** $3 \notin a^*$. If it also holds that $1 \notin a^*$, then the only odd entries in $a^*$ have the value 5. If $a_i^* = 2, 4$ or $6$, then $v_i \in \Lambda(S)$. However, if $a_i^* = 0$ or 5, then $v_i \notin \Lambda(S)$ but $v_i$ is defended by its neighbour in $L_i - b$, and so $\Lambda(S)$ is an SDS of $S$.

If $a^*$ contains exactly one unit entry, $a_i^* = 1$ (say), then $b$ defends $v_j$, all the other vertices in $N(b)$ are defended as above, and $\Lambda(S)$ is again an SDS of $S$.

If $a^*$ contains at least two unit entries, $a_i^* = 1$ and $a_k^* = 1$ (say), then $b$ can defend at most one of $v_j$ and $v_k$, so that $\Lambda(S)$ is not an SDS of $S$.

**Subcase 1(b):** $3 \in a^*$. If $1 \notin a^*$, then the only odd entries in $a^*$ are the values 3 and (possibly) 5. If $a_i^* = 3$, then $b$ defends $v_i$. If $a_i^* = 2, 4$ or 6, then $v_i$ defends itself. Finally, if $a_i^* = 0$ or 5, then $v_i$ is defended by its neighbour in $L_i - b$. $\Lambda(S)$ is therefore an SDS of $S$.

If $a^*$ contains both a unit entry and the value 3, $a_j^* = 1$ and $a_k^* = 3$ (say), then $b$ cannot defend both $v_j$ and $v_k$, so that $\Lambda(S)$ is not an SDS of $S$.
Case 2: $a^*$ contains only even entries. If $a_2^* = 2, 4$ or $6$, then $v_i \in \Lambda(S)$. Furthermore, if $a_i^* = 0$, then $v_i$ is defended by its neighbour in $L_1 \setminus b$. Since $b \notin \Lambda(S)$, we must verify that $b$ is defended by some vertex in $\Lambda(S)$ in, and only in, the above cases. Again we distinguish between two subcases (corresponding to parts (c) and (d) of the lemma):

Subcase 2(a): $6 \in a^*$. Suppose $a_j^* = 6$. Then $b$ is defended by $v_j$, and so $\Lambda(S)$ is an SDS of $S$.

Subcase 2(b): $6 \notin a^*$. If $a^*$ contains the value 4, $a_2^* = 4$ (say), and $a^*$ contains at most $r - 2$ zero entries, then $b$ is defended by $v_j$, and so $\Lambda(S)$ is an SDS of $S$.

If $a^*$ contains the value 4, $a_2^* = 4$ (say), and $r - 1$ zero entries, then $v_j$ is required to defend its neighbour in the reduced leg $L'_j$ while $v_i \notin \Lambda(S)$ for all $i \neq j$, leaving $b$ undefended. Hence, $\Lambda(S)$ is not an SDS of $S$.

Finally, if $4 \notin a^*$, then the only entries in $a^*$ are the values 0 or 2, and so every neighbour $v_i$ of $b$ is required to defend its neighbour in the reduced leg $L'_i$, leaving $b$ undefended, and hence $\Lambda(S)$ is not an SDS of $S$.

Let

$$R_1(S) = \bigcup_{i=1}^{r} V(L_i - b). \quad (3.2)$$

If $R_1(S) \neq \emptyset$, let $x$ be an arbitrary vertex of $R_1(S)$ and let

$$R'_1(S) = R_1(S) - \{x\}. \quad (3.3)$$

The following result contains a specification of how the set $\Lambda(S)$ should be augmented to form an SDS $\overline{\Lambda}(S)$ of the spider in the four subcases of the proof of Lemma 4 when $\Lambda(S)$ itself is not an SDS of the spider.

**Lemma 5 (If $\Lambda(S)$ is not an SDS, how to extend it to an MSDS $\overline{\Lambda}(S)$)** Let $S$ be a spider with characteristic vector $a^* = (a_1^*, \ldots, a_r^*)$ and branch vertex $b$. Furthermore, let $R_1(S)$ and $R'_1(S)$ be the sets in (3.2) and (3.3).

(a) If $3 \notin a^*$, but $a^*$ contains at least 2 unit entries, then $\overline{\Lambda}(S) = \Lambda(S) \cup R'_1(S)$ is an MSDS of $S$.

(b) If $1, 3 \in a^*$, then $\overline{\Lambda}(S) = \Lambda(S) \cup R'_1(S)$ is an MSDS of $S$.

(c) If $1, 3, 4, 5, 6 \notin a^*$, then $\overline{\Lambda}(S) = \Lambda(S) \cup \{b\}$ is an MSDS of $S$.

(d) If $1, 3, 5, 6 \notin a^*$ and $a^*$ contains at least $r - 1$ zero entries, then $\overline{\Lambda}(S) = \Lambda(S) \cup \{b\}$ is an MSDS of $S$.

**Proof:**

(a) If $3 \notin a^*$ but $a^*$ contains at least two unit entries, then $b \in \Lambda(S)$ and every vertex of $N[b] \setminus R_1(S)$ is defended by some vertex in $\Lambda(S)$ other than $b$. Since any vertex $v \in R_1(S)$ can only be defended by itself or by $b$ and since $b$ can defend at most one vertex in $R_1(S)$, it follows that $\Lambda(S) \cup R'_1(S)$ is an MSDS of $S$ by Lemma 3.

(b) If $1, 3 \in a^*$, then $b \in \Lambda(S)$ is required to defend all its neighbours in legs for which $a_i^* = 3$. Moreover, all the neighbours of $b$ in legs for which $a_i^*$ is even are defended by vertices in $\Lambda(S)$ other than $b$. Since every vertex $v \in R_1(S)$ can be defended by itself only, $\Lambda(S) \cup R'_1(S)$ is an MSDS of $S$ by Lemma 3.

