AUSTRALASIAN JOURNAL OF COMBINATORICS
Volume **36** (2006), Pages 101–111

Algorithmic aspects of minus total k-subdomination in graphs

Laura Harris

School of Mathematics, Statistics, & Information Technology
University of KwaZulu-Natal
Private Bag X01, Pietermaritzburg, 3209
South Africa

Johannes H. Hattingh

Department of Mathematics and Statistics
Georgia State University
Atlanta, Georgia 30303-3083
U.S.A.

Michael A. Henning[*]

School of Mathematics, Statistics, & Information Technology
University of KwaZulu-Natal
Private Bag X01, Pietermaritzburg, 3209
South Africa

Abstract

Let $G = (V, E)$ be a graph and let $k \in \mathbf{Z}^+$. A minus total k-subdominating function (mTkSF) is a function $f\colon V \to \{-1, 0, 1\}$ such that for at least k vertices v of G, the sum of the function values of f in the open neighborhood of v is positive. The minus total k-subdomination number of G is the minimum value of $f(V)$ over all mTkSF s f of G where $f(V)$ denotes the sum of the function values assigned to the vertices under f. In this paper, we show that the associated decision problem is **NP**-complete for bipartite graphs and also present cubic time algorithms to compute the minus total k-subdomination and minus k-subdomination numbers of a tree.

[*] Research supported in part by the University of KwaZulu-Natal and the National Research Foundation.

1 Introduction

Our mathematical model is a finite, simple graph $G = (V, E)$ with vertex set V and edge set E of order $n(G) = |V|$ and size $m(G) = |E|$.

The *open neighborhood* of a vertex v is $N(v) = \{u \mid uv \in E\}$. The *closed neighborhood* of v is $N[v] = N(v) \cup \{v\}$. A *minus dominating function* is defined in [2] as a function $f : V \to \{-1, 0, 1\}$ such that $f(N[v]) \geq 1$ for every $v \in V$. The *minus domination number* of a graph G is $\gamma^-(G) = \min\{f(V) \mid f$ is a minus dominating function on $G\}$. A *minus k-subdomination function* (mkSF) for G is defined in [1] as a function $f : V \to \{-1, 0, 1\}$ such that $f(N[v]) \geq 1$ for at least k vertices of G. The *minus k-subdomination number* of a graph G, denoted by $\gamma^-_{ks}(G)$, is equal to $\min\{f(V) \mid f$ is a mkSF of $G\}$.

An analogous theory for minus total k-subdominating functions arises when "closed" neighborhood in the definition of a minus k-subdominating function is changed to "open" neighborhood. A *minus total k-subdomination function* (mTkSF) for G is defined as a function $f : V \to \{-1, 0, 1\}$ such that $f(N(v)) \geq 1$ for at least k vertices of G. The *minus total k-subdomination number* of a graph G, denoted by $\gamma^-_{tks}(G)$, is equal to $\min\{f(V) \mid f$ is a mTkSF of $G\}$. In the special case where $k = |V|$, the minus total k-subdomination number is the *minus total domination number* $\gamma^-_t(G)$, which is studied in [4]. Specifically, a linear time algorithm to compute the total minus domination number of a tree appears in [4].

Let f be a mTkSF. The set of vertices *covered by* f is defined as $C_f = \{v \in V | f(N(v)) \geq 1\}$, while the set P_f is defined as $\{v \in V | f(v) = 1\}$.

The motivation for studying the total k-subdomination number is rich and varied from a modeling perspective. For example, by assigning the values -1, 0 or $+1$ to the vertices of a graph we can model networks of people or organizations in which global decisions must be made (e.g. positive, neutral or negative responses or preferences). We assume that each individual has one vote and that each individual has an initial opinion. We assign $+1$ to vertices (individuals) which have a positive opinion, 0 to vertices which have a neutral opinion and -1 to vertices which have a negative opinion. We also assume, however, that an individual's vote is affected by the opinions of neighboring individuals. In particular, each individual gives equal weight to the opinions of neighboring individuals (thus individuals of high degree have greater "influence"). A voter votes 'aye' if there are more vertices in its (open) neighborhood with positive opinion than with negative opinion, otherwise the vote is 'nay'. We seek an assignment of opinions that guarantee at least k vertices voting aye. We call such an assignment of opinions a k-positive assignment. Among all k-positive assignments of opinions, we are interested primarily in the minimum number of vertices (individuals) who have a positive opinion. The minus total k-subdomination number is the minimum possible sum of all opinions, -1 for a negative opinion and $+1$ for a positive opinion, in a k-positive assignment of opinions. The minus total k-subdomination number represents, therefore, the minimum number of individuals which can have positive opinions and in doing so force at least k individuals to vote aye.

