International Journal of Applied Mathematics

Volume 29 No. 4 2016, 401-423
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
doi: http://dx.doi.org/10.12732/ijam.v29i4.1

WEIGHTED DOMINATION NUMBER OF CACTUS GRAPHS

Tina Novak[1] §, Janez Žerovnik[2]

[1,2]Faculty of Mechanical Engineering
University of Ljubljana
Ljubljana, 1000, SLOVENIA

Abstract: In this paper we propose a linear algorithm for calculating the weighted domination number of a vertex-weighted cactus. The algorithm is based on the well known depth first search (DFS) structure. Our algorithm needs less than $12n + 5b$ additions and $9n + 2b$ min-operations where n is the number of vertices and b is the number of blocks in the cactus.

AMS Subject Classification: 05C69, 05C22, 05C85
Key Words: weighted domination problem, cactus graph, DFS structure

1. Introduction

The cactus graphs are interesting generalizations of trees, with numerous applications, for example in location theory [3, 17], communication networks [8, 18], stability analysis [1], and elsewhere. Usually, linear problems on trees imply linear problems on cacti. In this paper, we study the weighted domination number of a cactus graph with weighted vertices. It is well known that the problem of the weighted domination number on trees is linear, [5, 12]. Actually, we also have very general linear algorithm for computing domination-like problems on partial k-trees [13]. The time complexity of this algorithm is $\mathcal{O}(n|L|^{2k+1})$,

Received: May 5, 2016

§Correspondence author

where k is the treewidth and L is the set of vertex states (the different ways that a solution to a subproblem impact to the origin vertex). In the case of cactus graphs we have $k = 2$ and $|L| = 3$. Therefore, the time complexity of the general algorithm [13] on cacti is $\mathcal{O}(3^5 n)$.

It is well known that cactus graphs can be recognized by running an extended version of depth first search (DFS) algorithm that results a data structure of a cactus, see for example [16]. From the data structure, the vertices can be naturally divided into three types, i.e. each vertex either lies on a cycle and has degree 2 or lies on a cycle and has degree ≥ 3 or does not lie on a cycle (see [4]). Using this structure, we design an algorithm for general cacti. In this paper, we first illustrate the basic idea by writing a version of the algorithm for trees before generalizing the approach to arbitrary cactus graphs. Our algorithm has time complexity $\mathcal{O}(28n)$ which substantially improves the constant $3^5 = 243$. In fact, we will estimate time complexity of our algorithm more precisely (blocks will be formally defined later).

Theorem 1. *Let n be the number of vertices in a cactus and $b < n$ be the number of blocks. For computing the weighted domination number we need less than $12n + 5b$ additions and $9n + 2b$ min-operations.*

The rest of the paper is organized as follows. In the next section we first recall definitions of the domination number and the weighted domination number of general graphs. For cacti, we introduce the classification of vertices in relation to the skeleton structure, [4]. In Section 3 we define three parameters that are useful when considering the weighted domination problem. The simplified version of the algorithm that is used for computing the weighted domination number of a tree is presented in Section 4. Special cases of graphs, i.e. path-like graphs and cycle-like graphs are considered in Section 5. We write algorithms for calculating their weighted domination parameters and weighted domination number. In Section 6, the algorithm for general cacti is given and its time complexity is estimated.

2. Definitions and Preliminaries

2.1. A Vertex-Weighted Graph and the Weighted Domination Number

Let $G = (V, E)$ be a graph with a set of vertices $V = V(G)$ and a set of edges $E = E(G)$. Denote by $N(v)$ the open neighborhood of a vertex v, i.e. the set of vertices adjacent to the vertex v and by $N[v]$ the closed neighborhood of a vertex v: $N[v] = \{v\} \cup N(v)$. Let S be any subset of the set of vertices V. Denote by $N(S)$ the open neighborhood of the set S, i.e. the set of vertices adjacent to any vertex in S and similarly by $N[S]$ the closed neighborhood of S: $N[S] = S \cup N(S)$. A subset $D \subseteq V$ is a dominating set if $N[D] = V$. A domination number $\gamma(G)$ is the minimum cardinality among all dominating sets of the graph G.

In this article, a weighted graph (G, w) is a graph together with a positive real weight-function $w : V \to \mathbb{R}_+$. For the vertex $v_j \in V$ we shall write $w_j = w(v_j)$. The weight of a dominating set D is defined as $w(D) = \sum_{v_j \in D} w_j$. Finally, the weighted domination number (WDN) $\gamma_w(G)$ of the graph G is the minimum weight of a dominating set, more precisely

$$\gamma_w(G) = \min \left\{ w(D) \,\middle|\, D \text{ is a dominating set} \right\}. \tag{1}$$

2.2. Cactus Graph and its Skeleton

A graph $K = (V(K), E(K))$ is a cactus graph if and only if any two cycles of K have at most one vertex in common. Equivalently, any edge of a cactus lies on at most one cycle. Skeleton structure of a cactus is elaborated in [4], where it is shown that the vertices of a cactus graph are of three types:

- C-vertex is a vertex on a cycle of degree 2,
- G-vertex is a vertex not included in any cycle,
- H-vertex or a hinge is a vertex which is included in at least one cycle and is of degree ≥ 3.

By a *subtree* in a cactus we mean a tree induced by a subset of G-vertices and H-vertices only. A *graft* is a maximal subtree in a cactus. A subgraph of a cactus is called a *block* when it is either a cycle or a graft.

2.3. Depth First Search (DFS) Algorithm

The DFS is a well known method for exploring graphs. It can be used for recognizing cactus graphs providing the data structure (see [14], [16], [15], [17]). Let us consider a cactus graph K. We can distinguish one vertex as a root of K and denote it by r. After running the DFS algorithm, the vertices of K are DFS ordered. The order is given by the order in which DFS visits the vertices. (Note that the DFS order of a graph is not unique as we can use any vertex as the starting vertex (the root) and can visit the neighbors of a vertex in any order. However, here we can assume that the DFS order is given and is fixed.)

We denote by $\mathrm{DFN}(v)$ the position of v in the DFS order and we set $\mathrm{DFN}(r) = 0$. DFN is called the depth first number. Following [16] and [15], it is useful to store the information recorded during the DFS run in four arrays, called the DFS (cactus) data structure:

- $\mathrm{FATHER}(v)$ is the unique predecessor (father) of vertex v in the rooted tree, constructed with the DFS.

- $\mathrm{ROOT}(v)$ is the root vertex of the cycle containing v i.e. the first vertex of the cycle (containing v) in the DFS order. If v does not lie on a cycle, then $\mathrm{ROOT}(v) = v$. We set $\mathrm{ROOT}(r) = r$. (In any DFS order, if $\mathrm{DFN}(w) < \mathrm{DFN}(v)$ and w is the root of the cycle containing v and v is the root of another cycle (it is a hinge), then $\mathrm{ROOT}(v) = w$.)