(c) If $1, 3, 4, 5, 6 \notin a^*$, then $a^*$ only contains the values 0 and/or 2, and so $b$ is the only vertex in $N[b]$ that is not defended by any vertex in $\Lambda(S)$. Since $\Lambda(S) \cup \{b\}$ is, however, an SDS of $S$, it is also an MSDS of $S$ by Lemma 3.

(d) If $1, 3, 5, 6 \notin a^*$ and $a^*$ contains at least $r - 1$ zeros, then $a_i^* = 0, 2$ or 4 (say) and $a_i^* = 0$ for all $i \neq j$. In this case $b$ is only dominated (but not defended) by $v_j \in \Lambda(S)$. Since $\Lambda(S) \cup \{b\}$ is, however, an SDS of $S$, it is also an MSDS by Lemma 3.
Figure 3.5: Solid vertices are included in the set $\Lambda(S)$ in (3.1). The union of the sets of solid and grey vertices represent the MSDS $\overline{\Lambda}(S)$ for the spider in each case. (a) The spider $S(3, 1, 8, 2)$ with characteristic vector $a^* = (3, 1, 1, 2)$ for which the vertices $\{v_2, v_3\}$ in $N(b)$ are not securely dominated by $\Lambda(S)$. (b) The spider $S(2, 7, 9)$ with characteristic vector $a^* = (2, 0, 2)$ for which the branch vertex is not securely dominated by $\Lambda(S)$.

The manner in which $\Lambda(S)$ in (3.1) is augmented to form an SDS $\overline{\Lambda}(S)$ for a spider $S$ according to Lemma 5 is illustrated by means of two examples in Figure 3.5. The spider $S(3, 1, 8, 2)$ in Figure 3.5(a) has characteristic vector $a^* = (3, 1, 1, 2)$. The set $\Lambda(S)$ for this spider is denoted by the solid vertices in the figure. The set $\{v_2, v_3\}$ is added to $\Lambda(S)$ to yield the MSDS $\overline{\Lambda}(S)$ according to Lemma 5(b), which is indicated by the solid and grey vertices combined. The spider $S(2, 7, 9)$ in Figure 3.5(b) has characteristic vector $a^* = (2, 0, 2)$. Again the set $\Lambda(S)$ is denoted by the solid vertices. The branch vertex $b$ is added to $\Lambda(S)$ to deliver the MSDS $\overline{\Lambda}(S)$ by Lemma 5(c), which is again indicated in the figure by the solid and grey vertices combined.

4 From pendent spiders to trees in general

A pendent spider $S$ of a tree $T$ is a subtree of $T$ containing exactly one branch vertex $b$ of $T$ and $r - 1$ leaves of $T$, such that the degree of $b$ in $S$ is $r - 1$, where $r$ is the degree of $b$ in $T$. All the pendent spiders of the tree $T_1$ in Figure 4.1(a) are highlighted in grey in Figure 4.1(b). Let $v$ be the (only) neighbour of $b$ in $T$ that is not also in a pendent spider $S$ of $T$. The pendant path in $T$ associated with $S$ is the path starting at $v$ and continuing into $T - S$ up to (and including) the first branch vertex encountered in $T - S$. The pendant paths of the pendent spiders in $T_1$ are shown in boldface in Figure 4.1(b).

Figure 4.1: (a) A tree $T_1$ containing five branch vertices $b_1, b_2, b_3, b_4$ and $b_5$. (b) The three pendent spiders $S_1, S_2$ and $S_3$ of $T_1$ are highlighted in grey, while the corresponding pendant paths are shown in boldface.

Our approach towards constructing an MSDS for an arbitrary tree $T$ will hinge on repeatedly pruning away spiders from $T$ after having dominated the pendent spiders securely (with the possible exception of the vertex $b$, which may be defended by $v$), until only a final spider remains. We
therefore need to be able to compute a subset of the vertex set of \( S \) which forms part of some MSDS of \( T \). However, a pendent spider \( S \) with branch vertex \( b \) is itself a spider when viewed in isolation. We may therefore also refer to the characteristic vector of a pendant spider, and an approach very similar to the one described in §3 may be adopted to construct an SDS \( \Lambda(S) \) for a pendant spider, as described in Lemma 5.

The only difference between securely dominating a spider optimally and securely dominating a pendant spider optimally is that care should be taken when deciding whether or not to include the branch vertex \( b \) in \( \Lambda(S) \) in the case of a pendent spider, because then \( b \) is joined to a pendent path which extends to another branch vertex in \( T \). For this reason we define the set

\[
\Lambda^*(S) = \begin{cases} 
\Lambda(S) & \text{if any one of the four conditions in Lemma 4 holds for } S, \\
\Lambda(S) & \text{as defined in Lemma 5(a) or 5(b) if any one of the corresponding conditions holds and } S \text{ is a pendent spider of } T, \\
\Lambda(S) & \text{as defined in Lemma 5(a), 5(b), 5(c) or 5(d) if any one of the corresponding conditions holds and } S = T.
\end{cases}
\]

(4.1)

We immediately have the following result as a corollary of Lemma 2.

**Corollary 2** Let \( S \) be a pendent spider of a tree \( T \). Then no SDS of \( T \) contains fewer than \( |\Lambda^*(S)| \) vertices from \( S \).

The proof of Corollary 2 is virtually identical to that of Corollary 1 and is hence omitted. The decision of whether or not to include \( b \) in the SDS of the entire tree \( T \) is facilitated by labelling the vertices of \( T \) carefully.