In this paper, we show that the associated decision problem is **NP**-complete for bipartite graphs and also present cubic time algorithms to compute the minus total k-subdomination and minus k-subdomination numbers of a tree. The decision problem corresponding to the computation of the minus k-subdomination number was shown to be **NP**-complete for bipartite graphs in [5].

2 Complexity result

In this section we will show that the decision problem corresponding to the computation of the minus total k-subdomination number is **NP**-complete by describing a polynomial transformation from the **NP**-complete problem **EXACT COVER BY 3-SETS**.

Let $r = \frac{a}{b} \leq 1$ be a fixed positive rational number (in lowest terms). Consider the decision problem

MINUS TOTAL SUBDOMINATING FUNCTION (MTSF)

INSTANCE: A graph G and an integer ℓ.

QUESTION: Is there a function $f : V(G) \to \{-1, 0, 1\}$ of weight ℓ or less for G such that $|C_f| \geq r|V(G)|$?

In this section we show that **MTSF** is NP-complete by describing a polynomial transformation from the following NP-complete problem (see [3]):

EXACT COVER BY 3-SETS (X3C)

INSTANCE: A set $X = \{x_1, \ldots, x_{3q}\}$ and a set $\mathcal{C} = \{C_1, \ldots, C_m\}$ where $C_j \subseteq X$ and $|C_j| = 3$ for $j = 1, \ldots, m$.

QUESTION: Does $\mathcal{C}$ have a pairwise disjoint q-subset of $\mathcal{C}$ whose union is X (i.e. an exact cover)?

If $r = 1$, then **MTSF** is the **NP**-complete problem **MINUS TOTAL DOMINATING FUNCTION** (see [4]). Hence, we also assume that $r < 1$. For two real numbers a and b, we say that a *divides* b if there is an integer k such that $b = ka$.

Theorem 1 **MTSF** *is NP-complete, even for bipartite graphs.*

Proof. It is obvious that **MTSF** is in **NP**. To show that **MTSF** is an **NP**-complete problem, we will establish a polynomial transformation from the NP-complete problem **X3C**. Let $X = \{x_1, \ldots, x_{3q}\}$ and $\mathcal{C} = \{C_1, \ldots, C_m\}$ be an arbitrary instance of **X3C** where $C_j \subseteq X$ and $|C_j| = 3$ for $1 \leq j \leq m$. We will construct a bipartite graph G and an integer ℓ such that this instance of **X3C** will have an exact cover if and only if there is a function $f : V(G) \to \{-1, 0, 1\}$ of weight at most ℓ such that $|C_f| \geq r|V(G)|$.

Corresponding to each $x_i \in X$, associate the graph constructed from the path P_5, with vertices labeled x_i, y_i, z_i, v_i, w_i, and the path P_2, with vertices labeled u_i and t_i, by joining the vertices u_i and v_i. Corresponding to each C_j, associate the graph constructed from the path P_4, with vertices labeled c_j, d_j, e_j, f_j, and the path P_2, with vertices labeled g_j, h_j, by joining the vertices g_j and e_j. Add the edges $\{x_i c_j \mid x_i \in C_j\}$ and call the resulting graph H. Note that $n(H) = 6m + 21q$. Let

$$\mu = \begin{cases} 0 & \text{if } r \text{ divides } 6m + 21q \\ a - (6m + 21q) \bmod a & \text{otherwise} \end{cases}.$$

Then μ is the smallest nonnegative integer that may be added to $6m + 21q$ so that r divides $6m + 21q + \mu$ evenly. Construct the (bipartite) graph $G = (V, E)$ as follows. Take the disjoint union of two copies of H, say H_1 and H_2, add a set S of $\alpha := 2(\frac{6m+21q+\mu}{r} - (6m + 21q))$ vertices, and, with $S = \{s_1, s_2, \ldots, s_\alpha\}$, add the edges $s_k s_{k+1}$, where $k = 1, 3, \ldots, \alpha - 1$. The graph G has order $2(\frac{6m+21q+\mu}{r})$, and, since $0 \leq \mu \leq a - 1$ and a is a constant, G can be constructed from the input in polynomial time. Lastly, let $\ell = 2(8m + 28q + 2\mu - (\frac{6m+21q+\mu}{r}))$. We will denote a vertex ν_i or ν_j of H in H_β by $\nu_{i,\beta}$ or $\nu_{j,\beta}$, for $\beta = 1, 2$.