- For vertices on a cycle (i.e. $\mathrm{ROOT}(v) \neq v$), orientation of the cycle is given by $\mathrm{ORIEN}(v) = z$, where z is the son of $\mathrm{ROOT}(v)$ that is visited on the cycle first. If $\mathrm{ROOT}(v) = v$, then $\mathrm{ORIEN}(v) = v$.

- $\mathrm{IND}(v) := |\{u \mid \mathrm{FATHER}(u) = v\}|$ is the number of sons of v in the DFS tree.

Below we write the pseudocode of the DFS algorithm that provides the data structure of cacti. The idea is taken from [14]. To mark a visited vertex in the procedure, we introduce auxiliary array MARK (as in [14]). At the beginning of the algorithm, we set $\mathrm{MARK}(v) = 0$ for every vertex in K. During the algorithm, whenever a vertex v is visited for the first time, the value $\mathrm{MARK}(v)$ becomes 1 and $\mathrm{DFN}(v)$ is increased by 1.

The direct correspondence of the definitions of C, G, H-vertices in a rooted cactus (K, r) and arrays FATHER, ROOT, ORIEN and IND is described in the following lemma.

Algorithm 1 DFS algorithm

Data: Rooted cactus (K, r) with vertices $V(K)$ and edges $E(K)$;
initialize
$i = 0$;
For every vertex v in K set
FATHER$(v) = v$; MARK$(v) = 0$; ROOT$(v) = v$;
ORIEN$(v) = v$; IND$(v) = 0$; DFN$(v) = 0$;
and for the root r reset:
MARK$(r) = 1$;
$v = r$;

Lemma 2 (C,G,H-vertices in DFS array)**.**

1. *For a vertex $v \neq r$ the following holds*:
   (a) *v is a C-vertex if and only if* ROOT$(v) \neq v$ *and* IND$(v) = 1$,
   (b) *v is a G-vertex if and only if* ROOT$(v) = v$ *and* ORIEN$(v) = v$ *and for every son u of v we have* ROOT$(u) \neq v$,
   (c) *v is a H-vertex if and only if either* (ROOT$(v) = v$ *and* ORIEN$(v) = v$ *and for at least one son u of v we have* ROOT$(u) = v$) *or* (ROOT$(v) \neq v$ *and* IND$(v) > 1$).
2. *For the root r we have*:
   (a) *r is a C-vertex if and only if* IND$(r) = 1$ *and for the son u of r* (DFN$(u) = 1$) *we have* ROOT$(u) = v$,
   (b) *r is a G-vertex if and only if for every son u of r we have* ROOT$(u) = u$,
   (c) *r is a H-vertex if and only if* IND$(r) > 1$ *and for at least one son u of r we have* ROOT$(u) = r$.

Remark 3. For any vertex $v \in V(K)$ and his father $w =$ FATHER(v), vertices with DFN's

$$\text{DFN}(w), \text{DFN}(w) + 1, \ldots, \text{DFN}(v) - 1$$

(and all corresponding edges induced by $V(K)$) form a rooted subcactus with the root w, denote it $(\widetilde{K}_w, w)$. The graphs $\widetilde{K}_w$ and $\{v\}$ are disjoint.

Algorithm 2 DFS algorithm - Part 2

repeat
  if all the edges incident to vertex v have already been labeled "examined" (v *is completely scaned*) **then**
    $v = \mathrm{FATHER}(v)$
  else *(an edge* (v, w) *is not labeled "examined")*
    The edge (v, w) label "examined" and **do** the following
    if $\mathrm{MARK}(w) = 0$ **then**
      $i = i + 1$;
      $\mathrm{DFN}(w) = i$;
      $\mathrm{MARK}(w) = 1$;
      $\mathrm{FATHER}(w) = v$;
      $\mathrm{IND}(w) = \mathrm{IND}(v) + 1$;
      $v = w$.
    else*(*$\mathrm{MARK}(w) = 1$*, that means we have a cycle)*
      label the edge (w, v) "examined";
      $\mathrm{ROOT}(v) = w$;
      $u = \mathrm{FATHER}(v)$;
      repeat *(assigning the root* w *of vertices of the cycle)*
        $z = u$;
        $\mathrm{ROOT}(z) = w$;
        $u = \mathrm{FATHER}(z)$;
      until $u = w$. *(now* z *determines the orientation of the cycle with the root* w*)*
      repeat *(assigning the successor* z *i.e. the orientation of vertices of the cycle)*
        $\mathrm{ORIEN}(v) = z$;
        $v = \mathrm{FATHER}(v)$;
      until $v = w$.
      $v = w$;
    end if
  end if
until $v = r$ and all edges incident to r are "examined"
Result: arrays FATHER, ROOT, ORIEN, IND, MARK, DFN.

Observation. Assume the last vertex l in the DFS order of a cactus K lies on a subtree T in K. Let $w = \mathrm{FATHER}(l)$ and w be the root (according to DFS order) of any subcactus $\widetilde{K}_w$, such that $\{l\} \cap V(\widetilde{K}_w) = \emptyset$. If $\widetilde{v} \in V(\widetilde{K}_w)$,

then $\mathrm{DFN}(w) \leq \mathrm{DFN}(\widetilde{v}) < \mathrm{DFN}(l)$. Similar but perhaps a little less obvious fact is given in the next proposition.

Proposition 4. *Assume that the last vertex l in DFS order of a cactus K lies on a cycle C. Then the following is true*:

1. *The neighboring vertex of l in the cycle C, which is not the father of the vertex l, is the root of the cycle C.*
2. *The vertex l is not a hinge.*
3. *Let $w, v \in C$, $w = \mathrm{FATHER}(v)$, w is not the root of the cycle C and w is a hinge, i.e. the root of a subcactus $\widetilde{K}$, such that $V(C) \cap V(\widetilde{K}) = w$. For any $\widetilde{v} \in V(\widetilde{K})$, we have $\mathrm{DFN}(w) \leq \mathrm{DFN}(\widetilde{v}) < \mathrm{DFN}(v)$.*

Proof. 1. Denote by v a neighboring vertex of l in the cycle C, which is not the father of l. If v is not the root of C, then $\mathrm{DFN}(v) > \mathrm{DFN}(l)$. Contradiction.

2. If l is a hinge, according to DFS order, there exists at least one vertex with $\mathrm{DFN} > \mathrm{DFN}(l)$. Contradiction.

3. According to DFS order, the inequality $\mathrm{DFN}(w) \leq \mathrm{DFN}(\widetilde{v})$ holds. Consider there is $\widetilde{v} \in \widetilde{K}$ with $\mathrm{DFN}(\widetilde{v}) > \mathrm{DFN}(v)$. Following the DFS algorithm, we have then $\mathrm{DFN}(\widetilde{v}) > \mathrm{DFN}(l)$. Contradiction.