Our approach towards continuing the SDS pattern of Figure 3.2 from the branch vertex \( b \) of a pendant spider \( S \) with characteristic vector \( a^* = (a_1^*, \ldots, a_{r-1}^*) \) into the rest of the tree \( T \) along the pendent path of \( S \) is based on the assignment of a list of labels to each vertex of \( S \). Initially these lists of labels are all empty. The label list of each vertex \( u \) in the extended leg \( L_i \) of \( S \) is appended by including the distance modulo 7 from the leaf \( z_i \) in \( L_i \) to \( u \) in the list, for all \( i = 1, \ldots, r - 1 \). Note, therefore, that the label list of \( b \) contains the values \( a_1^*, \ldots, a_{r-1}^* \), while the label lists of every other vertex of \( S \) contains a single entry, as illustrated in Figure 4.2.

![Figure 4.2: The label lists of the vertices of the pendent spider \( S_1 \) of \( T_1 \) in Figure 4.1(b) with characteristic vector \( a^* = (7, 3, 2, 1) \). The branch vertex is indicated in grey and is the only vertex with more than one entry in its label list.](image)

The non-branch vertices of \( S \) are included in the SDS of \( T \) if they form part of the set \( \Lambda^*(S) \) (i.e. if their label lists contain odd entries). The inclusion or otherwise of the branch vertex \( b \) in the SDS of \( T \) is decided by assigning an *effective label* to \( b \), denoted by \( \ell(b) \), which depends on the contents of the characteristic vector and the label list of \( b \). The effective label \( \ell(b) \) is defined to be the most favourable value in Table 4.1 for which the corresponding description holds (where \( v \) is the neighbour of \( b \) on the pendent path of \( S \)). Note, therefore, that the value of the effective label \( \ell(b) \) of \( b \) depends on the specific structure of the set \( \Lambda^*(S) \) of vertices within \( T \).
Table 4.1: Definition of the labels of the branch vertex \( b \) of a pendant spider \( S \) in a tree \( T \), with \( v \) being the neighbour of \( b \) on the pendant path of \( S \) in \( T \).

The function of the effective label \( \ell(b) \) is that it serves as a starting point for the SDS pattern in Figure 3.2 along the pendant path of \( S \) into the rest of \( T \), as illustrated in Figure 4.3. The reason for choosing the effective label as favourably as possible (as indicated in Table 4.1) is to ensure that the final SDS constructed for \( T \) is as small as possible. In order to achieve this, the occurrence of two adjacent vertices along the fictitious path of length \( \ell(b) \) joined to the pendant path that are not in the SDS of \( T \) should be induced as soon as possible. Since two adjacent vertices not in \( \Lambda \) have the label 6 followed by the label 0, as may be seen in Figure 4.3, a more favourable effective label of \( b \) in Table 4.1 may potentially lead to a smaller eventual SDS of \( T \).

![Diagram of SDS pattern](image)

Figure 4.3: Using the effective label to determine the starting point for the continuation of the SDS pattern in Figure 3.2 along the pendant path of a spider into the rest of the tree.

The value of the effective label of the branch vertex of a pendant spider with characteristic vector \( a^* = (a^*_1, \ldots, a^*_r) \) is established in our next result. A few definitions are, however, required in the proof of this result. Let \( X \) be any SDS of a tree \( T \). A vertex \( v \in X \) is a single-defender if it defends a single vertex in \( T \) other than itself. Similarly, a vertex \( v \in X \) is a multi-defender if it defends at least two vertices in \( T \) other than itself. A vertex \( u \notin X \) is s-defended if it is defended by a single-defender in \( T \) and m-defended if it is defended by at least one multi-defender, but by no single-defender.

**Lemma 6 (The effective label of the branch vertex of a pendant spider)**

Let \( S \) be a pendant spider with characteristic vector \( a^* = (a^*_1, \ldots, a^*_r) \) and branch vertex \( b \) with effective label \( \ell(b) \). If \( 1, 3, 5, 6 \notin a^* \), but \( 4 \in a^* \) and \( a^* \) contains at most \( r - 2 \) zero entries, then \( \ell(b) = 6 \). Otherwise \( \ell(b) \) is the first element of the ordered set \( \{3, 1, 5, 6, 4, 2, 0\} \) also in \( a^* \). Moreover, no SDS of \( S - b \) of cardinality \( |\overline{\Lambda}(S)| \) other than \( \overline{\Lambda}(S) \) is capable of inducing a more favourable label value for \( b \) than the effective label \( \ell(b) \).
**Proof:** Denote the neighbour of \( b \) in the extended leg \( L_i \) of \( S \) by \( v_i \), for all \( 1 \leq i \leq r - 1 \) and let \( v \) be the neighbour of \( b \) on the pendant path of \( S \) throughout the proof of this lemma. We proceed systematically to verify that the most favourable feasible label value in Table 4.1 is the effective label \( \ell(b) \) of \( b \). We consider two mutually exclusive cases, namely where the characteristic vector \( a^* \) contains only even entries, and where it contains some odd entries.

**Case 1:** \( a^* \) contains only even entries. In this case \( b \notin \overline{N}(S) \) and \( \ell(b) \neq 1, 3, 5 \), since \( b \) cannot defend any vertex in \( N[b] \).

If \( a^*_e = 0 \), then \( \ell(b) \neq 4, 6 \), since \( b \) is not defended by any vertex in \( S \). Also, \( \ell(b) \neq 2 \), since \( b \) is not dominated by any vertex in \( S \), ensuring that \( \ell(b) = 0 \) (as \( b \) must be \( s \)-defended by \( v \) and is not dominated by any vertex in \( S \)).