Suppose $\mathcal{C}' \subseteq \mathcal{C}$ is an exact cover for X. Let $P = \cup_{i=1}^{3q}\{u_{i,1}, u_{i,2}, v_{i,1}, v_{i,2}, z_{i,1}, z_{i,2}\} \cup \cup_{j=1}^{m}\{g_{j,1}, g_{j,2}, e_{j,1}, e_{j,2}, d_{j,1}, d_{j,2}\} \cup \cup_{k=1}^{2\mu}\{s_k\} \cup \{c_{j,1}, c_{j,2} \mid C_j \in \mathcal{C}'\}$ and $M = \cup_{i=1}^{3q}\{w_{i,1}, w_{i,2}\} \cup \cup_{j=1}^{m}\{f_{j,1}, f_{j,2}\} \cup \cup_{k=2\mu+1}^{\alpha}\{s_k\}$.

Define $f : V \to \{-1, 0, 1\}$ by

$$f(x) = \begin{cases} 1 & \text{if } x \in P \\ -1 & \text{if } v \in M \\ 0 & \text{otherwise.} \end{cases}$$

Notice that $f(V) = \sum_{i=1}^{3q} 6 + \sum_{j=1}^{m} 6 + \sum_{k=1}^{2\mu} 1 + 2q - (\sum_{i=1}^{3q} 2 + \sum_{j=1}^{m} 2 + \sum_{k=2\mu+1}^{\alpha} 1) = 14q + 4m + 4\mu - \alpha = 2(8m + 28q + 2\mu - \frac{6m+21q+\mu}{r})$. Note that $x \in C_f$ for all $x \in V(H_1) \cup V(H_2) \cup \cup_{k=1}^{2\mu}\{s_k\}$. Thus, $|C_f| \geq 2(6m + 21q) + 2\mu = 2(6m + 21q + \mu) = r2\frac{6m+21q+\mu}{r} = r|V|$.

We now prove the converse. Let L be the set of all leaves of G. Among all functions $f : V \to \{-1, 0, 1\}$ for which $f(V) \leq \ell$ and $|C_f| \geq r|V(G)|$, choose one, say f, for which $f(L)$ is as small as possible. This implies that $f(x) \in \{-1, 1\}$ for all $x \in S$. Note that $|C_f| \geq r2\frac{6m+21q+\mu}{r} = 12m + 42q + 2\mu$.

We will show that $V(H_1) \cup V(H_2) \subseteq C_f$, while establishing function values for the elements of $V(H_1) \cup V(H_2)$. The assumption of an incorrect function value for a vertex in $V(H_1) \cup V(H_2)$ implies in all of the cases that we consider next $V(H_1) \cup V(H_2) \nsubseteq C_f$. Thus, if $V(H_1) \cup V(H_2) \nsubseteq C_f$, then $|C_f| \geq 12m + 42q$ implies that there exists $s_k \in S \cap C_f$, whence $f(s_{k-1}) = 1$ or $f(s_{k+1}) = 1$ — without loss of generality, we will assume the latter.

The function $g_{r_1,\rho_1,r_2,\rho_2}$ is the function obtained from f by assigning some vertex r_1 in $V(H_1) \cup V(H_2)$ the value ρ_1, some vertex r_2 in S the value ρ_2, while all other vertices are assigned the same value as under f, where $\rho_1, \rho_2 \in \{-1, 0, 1\}$. In all cases, a neighbor of some vertex r_1 in $V(H_1) \cup V(H_2)$ will become covered, while the neighbor of r_2 in S will no longer be covered, so $|C_g| \geq |C_f|$. Moreover, in all cases, $g(V) \leq f(V) \leq \ell$.

Let $i \in \{1, \dots, 3q\}, j \in \{1, \dots, m\}$ and $\beta \in \{1, 2\}$.

Fact 1. $f(g_{j,\beta}) = 1$ (and, similarly, $f(e_{j,\beta}) = f(u_{i,\beta}) = f(v_{i,\beta}) = 1$).

Proof. For suppose, to the contrary, $f(g_{j,\beta}) \leq 0$. Then $h_{j,\beta} \notin C_f$ and $g = g_{g_{j,\beta},1,s_{k+1},f(g_{j,\beta})}$ has $g(L) < f(L)$, which is a contradiction. $\diamondsuit$

Fact 2. $f(h_{j,\beta}) = 0$ (and, similarly, $f(t_{i,\beta}) = 0$).