□

3. Weighted Domination Parameters (WDP)

Let G be a graph and v any vertex in $V(G)$. Consider the following three parameters yielding related weighted domination parameters (defined in Chang [5]):

Definition 5.

1. $\gamma_w^{00}(G, v) = \min\{w(D) \mid D \text{ is a dominating set of } G - v\} = \\ = \gamma_w(G - v)$,
2. $\gamma_w^{1}(G, v) = \min\{w(D) \mid D \text{ is a dominating set of } G \text{ and } v \in D\}$,

3. $\gamma_w^0(G,v) = \min\{w(D) \mid D \text{ is a dominating set of } G \text{ and } v \notin D\}$.

It is obvious that

$$\gamma_w(G) = \min\left\{\gamma_w^1(G,v), \gamma_w^0(G,v)\right\}. \tag{2}$$

Since a dominating set of G, which does not contain the vertex v is also a dominating set of $G-v$, we have the relation

$$\gamma_w^{00}(G,v) \le \gamma_w^0(G,v). \tag{3}$$

Let D be a dominating set of $G-v$ such that $w(D) = \gamma_w(G-v)$. Then $D \cup \{v\}$ is a dominating set of G and clearly,

$$\gamma_w^1(G,v) \le w(v) + \gamma_w^{00}(G,v). \tag{4}$$

Lemma 6. *Let G_1 and G_2 be disjoined rooted graphs with roots v_1 and v_2 respectively, and let G be a disjoint union of G_1 and G_2 joined by the edge v_1v_2. Then the following is true*:

1. $\gamma_w^{00}(G,v_1) = \gamma_w^{00}(G_1,v_1) + \gamma_w(G_2)$,
2. $\gamma_w^1(G,v_1) = \gamma_w^1(G_1,v_1) + \min\left\{\gamma_w^1(G_2,v_2), \gamma_w^{00}(G_2,v_2)\right\}$,
3. $\gamma_w^0(G,v_1) = \min\left\{\gamma_w^0(G_1,v_1) + \gamma_w(G_2,v_2), \gamma_w^{00}(G_1,v_1) + \gamma_w^1(G_2,v_2)\right\}$.

The proof of Lemma 6 (for the domination number) appears in [5]. A generalization to weighted domination is straightforward and therefore omitted. A more general situation is described by the following lemma.

Lemma 7. *Let G_1 and G_2 be graphs with one common vertex v_0 and let $G_1 - v_0$ and $G_2 - v_0$ be disjoined. Denote by G the union of G_1 and G_2. Then we have*:

1. $\gamma_w^{00}(G,v_0) = \gamma_w^{00}(G_1,v_0) + \gamma_w^{00}(G_2,v_0)$,
2. $\gamma_w^1(G,v_0) = \gamma_w^1(G_1,v_0) + \gamma_w^1(G_2,v_0) - w(v_0)$,
3. $\gamma_w^0(G,v_0) = \min\left\{\gamma_w^0(G_1,v_0) + \gamma_w^{00}(G_2,v_0), \gamma_w^{00}(G_1,v_0) + \gamma_w^0(G_2,v_0)\right\}$.

Proof. 1. As $G_1 - v_0$ and $G_2 - v_0$ are disjoined, it follows $\gamma_w^{00}(G, v_0) = \gamma_w^{00}(G_1, v_0) + \gamma_w^{00}(G_2, v_0)$.

2. Let D be a dominating set of G with $v_0 \in D$ such that $w(D) = \gamma_w^1(G, v_0)$. Then $D_1 = D \cap V(G_1)$ is a dominating set of G_1 and $w(D_1) \geq \gamma_w^1(G_1, v_0)$. Similarly, $D_2 = D \cap V(G_2)$ is a dominating set of G_2 and $w(D_2) \geq \gamma_w^1(G_2, v_0)$. Hence $\gamma_w^1(G, v_0) \geq \gamma_w^1(G_1, v_0) + \gamma_w^1(G_2, v_0) - w(v_0)$. On the other hand, for any dominating sets D_1 and D_2 with $w(D_1) \geq \gamma_w^1(G_1, v_0)$ and $w(D_2) \geq \gamma_w^1(G_2, v_0)$, $D = D_1 \cup D_2$ dominates G. As $D_1 \cap D_2 = \{v_0\}$, we have $w(D) = w(D_1) + w(D_2) - w(v_0)$ and therefore $\gamma_w^1(G, v_0) \leq w(D) = \gamma_w^1(G_1, v_0) + \gamma_w^1(G_2, v_0) - w(v_0)$.

3. As we consider only dominating sets with $v_0 \notin D$, v_0 has to be dominated by some other vertex. We distinguish three cases: either v_0 is dominated by $D_1 = D \cap V(G_1)$, or $D_2 = D \cap V(G_2)$, or by both D_1 and D_2. Assuming $w(D) = \gamma_w^0(G, v_0)$, and recalling that $\gamma_w^{00}(G_i, v_0) \leq \gamma_w^0(G_i, v_0$ for $i = 1, 2$, it follows:

$$\begin{aligned}\gamma_w^0(G, v_0) \;=\; & \min\{\gamma_w^0(G_1, v_0) + \gamma_w^{00}(G_2, v_0),\\ & \gamma_w^{00}(G_1, v_0) + \gamma_w^0(G_2, v_0), \gamma_w^0(G_1, v_0) + \gamma_w^0(G_2, v_0)\}\\ \geq & \min\{\gamma_w^0(G_1, v_0) + \gamma_w^{00}(G_2, v_0), \gamma_w^{00}(G_1, v_0) + \gamma_w^0(G_2, v_0)\}.\end{aligned}$$

On the other hand, we can construct dominating sets of G by taking a union of two dominating sets D_1 and D_2 of G_1 and G_2 respectively. At least one of D_1, D_2 (or both) must dominate v_0. Taking either ($w(D_1) = \gamma_w^0(G_1, v_0)$ and $w(D_2) = \gamma_w^{00}(G_2, v_0)$) or ($w(D_1) = \gamma_w^{00}(G_1, v_0)$ and $w(D_2) = \gamma_w^0(G_2, v_0)$), we conclude that

$$\begin{aligned}\min\{\gamma_w^0(G_1, v_0) + \gamma_w^{00}(G_2, v_0), \gamma_w^{00}(G_1, v_0) + \gamma_w^0(G_2, v_0),\\ \gamma_w^0(G_1, v_0) + \gamma_w^0(G_2, v_0)\} \;\geq\\ \geq \min\{\gamma_w^0(G_1, v_0) + \gamma_w^{00}(G_2, v_0), \gamma_w^{00}(G_1, v_0) + \gamma_w^0(G_2, v_0)\} \;\geq\\ \geq \gamma_w^0(G, v_0).\end{aligned}$$

□

4. Algorithm for Trees

In this section, let $G = (V, E)$ be a vertex-weighted tree, and let T be an associated rooted tree with root r (r can be arbitrary but fixed vertex in $V(G)$).