If \( 4, 6 \notin a^* \) and \( 2 \in a^* \), then \( \ell(b) \neq 6 \), (since \( b \) is not \( s \)-defended by a vertex in \( S \)) and \( \ell(b) \neq 4 \) (since it is not \( m \)-defended by any vertex in \( S \)). Thus \( \ell(b) = 2 \) (since \( b \) can be \( m \)-defended by \( v \) and is dominated by a vertex in \( S \)).

If \( 6 \notin a^* \) and \( 4 \in a^* \), we consider three subcases.

*Subcase i:* The vector \( a^* \) contains at least two 4-entries, \( a^*_1 = 4 \) and \( a^*_2 = 4 \) (say). In this case \( \ell(b) = 6 \). Since \( b \) is \( m \)-defended by \( v_i \) and \( v_j \), \( b \) remains dominated by \( v_i \) or \( v_j \) for any swap set of \( \overline{X}(S) \).

*Subcase ii:* The vector \( a^* \) contains both a single 4-entry and a 2-entry, \( a^*_1 = 4 \) and \( a^*_2 = 2 \) (say). In this case again \( \ell(b) = 6 \), since \( b \) is \( m \)-defended by \( v_i \) and dominated by \( v_j \).

*Subcase iii:* The vector \( a^* \) contains a single 4-entry, \( a^*_1 = 4 \) (say), and no 2-entry. In this case \( \ell(b) = 4 \), since \( b \) is \( m \)-defended by \( v_i \) and can be defended by \( v \).

If \( 6 \in a^* \), then \( \ell(b) = 6 \), since \( b \) is \( s \)-defended by a vertex in \( S \).

**Case 2:** \( a^* \) contains some odd entries. In this case \( b \in \overline{N}(S) \) and \( \ell(b) \neq 0, 2, 4, 6 \), since \( b \) defends at least one vertex in \( N[b] \).

If \( 1, 3 \notin a^* \), then \( \ell(b) = 5 \), as \( b \) is a single-defender of \( v \), and is not required to defend any vertex \( N[b] - \{v\} \). Note that 5 is the most favourable effective label value for 6 in this case.

If \( 3 \notin a^* \) and \( 1 \in a^* \), then \( b \) is a single-defender of a single a vertex in \( S \), and hence \( \ell(b) = 1 \). Note that in this case 5 is not a more favourable effective label value than 1 for \( b \), because \( a^* \) contains at least one 1-entry, \( a^*_1 = 1 \) (say), and \( b \) is a single-defender of \( v_i \).

If \( 3 \in a^* \), then \( b \) is multi-defender of \( v \) and a vertex in \( S \), and hence \( \ell(b) = 3 \). Note that in this case 5 is not a more favourable effective label value than 3 for \( b \), because \( a^* \) contains at least one 3-entry, \( a^*_1 = 3 \) (say), and \( b \) is a multi-defender of \( v_i \) and \( v \).

It follows by Corollary 2 that no configuration of fewer than \( |\overline{N}(S)| \) vertices of \( S \) can form an SDS of \( S \). To show that no other configuration \( X \) of \( |\overline{N}(S)| \) vertices in an SDS of \( S - b \) can induce a more favourable effective label \( \ell(b) \) of \( b \) than does the set \( \overline{N}(S) \), we first note that by Lemmas 2 and 5(a)–(b) the only legs of \( S \) that can possibly share fewer vertices with another SDS \( X \) of \( S - b \) than with \( \overline{N}(S) \) are those for which \( a^*_i = 1 \). Therefore, if \( 1 \notin a^* \), the desired result follows directly from Lemma 2. It is easy to see that the only case where all legs of \( S \) for which \( a^*_i = 1 \) can collectively share fewer vertices with another SDS \( X \) of \( S - b \) than with \( \overline{N}(S) \) is when \( 3 \in a^* \) (see Figure 4.4). But clearly the effective label of \( b \) induced in this case by \( X \) is less favourable than that induced by \( \overline{N}(S) \). We conclude that we only have to consider configurations \( X \) for which each leg of \( S \) shares the same number of vertices with \( X \) as it does with \( \overline{N}(S) \). If such a configuration \( X \) of vertices of \( S \) were to exist, some leg \( L_i \) of \( S \) would have to contain a vertex \( u \) of \( X \) that is closer to \( b \) than the closest vertex vertex of \( L_i \) that is in \( \overline{N}(S) \). If \( a^*_i \neq 1 \), then Lemma 2 is contradicted. But if \( a^*_i = 1 \), then \( u \notin N[b] \) and so either Lemma 2 is contradicted or else \( v_i \in \overline{N}(S) \), again a contradiction. ■
As mentioned above, the purpose of the effective label $\ell(b)$ of the branch vertex $b$ of a pendent spider $S$ in a tree $T$ is to facilitate a continuation of the SDS pattern in Figure 3.2 from $b$ into the rest of $T$ along the pendent path of $S$. The following result shows that such a continuation approach towards securely dominating $T$ is viable.

**Lemma 7** Let $S$ be a pendent spider of a tree $T$. Then there exists an MSDS of $T$ containing the set $\overline{X}(S)$ in (4.1).