Proof. For suppose, to the contrary, $f(h_{j,\beta}) = -1$. Then $g_{j,\beta} \notin C_f$ and $g = g_{h_{j,\beta},0,s_{k+1},-1}$ has $g(L) < f(L)$, which is a contradiction. Furthermore, since $f(L)$ is a minimum, $f(h_{j,\beta}) = f(t_{i,\beta}) = 0$. $\diamondsuit$

Fact 3. $e_{j,\beta} \in C_f$ (and, similarly, $v_{i,\beta} \in C_f$).

Proof. For suppose, to the contrary, that $e_{j,\beta} \notin C_f$. Then $f(d_{j,\beta}) + f(f_{j,\beta}) \leq -1$. This implies that $f(d_{j,\beta}) \leq 0$. Then $g = g_{d_{j,\beta},1,s_{k+1},f(d_{j,\beta})}$ has $g(L) < f(L)$, which is a contradiction. $\diamondsuit$

Fact 4. $f(f_{j,\beta}) = -1$ and $f(d_{j,\beta}) = 1$ (and, similarly, $f(z_{i,\beta}) = 1$ and $f(w_{i,\beta}) = -1$).

Proof. Since $e_{j,\beta} \in C_f$, $f(d_{j,\beta}) + f(g_{j,\beta}) + f(f_{j,\beta}) = f(d_{j,\beta}) + 1 + f(f_{j,\beta}) \geq 1$, which implies $f(d_{j,\beta}) + f(f_{j,\beta}) \geq 0$. Now $f(L)$ is minimum and can be achieved by setting $f(f_{j,\beta}) = -1$ which forces $f(d_{j,\beta}) = 1$.

Fact 5. $d_{j,\beta} \in C_f$ and $f(c_{j,\beta}) \geq 0$ (and, similarly, $z_{i,\beta} \in C_f$ and $f(y_{i,\beta}) \geq 0$).

Proof. If $d_{j,\beta} \notin C_f$, then, since $f(e_{j,\beta}) = 1$, $f(c_{j,\beta}) = -1$. Then $g = g_{c_{j,\beta},1,s_{k+1},-1}$ has $g(L) < f(L)$, which is a contradiction. $\diamondsuit$

Fact 6. $y_{i,\beta} \in C_f$ and $f(c_{i,\beta}) \geq 0$.

Proof. If $y_{i,\beta} \notin C_f$, then, since $f(z_{i,\beta}) = 1$, $f(x_{i,\beta}) = -1$. Then $g = g_{x_{i,\beta},1,s_{k+1},-1}$ has $g(L) < f(L)$, which is a contradiction. $\diamondsuit$

Fact 7. $c_{j,\beta} \in C_f$ and $x_{i,\beta} \in C_f$.

Proof. Since $c_{j,\beta}$ is adjacent to $d_{j,\beta}$, which is assigned $+1$ under f, and three vertices in $\{x_{1,\beta}, \dots, x_{3q,\beta}\}$, all of which are assigned at least 0 under f, we have that $c_{j,\beta} \in C_f$.

Suppose, to the contrary, that $x_{i,\beta} \notin C_f$. Then all vertices adjacent to $x_{i,\beta}$ are assigned the value 0 under f – particularly $f(y_{i,\beta}) = 0$. It now follows that $g =$

$g_{y_{i,\beta},1,s_{k+1},-1}$ has $g(L) < f(L)$, which is a contradiction. $\diamondsuit$

Combining the facts above, we have $V(H_1) \cup V(H_2) \subseteq C_f$. Since $n(H_1) + n(H_2) = 12m + 42q$ and $|C_f| \geq 12m + 42q + 2\mu$, $|S \cap C_f| \geq 2\mu$.

Let $X_\beta = \{x_{1,\beta}, \ldots, x_{3q,\beta}\}$, $Y_\beta = \{y_{1,\beta}, \ldots, y_{3q,\beta}\}$, $C_\beta = \{c_{1,\beta}, \ldots, c_{m,\beta}\}$, $cx_\beta = |X_\beta \cap P_f|$, $cy_\beta = |Y_\beta \cap P_f|$ and $cc_\beta = |C_\beta \cap P_f|$.