In [12], the authors write the algorithm for calculating the weighted domination of a vertex-edge-weighted tree. It can of course be applied to a vertex-weighted tree, the case of interest in this paper. Another algorithm for calculating the weighted domination number of a tree appears in [5]. We write a new algorithm for weighted domination number of a weighted tree based on the DFS data structure here in order to illustrate the main idea on a well understood special case in order to clarify the development of the general algorithm in the following sections.

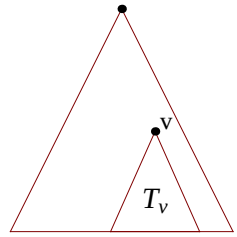

Figure 1: The rooted subtree

Denote by (T_v, v) the rooted subtree with the root v as is shown in Figure 1. In our algorithm we use the DFS order of vertices (i.e. the DFS cactus data structure provided by the DFS algorithm). We supplement the DFS data structure by four arrays of the initial values of the parameters γ_w^{00}, γ_w^1, γ_w^0 and γ_w. Initially, we set for every vertex v

$$\gamma_w^{00}(v) = 0, \quad \gamma_w^1 = w(v), \quad \gamma_w^0 = \infty \quad \text{and} \quad \gamma_w(v) = w(v)\,. \tag{5}$$

The algorithm's starting point is the last vertex v in the DFS order with the corresponding parameters $\gamma_w^{00}(v)$, $\gamma_w^1(v)$, $\gamma_w^0(v)$ and $\gamma_w(v)$. In the data structure we find the father of v and call it w. If $\mathrm{DFN}(w) \neq \mathrm{DFN}(v) - 1$ (i.e. $\mathrm{DFN}(w) < \mathrm{DFN}(v) - 1$), there exists rooted subtree $(\widetilde{T}_w, w)$ (see Remark 3). The algorithm calls itself recursively for the subtree $\widetilde{T}_w$ and then accordingly updates the parameters $\gamma_w^{00}(w) = \gamma_w^{00}(\widetilde{T}_w, w)$, $\gamma_w^1(w) = \gamma_w^1(\widetilde{T}_w, w)$, $\gamma_w^0(w) = \gamma_w^0(\widetilde{T}_w, w)$ and $\gamma_w(w) = \gamma_w(\widetilde{T}_w)$. When w and v are the last two vertices in the DFS order, the parameters at w are computed according to Lemma 6, and the computation continues regarding w as the last vertex. For pseudocode of the algorithm see Algorithm 3.

Proposition 8 (Time complexity of TREE)**.** *Algorithm TREE needs* $4(n-1)$ *additions and* $3(n-1)$ min-*operations.*

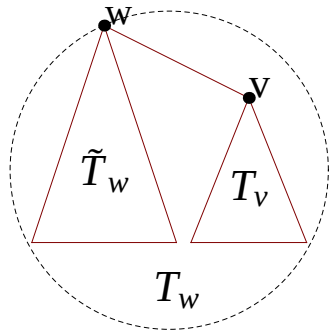

Figure 2: Subtrees T_v, $\widetilde{T}_w$ and T_w

Algorithm 3 TREE

Data: A rooted tree (T, r) with DFS ordered vertices in the DFS table
initialize $\gamma_w^{00}(v) = 0$, $\gamma_w^1(v) = w(v)$, $\gamma_w^0(v) = \infty$ and $\gamma_w(v) = w(v)$ for every vertex v in the DFS table
set v is the last vertex in the DFS order;
repeat
  $w = \mathrm{FATHER}(v)$;
  if $\mathrm{DFN}(w) \neq \mathrm{DFN}(v) - 1$ **then**
    call algorithm **TREE** for the rooted tree on vertices with $\mathrm{DFN} = \mathrm{DFN}(w), \ldots, \mathrm{DFN}(v) - 1$ and the root w *(we obtain new values for* $\gamma_w^{00}(w)$, $\gamma_w^1(w)$, $\gamma_w^0(w)$ *and* $\gamma_w(w)$*)*;
  end if
  $\gamma_w^{00}(w) = \gamma_w^{00}(w) + \gamma_w(v)$;
  $\gamma_w^1(w) = \gamma_w^1(w) + \min\{\gamma_w^1(v), \gamma_w^{00}(v)\}$;
  $\gamma_w^0(w) = \min\{\gamma_w^0(w) + \gamma_w(v), \gamma_w^{00}(w) + \gamma_w^1(v)\}$;
  $\gamma_w(w) = \min\{\gamma_w^1(w), \gamma_w^0(w)\}$;
  $v = w$;
until $v = r$
Result: $\gamma_w^*(T, r) = \gamma_w^*(v)$ for $* = 00, 1, 0$;
  $\gamma_w(T) = \gamma_w(v)$.

Proof. Using Lemma 6 and the equation

$$\gamma_w(T_w) = \min\{\gamma_w^1(T_w, w), \gamma_w^0(T_w, w)\}$$

in a step of the algorithm for rooted subtrees (T_w, w) and (T_v, v) (where $w =$ FATHER(v)), the calculation demands 4 additions and 3 min-operations. The algorithm sticks rooted subtrees (T_w, w) and (T_v, v) for every existing edge (w, v). □

5. Some More Basic Algorithms

The algorithm for cactus graph should exploit the tree structure obtained from DFS representation. It would be meaningful to preserve the form of algorithm TREE if the current vertex of a cactus lies on a tree. Special attention should be paid to the current vertex on a cycle. In this section we prepare subalgorithm CYCLE-LIKE for the rooted cycle (C, r), which calculates parameters $\gamma_w^{00}(C, r)$, $\gamma_w^1(C, r)$, $\gamma_w^0(C, r)$ and $\gamma_w(C)$.

5.1. Path-Like Cactus

Let $\{v_1, \ldots, v_n\}$ be a path and $(G_1, v_1), \ldots, (G_n, v_n)$ disjoined rooted graphs as is shown in Figure 3. Denote the obtained graph by G and consider it as a rooted graph (G, v_n).