**Proof:** Let $X$ be the smallest possible subset of vertices from $T - S$ for which $\overline{X}(S) \cup X$ is an SDS of $T$. If $\overline{X}(S) \cup X$ is an SDS of $T$, then the proof is complete. Otherwise there is an MSDS $Y$ of $T$ which is smaller than $\overline{X}(S) \cup X$. But $|Y \cap V(S)| \geq |\overline{X}(S)|$ by Corollary 2. Furthermore, $|Y \cap V(T - S)| \geq |X|$ since $b$ has the most favourable label in Table 4.1 by Lemma 6. But then $|Y| = |((Y \cap V(S)) \cup (Y \cap V(T - S)))| = |Y \cap V(S)| + |Y \cap V(T - S)| \geq |\overline{X}(S)| + |X| = |\overline{X}(S) \cup X|$, a contradiction. ■

Although the effective label $\ell(b)$ of the branch vertex $b$ of a pendent spider $S$ serves the purpose of indicating the starting point (effectively a left-shift) for continuing the SDS pattern of Figure 3.2 into the rest of the tree $T$ along the pendent path of $S$ (after having pruned away $S$ from $T$), as explained in Figure 4.3, we prefer to adopt a slightly different approach to managing the continuation of the SDS pattern along the pendent path of $S$ into $T$. Since the pendent path of a pendent spider is, of course, the start of a leg of another pendent spider of $T$, the effective label $\ell(b)$ may also be thought of as the length of a fictitious pendent path attached to $b$ in the place of $S$ (after having pruned away the legs of $S$ from $b$, thus, in effect, artificially extending or lengthening the pendent path) after which the SDS pattern of Figure 3.2 is applied from the start along the combined fictitious and pendent paths into the rest of $T$ without the need of having to keep track of a starting point (or effective left-shift) of the SDS pattern in Figure 3.2 for application to the pendent path of $S$. This alternative interpretation of the meaning of the effective label $\ell(b)$ of $b$ is illustrated in Figure 4.5 and has the advantage that none of the results of §3 have to be adapted to incorporate label values when considering pendent spiders instead of spiders.

![Figure 4.4](image1.png)

(a) $\overline{X}(S)$ ($\ell(b) = 3$)

(b) $X$ ($\ell(b) = 1$)

Figure 4.4: (a) The effective label of $b$ is 3, since $b$ is a multi-defender of $v$ and $v_j$ with $a_j^* = 3$. (b) The effective label of $b$ is 1, since $b$ is a single-defender of $v_j$ with $a_j^* = 1$.

![Figure 4.5](image2.png)

Figure 4.5: An alternative interpretation of the effective label $\ell(b)$ of the branch vertex $b$ of a pendent spider $S$ of $T$.

When adopting the alternative interpretation of the meaning of the effective label $\ell(b)$ of $b$, all the results of §3 for spiders (in particular, Lemmas 4 and 5) also hold as-is for pendent spiders, because
then all the (artificially extended) legs of the pendent spider require exactly the SDS patterns in Figures 3.2 and 3.3. The effect of the additional vertices on the fictitious pendent path extension of length ℓ may be neglected by subtracting the value of \([3ℓ(b)/7]\) (by Lemma 1) from the final value computed for \(γ_s(T)\), as is made more precise in the following result.

**Lemma 8** Suppose \(S\) is a pendent spider with branch vertex \(b\) in a tree \(T\). If another tree \(T'\) is formed by replacing \(S\) with a path \(P_{1+ℓ(b)}\) of length \(ℓ(b)\) (attached after the removal of the legs of \(S\)), then \(γ_s(T) = γ_s(T') + |X(S)| − [3ℓ(b)/7]\).

**Proof:** It follows by Lemmas 1 and 7 that there exists an MSDS \(X\) of \(T'\) containing \([3ℓ(b)/7]\) vertices of \(P_{1+ℓ(b)}\) and an MSDS of \(T\) containing \(|X(S)|\) vertices of \(S\). Therefore, \(γ_s(T − S) = γ_s(T) − |X(S)|\) and \(γ_s(T' − P_{1+ℓ(b)}) = γ_s(T') − [3ℓ(b)/7]\). The result follows since \(T − S = T' − P_{1+ℓ(b)}\).

Given a tree \(T\), our approach towards computing the value of \(γ_s(T)\) is to select a pendent spider \(S\) of \(T\), compute an SDS for this spider \(S\), prune away the spider, and repeat the process for the smaller tree thus formed, until a tree is reached which is itself a spider for which the set \(X(S)\) as defined in Lemma 5 is an MSDS. The value of \(γ_s(T)\) is then the sum of the cardinalities of the MSDSs of spiders pruned away and that of the final spider, as we prove next.

**Theorem 1** Let \(T_1, T_2, \ldots, T_Ω\) be the sequence of trees in which \(T_{i+1}\) is formed from \(T_i\) by pruning away a pendent spider \(S_i\) from \(T_i\) for all \(i = 1, \ldots, Ω − 1\) until a spider \(T_Ω\) is reached. Then

\[
γ_s(T_1) = \sum_{i=1}^{Ω} |X(S_i)|.
\]

**Proof:** It follows by Lemma 8 that \(γ_s(T_i) = γ_s(T_i') + |X(S_i)| − [3ℓ(b_i)/7]\), where \(T_i'\) is formed by replacing \(S_i\) with a fictitious path \(P_{1+ℓ(b_i)}\) for all \(i = 1, \ldots, Ω − 1\). But then \(T_{i+1} = T_i' − P_{1+ℓ(b_i)}\) so that

\[
γ_s(T_i) = γ_s(T_{i+1}) + \left[\frac{3ℓ(b_i)}{7}\right] + |X(S_i)| − \left[\frac{3ℓ(b_i)}{7}\right] = γ_s(T_{i+1}) + |X(S_i)|;
\]

which enables us to write

\[
|X(S_i)| = γ_s(T_i) − γ_s(T_{i+1}), \quad i = 1, \ldots, Ω − 1.
\]

Therefore,

\[
\sum_{i=1}^{Ω} |X(S_i)| = |X(S_Ω)| + \sum_{i=1}^{Ω-1} |X(S_i)| = |X(S_Ω)| + \sum_{i=1}^{Ω-1} [γ_s(T_i) − γ_s(T_{i+1})] \quad \text{by (4.2)}
\]

\[
= |X(S_Ω)| + [γ_s(T_1) − γ_s(T_2)]
\]

\[
= |X(S_Ω)| + γ_s(T_1) − γ_s(T_2)
\]

\[
= |X(S_Ω)| + γ_s(T_1) − |X(S_1)|
\]

\[
= γ_s(T_1),
\]

as desired.