Since $f(V(H_1 \cup H_2) - X_1 - X_2 - Y_1 - Y_2 - C_1 - C_2) = 2(2(3q) + 2m) = 12q + 4m$, $f(V) \geq 12q + 4m + cx_1 + cx_2 + cy_1 + cy_2 + cc_1 + cc_2 + 2\mu + (-1)(2\frac{6m+21q+\mu}{r} - 2(6m + 21q) - 2\mu) = 54q + 16m + 4\mu + cx_1 + cx_2 + cy_1 + cy_2 + cc_1 + cc_2 - 2(\frac{6m+21q+\mu}{r})$. But $f(V) \leq 56q + 16m + 4\mu - 2(\frac{6m+21q+\mu}{r})$, and so $cx_1 + cx_2 + cy_1 + cy_2 + cc_1 + cc_2 \leq 2q$. Hence $cc_1 + cc_2 \leq 2q - (cx_1 + cx_2 + cy_1 + cy_2)$, and so at most $3(c_1 + c_2) \leq 6q - 3(x_1 + x_2 + y_1 + y_2)$ vertices of $X_1 \cup X_2$ are adjacent to vertices of $(C_1 \cup C_2) \cap P_f$, cx_1 vertices in X_1 are assigned a $+1$ under f, cx_2 vertices in X_2 are assigned a $+1$ under f, cy_1 vertices in Y_1 are assigned a $+1$ under f, and cy_2 vertices in Y_2 are assigned a $+1$ under f. Thus, at most $6q - 3(cx_1 + cx_2 + cy_1 + cy_2) + cx_1 + cx_2 + cy_1 + cy_2 = 6q - 2(cx_1 + cx_2 + cy_1 + cy_2)$ of $X_1 \cup X_2$ are either adjacent to a vertex of $(Y_1 \cup Y_2 \cup C_1 \cup C_2) \cap P_f$ or assigned a $+1$ under f. If $cx_1 + cx_2 + cy_1 + cy_2 > 0$, then there is a vertex in $X_1 \cup X_2$, say x, such that $x \notin C_f$, which is a contradiction. Thus, $cx_1 + cx_2 + cy_1 + cy_2 = 0$, and $c_1 + c_2 \leq 2q$. Since $x_{i,\beta} \in C_f$ for $i = 1, \ldots 3q$ and $\beta = 1, 2$, $c_1 = q$ and $c_2 = q$. It now follows that $\mathcal{C}' = \{C_j \mid f(c_{j,1}) = 1\}$ is an exact three cover for X. $\diamondsuit$

3 A cubic algorithm to compute $\gamma^-_{tks}(T)$ of a tree T

In this section, we will present a cubic time algorithm to compute the minus total k-subdomination number of a tree. The tree T will be rooted and represented by the resulting parent array parent[1 ... n]. We make use of the well-known fact that the tree T can be constructed recursively from the single vertex K_1 using only one rule of composition, which combines two trees (G, x) and (H, y), by adding an edge between x and y and calling x the root of the larger tree F. We express this as follows: $(F, x) = (G, x) \circ (H, y)$. With each such subtree (F, x), we associate the following data structure:

1. **table**$[x]$**.numvertices:** the number of vertices in the subtree (F, x).
2. **table**$[x]$**.degree:** $\deg_F(x)$.
3. **table**$[x]$**.sum**$[f(x), t, k]$: the minimum weight of a function $f : V(F) \to \{-1, 0, 1\}$ such that x is assigned $f(x)$, $|t| \leq \deg_T(x) - \deg_F(x)$ (representing all possible sums of assignments of -1, 0 and $+1$ to the vertices of $N_T(x) - N_F(x)$ and $|M(f, F, t, x)| \geq k$ for $0 \leq k \leq$ **table**$[x]$**.numvertices**, where $M(f, F, t, x)$ is defined as $\{v \mid f(N_F(v)) + t \geq 1$ when $v = x$ and $f(N_F(v)) \geq 1$ when $v \neq x\}$.

Our input consist of the order of the tree T, say **n**, and the **parent** array of the tree, rooted at a certain vertex. The root of the tree T is labeled with 1, the vertices on the next level are labeled with 2 through 2 plus the number of vertices on level 2, and so on. Using the **parent** array, we compute $\deg_T(x)$ for each vertex x, $x = 1, \ldots, n$. We then initialize the variable **table**$[x]$ for each vertex x, where $x = 1, \ldots, n$. Let x be an arbitrary vertex of T. Initially, $(F, x) = (K_1, x)$, whence **table**$[x]$**.numvertices**=1 and **table**$[x]$**.degree**=0. Suppose t is an integer such that $|t| \leq \deg_T(x) - \deg_F(x) = \deg_T(x)$, representing all possible sums of assignments of -1, 0 and $+1$ to the vertices of $N_T(x) - N_F(x) = N_T(x)$. Then $t \in \{-\deg_T(x), \ldots, \deg_T(x)\}$. The only way for $f(N_F(x)) + t = t \geq 1$, is for $t \geq 1$. Thus, we have the following initializations:

Case 1: $t \in \{1, \ldots, \deg_T(x)\}$. Then **table**$[x]$**.sum**$[f(x), t, 1]$ = **table**$[x]$**.sum**$[f(x), t, 0] = f(x)$ where $f(x) \in \{-1, 0, 1\}$.