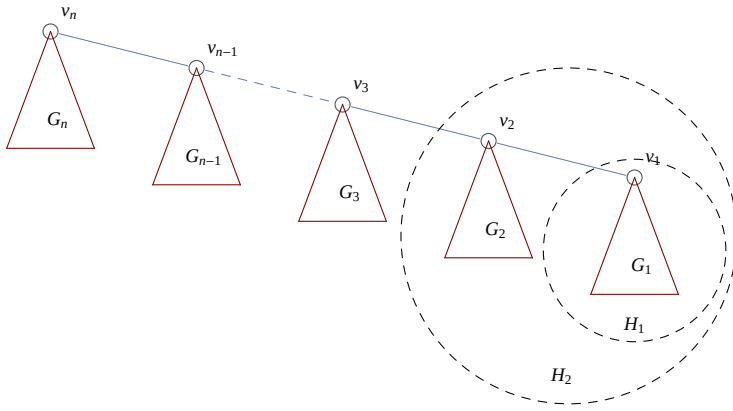

Figure 3: Path-like cactus

Lemma 9. *Let $G_1, G_2, \ldots, G_n$ be disjoined graphs with specific vertices $v_1, v_2, \ldots, v_n$ respectively and let G be the disjoint union of $G_1, G_2, \ldots, G_n$, joined by the edges $v_1v_2, v_2v_3, \ldots, v_{n-1}v_n$. For every $i \in \{1,...,n\}$, denote by*

Algorithm 4 PATH-LIKE

Data: a path-like cactus (P, r) with the DFS ordered path's vertices and corresponding parameters γ_w^{00}, γ_w^1, γ_w^0 and γ_w (i.e. WDP and WDN of rooted subgraphs (G_i, v_i) as is shown in Figure 3)
set v is the last vertex in the DFS order;
repeat
$w = \text{FATHER}(v)$;
$\gamma_w^{00}(w) = \gamma_w^{00}(w) + \gamma_w(v)$;
$\gamma_w^1(w) = \gamma_w^1(w) + \min\{\gamma_w^1(v), \gamma_w^{00}(v)\}$;
$\gamma_w^0(w) = \min\{\gamma_w^0(w) + \gamma_w(v), \gamma_w^{00}(w) + \gamma_w^1(v)\}$;
$\gamma_w(w) = \min\{\gamma_w^1(w), \gamma_w^0(w)\}$;
$v = w$;
until $v = r$.
Result: $\gamma_w^*(P, r) = \gamma_w^*(v) \quad \text{for} \quad * = 00, 1, 0$;
$\gamma_w(P) = \gamma_w(v)$.

H_i the union of graphs G_1,...,G_i, i.e. $H_i = \left(\bigcup_{j=1}^{i} G_j\right) \cup \left(\bigcup_{j=1}^{i-1}(v_j, v_{j+1})\right)$. If $i > 1$, the following is true:

$$\gamma_w^{00}(H_i, v_i) = \gamma_w^{00}(G_i, v_i) + \gamma_w(H_{i-1}), \tag{6}$$

$$\gamma_w^1(H_i, v_i) = \gamma_w^1(G_i, v_i) + \min\left\{\gamma_w^1(H_{i-1}, v_{i-1}), \gamma_w^{00}(H_{i-1}, v_{i-1})\right\}, \tag{7}$$

$$\gamma_w^0(H_i, v_i) = \min\left\{\gamma_w^0(G_i, v_i) + \gamma_w(H_{i-1}), \gamma_w^{00}(G_i, v_i) + \gamma_w^1(H_{i-1}, v_{i-1})\right\}, \tag{8}$$

$$\gamma_w(H_i) = \min\left\{\gamma_w^1(H_i, v_i), \gamma_w^0(H_i, v_i)\right\}. \tag{9}$$

Proof. Look at the graph H_i as the disjoint union of subgraphs G_i and H_{i-1} with roots v_i and v_{i-1} respectively and joined by the edge $v_{i-1}v_i$. These are exactly the assumptions of Lemma 6. □

Proposition 10 (Time complexity of PATH-LIKE)**.** *Algorithm PATH-LIKE needs* $4(n-1)$ *additions and* $3(n-1)$ min*-operations.*

Proof. By counting all operations in (6), (7), (8) and (9), the proposition follows. □

5.2. D-Closed Path-Like Cactus

Let $\{v_1, \ldots, v_n\}$ be a path and $(G_1, v_1), \ldots, (G_n, v_n)$ disjoined rooted graphs. We require that both v_1 and v_n are members of a dominating set. Such a graph G is drawn on Figure 4.

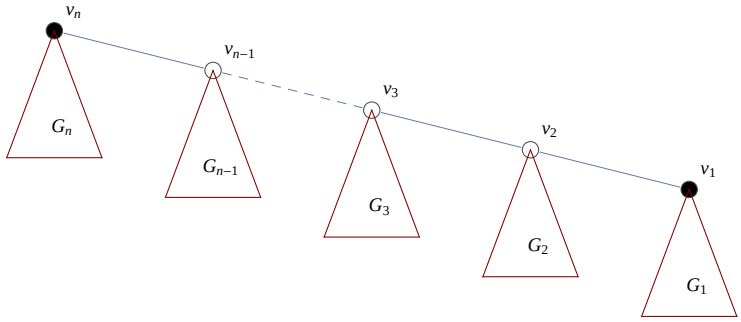

Figure 4: D-closed path-like cactus

To calculate the weighted domination parameters and the weighted domination number with the condition that $v_1 \in D$, we introduce some additional notation (for $u \neq v_1$):

$$\begin{aligned}
\gamma_{w,v_1}(G) &= \gamma^1_w(G, v_1) \\
\gamma^{00}_{w,v_1}(G, u) &= \gamma_{w,v_1}(G - u) \\
\gamma^1_{w,v_1}(G, u) &= \min\left\{w(D) \,\middle|\, \{u, v_1\} \subseteq D\right\} \\
\gamma^0_{w,v_1}(G, u) &= \min\left\{w(D) \,\middle|\, v_1 \in D, u \notin D\right\}.
\end{aligned}$$

In the new algorithm for calculating the WDN of a D-closed path-like cactus we have to provide that the first vertex (v_1) is a member of a dominating set. We apply algorithm PATH-LIKE and make changes in the first step of the loop **repeat-until**. According to Figure 3 and Figure 4, this means

$$\gamma^{00}_{w,v_1}(H_2, v_2) = \gamma^{00}_w(G_2, v_2) + \gamma^1_w(H_1, v_1) \tag{10}$$

$$\gamma^1_{w,v_1}(H_2, v_2) = \gamma^1_w(G_2, v_2) + \gamma^1_w(H_1, v_1) \tag{11}$$

$$\gamma^0_{w,v_1}(H_2, v_2) = \gamma^{00}_w(G_2, v_2) + \gamma^1_w(H_1, v_1). \tag{12}$$

The other steps do not need corrections and the vertex v_1 on Figure 4 (the last vertex in the DFS order in the algorithm below) on a path remains in a dominating set.