Consider, as an example, the tree \(T_1\) in Figure 4.6(a) with pendent spider \(S_1\) which has the characteristic vector \((7, 3, 2, 1)\). The set \(X(S_1)\) is indicated by means of the solid vertices in the
figure and has cardinality $|\overline{X}(S_1)| = 8$. The label list of the branch vertex $b_1$ is $\{1, 3, 2, 1\}$. The effective label $\ell(b)$ is therefore 3 according to Lemma 6. This value represents a starting point for the SDS pattern in Figure 3.2 into $T_2 = T_1 - S_1$ along the pendant path of $S_1$, as shown in Figure 4.6(b). This pendant path is, of course, a leg of another pendant spider in $T_2$, which will be securely dominated later. The pendant spider $S_2$ in the tree $T_2$ has the characteristic vector $(2, 2)$ and the set $\overline{X}(S_2)$ has cardinality $|\overline{X}(S_2)| = 2$, as indicated by means of the solid vertices in Figure 4.6(c). The label list of $b_2$ is $\{2, 2\}$ and so the effective label of $b_2$ is 2 by Lemma 6. The pendant spider $S_3$ in $T_3 = T_2 - S_2$, as shown in Figure 4.6(c), has the characteristic vector $(1, 1)$, which delivers a label list $\{1, 1\}$ for the branch vertex $b_3$ and so the effective label of $b_3$ is 1. The set $\overline{X}(S_3)$ is indicated by the $|\overline{X}(S_3)| = 2$ solid vertices in the figure. Figure 4.6(d) shows the tree $T_4 = T_3 - S_3$, in which $b_4$ has label list $\{2, 4\}$ and effective label 6 according to Lemma 6. The set $\overline{X}(S_4)$ is indicated by the solid vertex in the figure. The tree $T_5 = S_5 = T_4 - S_4$ is a spider with characteristic vector $(1, 3, 5)$ for which the set $\overline{X}(S_5)$ has cardinality $|\overline{X}(S_5)| = 3$, as indicated by means of the solid vertices in Figure 4.6(f). An MSDS of $T_1$ therefore has cardinality $\gamma_5(T_1) = |\overline{X}(S_1)| + |\overline{X}(S_2)| + |\overline{X}(S_3)| + |\overline{X}(S_4)| + |\overline{X}(S_5)| = 8 + 2 + 2 + 1 + 3 = 16$, according to Theorem 1.

5 Linear algorithmic implementation

An ordering of the vertices of a tree $T$ of order $n$ is an assignment of the indices $1, \ldots, n$ to the vertices of $T$, one index to a vertex. A canonical ordering of the vertices of a rooted tree $T$ is an ordering of the vertices of $T$ such that the index of the parent of vertex $i$, denoted by $\text{Parent}[i]$, is smaller than the vertex indexed $i$. The root of $T$ therefore has index 1 and we adopt the special convention that the “index” of $\text{Parent}[1]$ is 0. We assume, without loss of generality, that the root of the tree $T$ for which an MSDS is sought is a branch vertex of $T$. Let $\text{Parent}[i_1, \ldots, i_j]$ denote the set of vertex indices of the parents of the vertices indexed $i_1, \ldots, i_j$. Then a canonical ordering of the vertices of $T$ has the property that the tree induced by the set $\text{Parent}[1, \ldots, k]$ is a subtree of the tree induced by the set $\text{Parent}[1, \ldots, m]$ if $k \leq m$ [9, 13].

Our algorithm for secure domination of a (rooted) tree $T$ follows the approach described in §4 by traversing each vertex of $T$ once while constructing an MSDS for $T$, deciding whether or not that vertex should be included in the MSDS and occasionally including a previously visited vertex in the MSDS if certain conditions apply. Six linear arrays are maintained during this traversal process:

- $\text{Parent}[i]$ contains the index of the parent of vertex $i$ in a canonical ordering of the vertices of $T$.
- $\text{Branch}[i]$ contains the Boolean value True if vertex $i$ is a branch vertex of $T$, or the Boolean value False otherwise.
- $\text{Labels}[i]$ is initialised as an array of seven zeros. The entry in position $j \in \{0, \ldots, 6\}$ of $\text{Labels}[i]$ is denoted by $\text{Labels}[i, j]$ and eventually represents the number of times the value $j$ occurs in the label list of the vertex indexed $i$. This array is updated by increasing $\text{Labels}[$$\text{Parent}[i], \ell(i) + 1 \mod 7$$]$ by one as vertex $i$ is visited during the traversal process. Note that if $i$ is a branch vertex, then the weight (sum of the entries) of the array $\text{Labels}[i]$ will eventually be more than one, whereas the weight of $\text{Labels}[i]$ will eventually be one if $i$ is not a branch vertex.
- $\text{PreviousLabel}[i]$ contains the last vertex index of the child of branch vertex $i$ that caused the branch vertex to be assigned a value of 1 in its label list $\text{Labels}[i]$. It is necessary to keep track of this index because if a branch vertex $i$ has a zero value in $\text{Labels}[i, 3]$, all its children with zero labels have to be included in the SDS according to Lemma 6. It is therefore only possible to include $i$ in the SDS if the next 1-label is included in the list $\text{Labels}[i]$. However, if $\text{Labels}[i, 3]$ is already positive, then $i$ can immediately be included in the SDS by Lemma 6. This explains the necessity of our next linear array, namely $\text{A3Label}$. 