Case 2: $t \in \{-\deg_T(x), \ldots, 0\}$. Then **table**$[x]$**.sum**$[f(x), t, 1]$ is undefined, and **table**$[x]$**.sum**$[f(x), t, 0]$=$f(x)$ where $f(x) \in \{-1, 0, 1\}$.

The following code implements the aforementioned discussion.

```
for vertex ← 1 to n
  do degree[vertex] ← 0

for vertex ← 2 to n
  do { degree[vertex] ← degree[vertex] + 1
       degree[parent[vertex]] ← degree[parent[vertex]] + 1
for vertex ← 1 to n
  do { table[vertex].numvertices ← 1
       table[vertex].degree ← 0

       for excessvalue ← 1 to degree[vertex]
         do for rootvalue ← $-1$ to 1
         do { table[vertex].sum[rootvalue,excessvalue,1] ← rootvalue
              table[vertex].sum[rootvalue,excessvalue,0] ← rootvalue

       for excessvalue ← -degree[vertex] to 0
         do for rootvalue ← $-1$ to 1
         do { table[vertex].sum[rootvalue,excessvalue,1] ← $\infty$
              table[vertex].sum[rootvalue,excessvalue,0] ← rootvalue
```

Inputting the **parent** array takes $O(n)$ steps, while computing the **degree** array from the **parent** array also takes $O(n)$ steps. Initializing the array **table** takes $O(\sum_{\text{vertex}=1}^{n}(2\deg_T(\text{vertex}) + 1) \times 3) = O(6 \times 2m(T)) + O(3n) = O(12(n-1)) + O(n) = O(n)$ steps. Thus, the overall complexity here is $O(n)$.

Our next result shows that our algorithm is correct.

Theorem 2 *Suppose (G,x) and (H,y) are two disjoint rooted subtrees, and let $(F,x) = (G,x) \circ (H,y)$. Let $s \in \{-1,0,1\}$, t be an integer such that $|t| \leq \deg_T(x) - \deg_F(x)$ and k be an integer with $0 \leq k \leq |V(F)|$. Then*

$\mathbf{table}[x]\mathbf{.sum}[s,t,k]= \min\{\mathbf{table}[x]\mathbf{.sum}[s,t{+}s',j]{+}\mathbf{table}[y]\mathbf{.sum}[s',s,k{-}j] \mid s' \in \{-1,0,1\}, 0 \leq j \leq k\} =$
$\min\{\mathbf{table}[x]\mathbf{.sum}[s,t{+}s',j]{+}\mathbf{table}[y]\mathbf{.sum}[s',s,k{-}j] \mid s' \in \{-1,0,1\}, \max\{0,k-|V(H)|\} \leq j \leq \min\{k,|V(G)|\}$.

Moreover, $|t| \leq \deg_T(x) - \deg_F(x)$ if and only if $-(\deg_T(x) - \deg_G(x) - 1) \leq t \leq \deg_T(x) - \deg_G(x) - 1$.

Proof. Suppose $f : V(F) \to \{-1,0,1\}$ such that $\mathbf{table}[x]\mathbf{.sum}[s,t,k]= f(V(F))$. Let g (respectively, h) be the restriction of f on $V(G)$ (respectively, $V(H)$) and $s^* = h(y) = f(y)$. Note that $f(N_F(x)) + t = g(N_G(x)) + t + s^*$ and $f(N_F(v)) = g(N_G(v))$ for all $v \in V(G) - \{x\}$, while $f(N_F(y)) = h(N_H(y)) + s$ and $f(N_F(v)) = g(N_H(v))$ for all $v \in V(H) - \{y\}$. Thus, $k \leq |M(f,F,t,x)| = |M(g,G,t+s^*,x)| + |M(h,H,s,y)|$. If $j = |M(g,G,t{+}s^*,x)|$, then $k-j \leq |M(h,H,s,y)|$. It now follows that $\mathbf{table}[x]\mathbf{.sum}[s,t+s^*,j] + \mathbf{table}[y]\mathbf{.sum}[s^*,s,k-j] \leq g(V(G)) + h(V(H)) = \mathbf{table}[x]\mathbf{.sum}[s,t,k]$. Hence, $\min\{\mathbf{table}[x]\mathbf{.sum}[s,t+s',j]{+}\mathbf{table}[y]\mathbf{.sum}[s',s,k-j] \mid s' \in \{-1,0,1\}, 0 \leq j \leq k\} \leq \mathbf{table}[x]\mathbf{.sum}[s,t,k]$.