Algorithm 5 *D*-CLOSED PATH-LIKE

Data: *D*-closed path-like cactus (P, r): with DFS ordered path's vertices and corresponding parameters γ_w^{00}, γ_w^1, γ_w^0 and γ_w, i.e. WDP and WDN of rooted subgraphs (G_i, v_i)
set v is the last vertex in DFS order;
$l = v$;
$w = \text{FATHER}(v)$;
$\gamma_w^{00}(w) = \gamma_w^{00}(w) + \gamma_w^1(v)$;
$\gamma_w^1(w) = \gamma_w^1(w) + \gamma_w^1(v)$;
$\gamma_w^0(w) = \gamma_w^{00}(w) + \gamma_w^1(v)$;
$\gamma_w(w) = \min\{\gamma_w^1(w), \gamma_w^0(w)\}$;
$v = w$;
repeat
  $w = \text{FATHER}(v)$;
  $\gamma_w^{00}(w) = \gamma_w^{00}(w) + \gamma_w(v)$;
  $\gamma_w^1(w) = \gamma_w^1(w) + \min\{\gamma_w^1(v), \gamma_w^{00}(v)\}$;
  $\gamma_w^0(w) = \min\{\gamma_w^0(w) + \gamma_w(v), \gamma_w^{00}(w) + \gamma_w^1(v)\}$;
  $\gamma_w(w) = \min\{\gamma_w^1(w), \gamma_w^0(w)\}$;
  $v = w$;
until $v = r$.
Result: $\gamma_{w,l}^*(P, r) = \gamma_w^*(v) \quad \text{for} \quad * = 00, 1, 0;$
$\gamma_{w,l}(P) = \gamma_w(v)$.

Proposition 11 (Time complexity of *D*-CLOSED PATH-LIKE)**.** *Algorithm D-CLOSED PATH-LIKE needs less than* $4(n-1)$ *additions and* $3(n-1)$ min-*operations.*

Proof. In the first step of the algorithm we have 3 additions and one min-operation. For the loop we need $4(n-2)$ additions and $3(n-2)$ min-operations. □

5.3. Cycle-Like Cactus

We now consider the case when the specific vertices $v_1, \ldots, v_n$ in a graph are vertices of a cycle. Let (C_n, v_n) be a rooted cycle with vertices $v_1, \ldots, v_n$ and corresponding weights $w_1, \ldots, w_n$, and let $(G_1, v_1), \ldots, (G_{n-1}, v_{n-1})$ be disjoined rooted graphs (in our case cacti). Denote by (K_n, v_n) the union of $(G_1, v_1), \ldots$ $\ldots, (G_{n-1}, v_{n-1})$ and v_n, joined by the edges $v_1v_2, v_2v_3, \ldots, v_{n-1}v_n$ and v_nv_1.

Graph (K_n, v_n) is depicted in Figure 5.

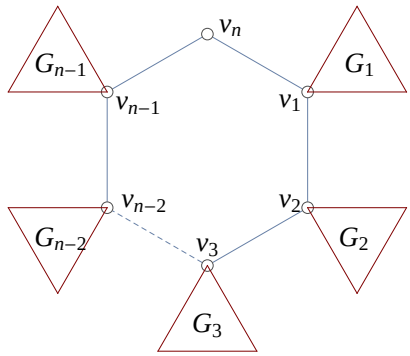

Figure 5: Cycle-like cactus

Let (K'_n, v_n) be the path-like cactus with specific vertices on the weighted path $\{v'_n, v_1, \ldots, v_n\}$, where we additionally define $w(v'_n) = w_n$. Graph (K'_n, v_n) is obtained from (K_n, v_n) as is shown in Figure 6.

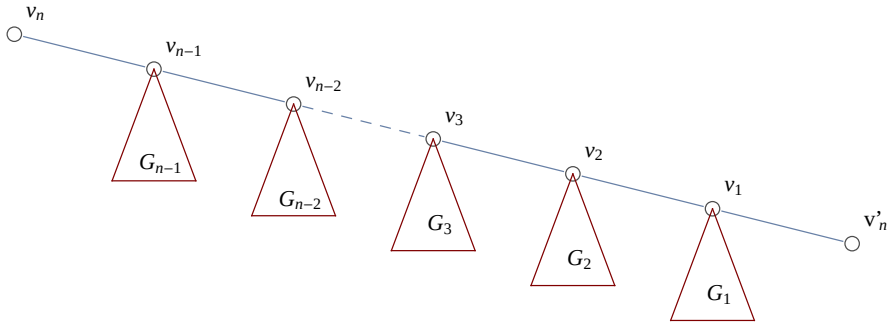

Figure 6: Graph (K'_n, v_n) obtained from (K_n, v_n)

Lemma 12. *For a rooted cycle-like cactus (K_n, v_n), the weighted domination parameters and the weighted domination number are the following*:

$$\gamma_w^{00}(K_n, v_n) = \gamma_w(K_n - v_n), \tag{13}$$

$$\gamma_w^{1}(K_n, v_n) = \gamma_{w,v'_n}^{1}(K'_n, v_n) - w_n, \tag{14}$$

$$\gamma_w^{0}(K_n, v_n) = \min\left\{\gamma_w^{1}(K_n - v_n, v_1), \gamma_w^{1}(K_n - v_n, v_{n-1})\right\}, \tag{15}$$

$$\gamma_w(K_n) = \min\left\{\gamma_w^{1}(K_n, v_n), \gamma_w^{0}(K_n, v_n)\right\}. \tag{16}$$

Algorithm 6 CYCLE-LIKE

Data: a rooted cycle (C, r): with DFS ordered vertices and corresponding parameters γ_w^{00}, γ_w^1, γ_w^0 and γ_w, i.e. WDP and WDN of rooted subgraphs (G_i, v_i)

set v is the last vertex in DFS order;
r is the first vertex in DFS order;
$s = \text{ORIEN}(v)$;
$l = v$;

Algorithm 7 CYCLE-LIKE - Part 2

set new vertex r' in the DFS table with:
$\text{DFN}(r') = \text{DFN}(v) + 1$;
$\text{FATHER}(r') = v$;
$(\text{MARK}(r') = 1, \text{ROOT}(r') = r', \text{ORIEN}(r') = r',$
$\text{IND}(r') = 1)$;
$\gamma_w^{00}(r') = \gamma_w^{00}(r)$, $\gamma_w^1(r') = \gamma_w^1(r)$, $\gamma_w^0(r') = \gamma_w^0(r)$ and
$\gamma_w(r') = \gamma_w(r)$.

calculate

- $\gamma_w(C - r)$ and $\gamma_w^1(C - r, s)$ using PATH-LIKE on DFS ordered path's vertices $\{s, \ldots, v\}$
- $\gamma_{w,r'}^1(C \cup \{r'\}, r)$ using D-CLOSED PATH-LIKE on DFS ordered path's vertices $\{r, s, \ldots, v, r'\}$
- $\gamma_{w,v}(C - r)$ using D-CLOSED PATH-LIKE ($v \in D$) on DFS ordered path's vertices $\{s, \ldots, v\}$

Result: $\gamma_w^{00}(C, r) = \gamma_w(C - r)$;
$\gamma_w^1(C, r) = \gamma_{w,r'}^1(C \cup \{r'\}, r) - \gamma_w^1(r)$;
$\gamma_w^0(C, r) = \min\left\{\gamma_w^1(C - r, s), \gamma_{w,v}(C - r)\right\}$;
$\gamma_w(C) = \min\left\{\gamma_w^1(C, r), \gamma_w^0(C, r)\right\}$.