13
Figure 4.6: (a) The pendant spider $S_1$ with branch vertex $b_1$ of the tree $T_1$ in Figure 4.1 connected by a pendent path to the rest of the tree. (b) The pendant spider $S_2$ with branch vertex $b_2$ of the tree $T_2$ after having pruned away the pendant spider $S_1$. (c) The pendant spider $S_3$ with branch vertex $b_3$ of the tree $T_3$ after having pruned away the pendant spider $S_2$. (d) An intermediate step where the pendant spider $S_3$ is pruned from $T_3$ to deliver the tree $T_4$. (e) The pendant spider $S_4$ with branch vertex $b_4$ of the tree $T_4$. (f) The final spider $S_5$ with branch vertex $b_5$.

$A3Label[i]$ is initialized to contain the Boolean value FALSE and is then updated to contain the Boolean value TRUE as soon as it becomes known that the value $Labels[i,3]$ is positive and $Branch[i]$ is TRUE.

$X[i]$ is initialised to contain the boolean value FALSE and is updated to contain the boolean value TRUE if the vertex indexed $i$ is at some point included in the MSDS of $T$.

Steps 1–6 of Algorithm 1 initialise the arrays as described above. All the vertices of $T$ are traversed in the for-loop spanning Steps 7–21, one vertex at a time, except the root. The function $EffectiveLabel(Labels[i])$ returns a single value in Step 8. If $i$ is not a branch vertex, $Labels[i]$ will contain a single counter with a value of 1, in which case it returns the index of the counter. If $i$ is a branch vertex, Lemma 6 is used to determine the effective label, depending on the entries of $Labels[i]$. In Step 9 the vertex $i$ is added to the MSDS if its effective label is odd. Step 10 assigns a new label to the parent of vertex $i$. Steps 11–14 ensure that all vertices which caused the assignment of a 1-label to a branch vertex $i$ are included in the MSDS, if $A3Label[i]$ is TRUE. Similarly, all but one vertex that causes the assignment of a 1-label to a branch vertex $i$ is included.
Algorithm 1: DefendTree

\begin{algorithm}
\begin{algorithmic}[1]
\State \textbf{Input} : A tree $T$ represented by an array $\text{Parent}[1, \ldots, n]$. 
\State \textbf{Output}: An MSDS of $T$, represented by a boolean array $X$.
\For{$i \leftarrow 1$ to $n$} 
\State $\text{A3Label}[i] \leftarrow \text{FALSE}$ 
\State $X[i] \leftarrow \text{FALSE}$ 
\State $\text{Labels}[i] \leftarrow \{0, 0, 0, 0, 0, 0, 0\}$ 
\If{vertex $i$ is a branch vertex of $T$} \textbf{then} 
\State $\text{Branch}[i] \leftarrow \text{TRUE}$ 
\EndIf 
\State $\text{Previous1Label}[i] \leftarrow 0$
\EndFor 
\For{$i \leftarrow n$ down to $2$} 
\State $\ell \leftarrow \text{EffectiveLabel}($\text{Labels}[$i$]) 
\If{$\ell$ is odd} \textbf{then} 
\State $X[i] \leftarrow \text{TRUE}$ 
\EndIf 
\If{$\ell = 2$ and $\text{Branch}[$\text{Parent}[$i$]]} \textbf{then} 
\State $\text{A3Label}[$\text{Parent}[$i$]] \leftarrow \text{TRUE}$ 
\State $\text{prev} \leftarrow \text{Previous1Label}[$\text{Parent}[$i$]]$
\If{$\text{prev} > 0$} \textbf{then} 
\State $X[\text{prev}] \leftarrow \text{TRUE}$ 
\EndIf 
\EndIf 
\If{$\ell = 0$ and $\text{Branch}[$\text{Parent}[$i$]]} \textbf{then} 
\If{$\text{A3Label}[$\text{Parent}[$i$]]} \textbf{then} 
\State $X[i] \leftarrow \text{TRUE}$ 
\Else 
\State $\text{prev} \leftarrow \text{Previous1Label}[$\text{Parent}[$i$]]$
\If{$\text{prev} > 0$} \textbf{then} 
\State $X[\text{prev}] \leftarrow \text{TRUE}$ 
\EndIf 
\State $\text{Previous1Label}[$\text{Parent}[$i$]] \leftarrow i$
\EndElse 
\EndIf 
\State $\ell \leftarrow \text{EffectiveLabel}($\text{Labels}[$1$]) 
\If{$\ell$ is odd} \textbf{then} 
\State $X[1] \leftarrow \text{TRUE}$ 
\EndIf 
\If{$\text{Labels}[$1$, j$] = 0 for $j = 1, 3, 4, 5, 6$} \textbf{then} 
\State $X[1] \leftarrow \text{TRUE}$ 
\EndIf 
\If{$\text{Labels}[$1$, j$] = 0 for $j = 1, 3, 5, 6$ and $\text{Labels}[$1$, 0$] $\geq |\text{Labels}[$1$] - 1|$} \textbf{then} 
\State $X[1] \leftarrow \text{TRUE}$ 
\EndIf 
\If{$\ell = 3$} 
\State $\text{prev} \leftarrow \text{Previous1Label}[$1$]$
\If{$\text{prev} > 0$} \textbf{then} 
\State $X[\text{prev}] \leftarrow \text{TRUE}$ 
\EndIf 
\EndIf 
\EndFor 
\State \textbf{return} $[X]$ 
\end{algorithmic}
\end{algorithm}

in the MSDS, if $\text{A3Label}[i]$ is FALSE, which occurs in Steps 15–21. Finally Steps 22–28 determine whether or not the root forms part of the MSDS and ensure that all vertices which caused the assignment of a 1-label to the root are included in the MSDS.