On the other hand, suppose $g : V(G) \to \{-1,0,1\}$ such that $g(V(G)) = \mathbf{table}[x]\mathbf{.sum}[s,t+s',j]$ and $h : V(H) \to \{-1,0,1\}$ such that $h(V(H)) = \mathbf{table}[y]\mathbf{.sum}[s',s,k-j]$. Define $f : V(F) \to \{-1,0,1\}$ by $f(v) = g(v)$ if $v \in V(G)$ and $f(v) = h(v)$ for all $v \in V(H)$. As before, $f(N_F(x)) + t = g(N_G(x)) + t + s'$ and $f(N_F(v)) = g(N_G(v))$ for all $v \in V(G) - \{x\}$, while $f(N_F(y)) = h(N_H(y)) + s$ and $f(N_F(v)) = g(N_H(v))$ for all $v \in V(H) - \{y\}$. Thus, $|M(f,F,t,x)| = |M(g,G,t+s',x)| + |M(h,H,s,y)| \geq j + (k-j) = k$. Hence, $\mathbf{table}[x]\mathbf{.sum}[s,t,k] \leq f(V(F)) = g(V(G)) + h(V(H)) = \mathbf{table}[x]\mathbf{.sum}[s,t+s',j] + \mathbf{table}[y]\mathbf{.sum}[s',s,k-j]$. Thus, $\mathbf{table}[x]\mathbf{.sum}[s,t,k] \leq \min\{\mathbf{table}[x]\mathbf{.sum}[s,t{+}s',j]{+}\mathbf{table}[y]\mathbf{.sum}[s',s,k{-}j] \mid s' \in \{-1,0,1\}, 0 \leq j \leq k\}$.

Since $0 \leq j \leq |V(G)|$ and $j \leq k$, we have $0 \leq k-j \leq |V(H)|$, so that $0 \geq j-k \geq -|V(H)|$, whence $j \geq k - |V(H)|$. We conclude that $\max\{0,k-|V(H)|\} \leq j \leq \min\{k,|V(G)|\}$.

Lastly, $|t| \leq \deg_T(x) - \deg_F(x)$ if and only if $-\deg_T(x) + \deg_G(x) + 1 \leq t \leq \deg_T(x) - \deg_G(x) - 1$, since $\deg_F(x) = \deg_G(x) + 1$. $\diamondsuit$

At the conclusion of our algorithm, $T = F$, and so $t = 0$. Clearly, $\gamma^-_{tks}(T) = \min\{\ \mathbf{table}[1]\mathbf{.sum}[-1,0,k],\ \mathbf{table}[1]\mathbf{.sum}[0,0,k]\ \mathbf{table}[1]\mathbf{.sum}[1,0,k]\ \}$.

We are now in a position to present the remainder of the algorithm.

```
for oldroot ← n downto 2
do
    resulttable.numvertices ← table[oldroot].numvertices + table[parent[oldroot]].numvertices
    resulttable.degree ← table[parent[oldroot]].degree + 1
    range ← degree[parent[oldroot]] – resulttable.degree
    for newrootvalue ← -1 to 1
      do for newrootexcess ← -range to range
      do for k ← 0 to resulttable.numvertices
      do
        minimum ← ∞
        startvalue ← max(0, k - table[oldroot].numvertices)
        stopvalue ← min(k, table[parent[oldroot]].numvertices)
        for j ← startvalue to stopvalue
        do
          for oldrootvalue ← -1 to 1
          do
            number ← degree[parent[oldroot]] - table[parent[oldroot]].degree - 1
            if -number ≤ newrootexcess ≤ number
              then
                summand1 ← table[parent[oldroot]].
                sum[newrootvalue, newrootexcess + oldrootvalue, j]
                summand2 ← table[oldroot].sum[oldrootvalue,
                                    newrootvalue, k-j]
                temp ← summand1 + summand2
            if (temp < minimum)
              then minimum ← temp
        resulttable.sum[newrootvalue, newrootexcess, k] ← minimum
    table[parent[oldroot]] ← resulttable
for k ← 0 to n
  do output (k, min{table[1].sum[-1, 0, k], table[1].sum[0, 0, k], table[1].sum[1, 0, k]})
```