Proof. The lemma is a direct consequence of the construction of the graph K', the definition of $\gamma_{w,v_n'}^1$ and the properties of the parameters γ_w^0 and γ_w^{00}.

Recall that the weighted domination parameters $\gamma_w^{00}(K_n, v_n)$, $\gamma_w^1(K_n, v_n)$ and $\gamma_w^0(K_n, v_n)$ can be calculated using the previous two lemmas. □

Proposition 13 (Time complexity of CYCLE-LIKE)**.**

If a cycle C has n vertices, the algorithm CYCLE-LIKE needs less than $12(n-1)$ *additions and* $9(n-1)$ min*-operations.*

Proof. Using algorithms PATH-LIKE and D-CLOSED PATH-LIKE we obtain:

- For calculating the parameters $\gamma_w(C-r)$ and $\gamma_w^1(C-r,s)$ using PATH-LIKE algorithm on $n-1$ vertices, we need $4(n-2)$ additions and $3(n-2)$ min-operations.
- For calculating the parameter $\overline{\gamma}_w^1(C\cup\{r'\},r)$ using D-CLOSED PATH-LIKE algorithm on $n+1$ vertices, we need $4n$ additions and $3n$ min-operations.
- For calculating $\overline{\gamma}_w(C-r)$ using D-CLOSED PATH-LIKE algorithm on $n-1$ vertices, we need $4(n-2)$ additions and $3(n-2)$ min-operations.

Additionally, at the end of the algorithm, we need one addition and two min-operations. Adding up all operations, we confirm

$$4(n-2)+4n+4(n-2)+1=12n-15<2(n-1) \tag{17}$$

additions, and

$$3(n-2)+3n+3(n-2)+2=9n-10<9(n-1) \tag{18}$$

min-operations. □

6. Algorithm for Weighted Domination of Cacti

As we indicated in the previous sections, the general algorithm for calculating WDN of a cactus graph can be seen as an upgrade of the algorithm TREE. The input data of the main algorithm is the DFS cactus data structure, which is supplemented by four arrays of the initial values of the parameters γ_w^{00}, γ_w^1, γ_w^0 and γ_w for every vertex. The starting point (vertex) of the algorithm is the last unread vertex in the DFS order. If the last vertex lies on a tree, we proceed like in the algorithm TREE. (Some care must be taken for the root of the tree, to correct its parameters following Lemma 7.) However, if the last vertex lies on a cycle, we have to read and remember all cycle's vertices. Following Remark 3 and Proposition 4, the algorithm calls itself for rooted subcacti, for which the

Algorithm 8 CACTUS

Data: A rooted cactus (K, r) with DFS ordered vertices in the DFS table;
initialize $\gamma_w^{00}(v) = 0$, $\gamma_w^1(v) = w(v)$, $\gamma_w^0(v) = \infty$ and $\gamma_w(v) = w(v)$ for every vertex v in the DFS table;
set v is the last vertex in the DFS order
    Last $= v$;
while Last $\neq r$ **do**
  if Last does not lie on a cycle **then**
    repeat
      $w = \mathrm{FATHER}(v)$;
      $u = v$;
      if $(\mathrm{ROOT}(w) = w)$ and $(\mathrm{DFN}(w) < \mathrm{DFN}(v) - 1)$ **then**
        do CACTUS of the rooted subcactus on vertices in the DFS table with $\mathrm{DFN} = \mathrm{DFN}(w), \ldots, \mathrm{DFN}(v) - 1$ and the root w *(we get new values* $\gamma_w^{00}(w)$, $\gamma_w^1(w)$, $\gamma_w^0(w)$, $\gamma_w(w)$*)*
      end if
      $\gamma_w^{00}(w) = \gamma_w^{00}(w) + \gamma_w(v)$;
      $\gamma_w^1(w) = \gamma_w^1(w) + \min\{\gamma_w^1(v), \gamma_w^{00}(v)\}$;
      $\gamma_w^0(w) = \min\{\gamma_w^0(w) + \gamma_w(v), \gamma_w^{00}(w) + \gamma_w^1(v)\}$;
      $\gamma_w(w) = \min\{\gamma_w^1(w), \gamma_w^0(w)\}$;
      $v = w$;
    until $(\mathrm{ROOT}(v) \neq v)$ or $(\mathrm{DFN}(v) = 0)$
  else (see next page)
  end if

roots are hinges of the cycle. This forces the hinges to obtain new values of parameters where all subcacti rooted at the hinges are considered. Then the algorithm calls subalgorithm CYCLE-LIKE and applies Lemma 7 to correct the values of the parameters of the root of cycle. The algorithm continues until the last unread vertex in the DFS order is the root of the cactus. Pseudocode is given below (Algorithm 8).

Denote by b the number of blocks, i.e. the total number of cycles and grafts.

Proposition 14. *Algorithm CACTUS properly calculates the weighted domination number of a cactus.*