The MSDS determined by means of Algorithm 1 for the tree $T_1$ of Figure 3.4 with canonical ordering as shown in Figure 5.1(a) is given in Figure 5.1(b). The six linear arrays maintained during execution of Algorithm 1 when applied to this tree as well as the function $\text{EffectiveLabel}$ are shown in Table 5.1.

We conclude this section with a result on the space and time complexities of Algorithm 1.

**Theorem 2 (Complexity of Algorithm 1)** If the input tree to Algorithm 1 is of order $n$, then both the space complexity and the worst-case time complexity of Algorithm 1 are $\mathcal{O}(n)$.

**Proof:** The function $\text{EffectiveLabel}[i]$ returns the value in the label list if $i$ is a non-branch vertex of $T$ or the value described in Lemma 6 if $i$ is a branch vertex of $T$. The use of an array of length 7 to store the values in the list $\text{Labels}[i]$ (and a similar approach towards storing the
characteristic vector of each pendent spider) ensures that the function $\text{EffectiveLabel}[i]$ can be performed in $O(1)$ time. Furthermore, each vertex of $T$, except the root, is considered exactly once in both the for-loops spanning Steps 1–6 and 7–21. Each operation in these for-loops can be performed in $O(1)$ time. Finally, the root is considered three times in Steps 22–28 and each operation in these three steps can be performed in $O(1)$ time.

A total of $7n$ memory units are required to store the array $\text{Labels}$, while $n$ memory units are required to store each of the five arrays $\text{Parent}$, $\text{Branch}$, $\text{Previous1Label}$, $\text{A3Label}$, $X$ and the function $\text{EffectiveLabel}$. Finally, three memory units are required to store the values of the variables $i$, $\ell$ and $\text{prev}$. The space complexity of Algorithm 1 is therefore $13n + 3 = O(n)$.

\section{Validation of the algorithm}

We implemented the algorithm presented in §5 in Wolfram’s Mathematica \cite{16} and validated the algorithm in five fundamentally different ways. First, we examined manually the output of the algorithm for a large number of randomly generated trees of varying orders and satisfied ourselves that the set $X$ returned by the algorithm indeed represented an SDS in each case. Secondly, we satisfied ourselves that the algorithm yielded MSDSs for a number of trees $T$ with special structure for which we know the value of $\gamma_s(T)$ (such as paths, stars, double stars and complete binary trees). Thirdly, we also verified that the value of $\gamma_s(T)$ returned by the algorithm did not violate any of the known theoretical bounds on the secure domination number of a graph. For example,

$$\frac{n}{1 + \Delta(G)} \leq \gamma_s(G) \leq n - M,$$

for any (connected) graph $G$ of order $n$ with maximum degree $\Delta(G)$ and matching number $M$ \cite{3}.

In addition, we verified that the algorithm yields the same value of the secure domination number $\gamma_s(T)$ for a tree $T$ when choosing different branch vertices as the root of $T$. We did this for randomly generated trees of different orders, choosing each branch vertex of each tree in turn as the root.

Finally, we ensured that the algorithm yielded the same value of $\gamma_s(T)$ for trees $T$ of small order as did the two exact algorithms (a branch-and-bound algorithm and a branch-and-reduce algorithm) for arbitrary graphs in \cite{4}. We did this for thirty randomly generated trees of orders 10, 15, 20, 25 and 30 each, noting the execution times of the three algorithms. These times are shown in Table 6.1. Although a much faster implementation of the algorithm in §5 will be possible when using a low-level programming language such as C or C++ instead of the high-level language
Table 5.1: The six linear arrays maintained during execution of Algorithm 1 as well as the output of the function EffectiveLabel when applied to the tree $T_1$ in Figure 5.1.

Mathematica, the times in Table 6.1 serve the purpose of estimating the benefit (in the form of a speed-up factor) of using a linear, purpose-designed algorithm for computing the secure domination number of a tree rather than an exponential algorithm for general graphs.

7 Conclusion

An algorithm was presented in this paper for computing the secure domination number of an arbitrary tree. After establishing a preliminary property of any SDS of a tree in §2, a method was presented for constructing an MSDS for a spider in §3. The algorithmic approach for secure domination of arbitrary trees in §4 entails including in the MSDS of $T$ those vertices required in an MSDS of a pendent spider of $T$, pruning away the pendent spider to form a smaller tree $T'$ and repeating this process until only a final spider remains. It was shown in §5 that this algorithmic approach may be implemented in linear space and time and explained in §6 how the algorithm was validated.

Acknowledgement

The research towards this paper was supported financially by the South African National Research Council under grant numbers 77248 (first author), 81558 (second author) and 70593 (third author).


<table>
<thead>
<tr>
<th>Trees of order</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 1 in §5</td>
<td>≤ 0.01</td>
<td>≤ 0.01</td>
<td>≤ 0.01</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
</tr>
<tr>
<td>Branch-and-bound algorithm in [4]</td>
<td>0.06</td>
<td>0.97</td>
<td>13.66</td>
<td>262.38</td>
<td>2692.03</td>
</tr>
<tr>
<td>Branch-and-reduce algorithm in [4]</td>
<td>0.05</td>
<td>0.93</td>
<td>14.16</td>
<td>263.54</td>
<td>2719.93</td>
</tr>
</tbody>
</table>

Table 6.1: Comparison of the execution times of three algorithms when computing the secure domination number of trees of small order. Times are measured in seconds and are the averages of the times required for 30 randomly generated instances of trees each order. All three algorithms were implemented in Wolfram’s Mathematica [16] on a 3.4 GHz Intel(R) Core(TM) i7-3770 processor with 8 GiB RAM running in Ubuntu 12.04 [6].

References