Proof. Recall that by definition we have two essential situations. If the

Algorithm 9 CACTUS - Part 2

if Last does not lie on a cycle **then** see previous page
else
$v = \text{Last}$;
repeat *(mark the roots of the cycle and correct WDP and WDN arrays of hinges on a cycle)*
$w = \text{FATHER}(v)$;
if $\text{DFN}(w) < \text{DFN}(v) - 1$ **then**
do CACTUS of rooted subcactus on vertices in the DFS table with $\text{DFN} = \text{DFN}(w), \ldots, \text{DFN}(v) - 1$ and the root w *(new values* $\gamma^*_w(w)$, $* = 00, 1, 0$ *and* $\gamma_w(w)$*)*
end if
$v = w$;
until $v = \text{ORIEN}(v)$.
$u = v$;
$w = \text{FATHER}(v)$; *(w is now the root of the cycle)*
Make cycle table:
$C = \emptyset$;
$v = \text{Last}$;
$C = DFS(v)$;
repeat
$v = \text{FATHER}(v)$;
$C \cup DFS(v)$;
until $v = w$.
Do CYCLE-LIKE algorithm on the rooted cycle (C, v) with vertices in the table C. We obtain parameters of the cycle C: $\gamma^{00}_w(C, v)$, $\gamma^1_w(C, v)$, $\gamma^0_w(C, v)$ and $\gamma_w(C)$;
$\gamma^{00}_w(w) = \gamma^{00}_w(w) + \gamma^{00}_w(C, v)$;
$\gamma^1_w(w) = \gamma^1_w(w) + \gamma^1_w(C, v) - \text{WEIGHT}(w)$;
$\gamma^0_w(w) = \min\{\gamma^0_w(w) + \gamma^{00}_w(C, v), \gamma^{00}_w(w) + \gamma^0_w(C, v)\}$;
$\gamma_w(w) = \min\{\gamma^1_w(w), \gamma^0_w(w)\}$;
end if
Last is determined by $\text{DFN}(\text{Last}) = \text{DFN}(u) - 1$.
Result: $\gamma^*_w(K, r) = \gamma^*_w(w)$ for $* = 00, 1, 0$;
$\gamma_w(K) = \gamma_w(w)$.

current vertex v is a G-vertex on a subtree or a root of a cycle, the algorithm

CACTUS calculates the WDP and the WDN of the subcactus of all vertices with DFN $\geq$ DFN(v). In particular, when $v = r$ the algorithm CACTUS calculates the WDP and the WDN of the given cactus and we have

$$\gamma_w(K) = \gamma_w(r)\,. \tag{19}$$

□

Below we show that algorithm CACTUS needs less than $12n + 5b$ additions and $9n + 2b$ min operations and thus prove Theorem 1.

Proof. (of Theorem 1.) Let $B_1, \ldots, B_b$ be blocks in the cactus and denote by n_j the number of vertices in the block B_j for each $j = 1, \ldots, b$. Since a hinge can be the root of more than one block, the number of hinges is less or equal b. Therefore, we have the inequality

$$n_1 + n_2 + \ldots + n_b - b \leq n\,. \tag{20}$$

Since the algorithm requires much more time for a cycle block (in comparison with a graft), we can estimate that for each block B_j we need less than $12(n_j - 1)$ additions and $9(n_j - 1)$ min-operations. Furthermore, 5 additions and 2 min-operations are needed for sticking blocks in a hinge. Summing up all operations for all blocks in the cactus, we get

$$\sum_{j=1}^{b} 12(n_j - 1) + 5b = 12(\sum_{j=1}^{b} n_j - b) + 5b \leq 12n + 5b$$

and

$$\sum_{j=1}^{b} 9(n_j - 1) + 2b \leq 9n + 2b\,.$$

□

Remark 15. In the proof above, we have assumed that the DFS data structure of the cactus is given. The reason is that the algorithm for k-trees [13] assumes the structural information of a partial k-tree is given. It is however well-known that the DFS algorithm is linear in the number of edges of a graph, which for trees and cactus graphs implies that it is also linear in the number of vertices. More precisely, $4m$ operations are needed when traversing the graph during DFS that provides the DFS cactus data structure: the DFS search has to be followed by a traversal in the opposite DFS order and, in addition. each cycle has to be traversed two more times to assign the roots and the successors to all vertices of a cycle.

Acknowledgments

This work was supported in part by Slovenian Research Agency ARRS (Grant number P1-0285-0101).

References

[1] M. Arcak, Diagonal stability on cactus graphs and application to network stability analysis, *IEEE Trans. Autom. Control*, **56** (2011), 2766-2777.

[2] N. Betzler, R. Niedermeier and J. Uhlmann, Tree decompositions of graphs: Saving memory in dynamic programming, *Discrete Optimization*, **3** (2006), 220-229.

[3] B.-M. Boaz, B. Binay and S. Qiaosheng, Efficient algorithms for the weighted 2-center problem in a cactus graph, *Algorithms and Computation, 16th Int. Symp., ISAAC 2005 Lecture Notes in Computer Science*, **3827** (2005), 693-703; doi:10.1007/11602613_70.

[4] R.E. Burkard and J. Krarup, A linear algorithm for the Pos/Neg-Weighted 1-Median problem on a cactus, *Computing*, **60** (1998), 193-215.

[5] G.J. Chang, Algorithmic aspects of domination in graphs, In: *Handbook of Combinatorial Optmization* (2013), 221-282.

[6] E. Cockayne, S. Goodman and S. Hedetniemi, A linear algoithm for the domination number of a tree, *Information Processing Letters*, **4** (1975), 41-44.

[7] P. Dankelmann, D. Rautenbach and L. Volkmann, Weighted domination in triangle-free graphs, *Discrete Mathematics*, **250** (2002), 233-239.

[8] B. Elenbogen and J.F. Fink, Distance distributions for graphs modeling computer networks, *Discrete Applied Mathematics*, **155** (2007), 2612-2624.

[9] S.T. Hedetniemi and R.C. Laskar, Bibliography and domination in graphs and some basic definitions of domination parameters, *Discrete Mathematics*, **86** (1990), 257-277.

[10] S.T. Hedetniemi, R. Laskar and J. Pfaff, A linear algorithm for finding a minimum dominating set in a cactus, *Discrete Applied Mathematics*, **13** (1986), 287-292.

[11] Y. Lan, Y. Wang and H. Suzuki, A linear-time algorithm for solving the center problem on weighted cactus graphs, *Information Processing Letters*, **71** (1999), 205-212.

[12] K.S. Natarajan and L.J. White, Optimum domination in weighted trees, *Information Processing Letters*, **7** (1987), 261-265.

[13] J.A. Telle and A. Proskurowski, Practical algorithms on partial k-trees with an application to domination-like problems, *Lecture Notes in Computer Science*, **709** (1993), 610-621.

[14] K. Thulasiraman and M.N.S. Swamy, *Graphs: Theory and Algorithms*, John Wiley & Sons, Inc. (1992).

[15] B. Zmazek and J. Žerovnik J, The absolute center problem on weighted cactus graph, In: *SOR '01 Proceedings Lenart, Zadnik Stirn and Drobne Ljubljana: Slovenian Society Informatika, Section for Operational Research*, (2001), 189-194.

[16] B. Zmazek and J. Žerovnik, Computing the weighted Wiener and Szeged number on weighted cactus graphs in linear time, *Croat. Chem. Acta*, **76** (2003), 137-143.

[17] B. Zmazek and J. Žerovnik, The obnoxious center problem in weighted cactus graphs, *Discrete Applied Mathematics*, **136** (2004), 377-386.

[18] B. Zmazek B and J. Žerovnik, Estimating the traffic on weighted cactus networks in linear time, In: *Ninth International Conference on Information Visualisation (IV'05)* (2005), 536-541; doi:10.1109/IV.2005.48.