The complexity of the above part of the algorithm, excluding the output phase, is

$$\begin{aligned}
&O(\sum_{\text{n-oldroot}=0}^{(n-2)} 3 \times (2 \times \deg_T[\text{parent[oldroot]}] + 1) \times n \times n \times 3) \\
&= O(18n^2 \sum_{v\in V(T)} \deg_T(v)) + O(n \times 9n^2) = O(18n^2 2m(T)) + O(n^3) \\
&= (18n^2 \times 2 \times (n-1)) + O(n^3) \\
&= O(n^3) + O(n^3) = O(n^3),
\end{aligned}$$

while the complexity of the output phase is $O(n)$. Thus, the overall complexity of the algorithm is $O(n^3)$.

4 A cubic algorithm to compute $\gamma^-_{ks}(T)$ of a tree T

In this section, we present a cubic algorithm to compute $\gamma^-_{ks}(T)$ of a tree T. The approach here is similar to what we described in the previous section. Here we have the following data structure, associated with the subtree (F, x).

1. **table[x].numvertices:** the number of vertices in the subtree (F, x).

2. **table**$[x]$**.degree:** $\deg_F(x)$.

3. **table**$[x]$**.sum**$[f(x), t, k]$: the minimum weight of a function $f : V(F) \to \{-1, 0, 1\}$ such that x is assigned $f(x)$, $|t| \le \deg_T(x) - \deg_F(x)$ (representing all possible sums of assignments of -1, 0 and $+1$ to the vertices of $N_T(x) - N_F(x)$ and $|\{v \mid f(N_F[v]) + t \ge 1$ when $v = x$ and $f(N_F[v]) \ge 1$ when $v \ne x\}| \ge k$, where $0 \le k \le$ **table**$[x]$**.numvertices**.

The initialization phase here proceeds as follows.

Let x be an arbitrary vertex of T. Initially, $(F, x) = (K_1, x)$, whence

table$[x]$**.numvertices** $= 1$ and **table**$[x]$**.degree** $= 0$.

Suppose t is an integer such that $|t| \le \deg_T(x) - \deg_F(x) = \deg_T(x)$, representing all possible sums of assignments of -1, 0 and $+1$ to the vertices of $N_T(x) - N_F(x) = N_T(x)$. Then $t \in \{-\deg_T(x), \ldots, \deg_T(x)\}$. The only way for $f(N_F(x)) + f(x) + t = f(x) + t \ge 1$, is for $t \ge 1 - f(x)$. Thus, we have the following initializations:

Case 1: $t \in \{1 - f(x), \ldots, \deg_T(x)\}$. Then **table**$[x]$**.sum**$[f(x), t, 1]$ $=$ **table**$[x]$**.sum**$[f(x), t, 0]$ $= f(x)$ where $f(x) \in \{-1, 0, 1\}$.

Case 2: $t \in \{-\deg_T(x), \ldots, -f(x)\}$. Then **table**$[x]$**.sum**$[f(x), t, 1]$ is undefined, and **table**$[x]$**.sum**$[f(x), t, 0]$$=$$f(x)$ where $f(x) \in \{-1, 0, 1\}$.

One may prove a result analogous to Theorem 2.

We are now in a position to state the algorithm. Note that the initialization phase of the algorithm has complexity $O(\sum_{v \in V(T)} 3 \times (2\deg_T(v) + 1)) = O(6 \times 2m(T)) + O(3n) = O(12(n-1)) + O(n) = O(n)$. Thus, the overall complexity of the algorithm is also $O(n^3)$.

```
for vertex ← 1 to n
  do degree[vertex] ← 0

for vertex ← 2 to n
  do { degree[vertex] ← degree[vertex] + 1
     { degree[parent[vertex]] ← degree[parent[vertex]] + 1

for vertex ← 1 to n
  do for rootvalue ← -1 to 1
     do { for excessvalue ← 1 - rootvalue to degree[vertex]
        {   do { table[vertex].sum[rootvalue,excessvalue,1] ← rootvalue
        {      { table[vertex].sum[rootvalue,excessvalue,0] ← rootvalue
        { for excessvalue ← -degree[vertex] to -rootvalue
        {   do { table[vertex].sum[rootvalue,excessvalue,1] ← ∞
        {      { table[vertex].sum[rootvalue,excessvalue,0] ← rootvalue

for oldroot ← n downto 2
```

```
do
  resulttable.numvertices ← table[oldroot].numvertices + table[parent[oldroot]].numvertices
  resulttable.degree ← table[parent[oldroot]].degree + 1
  range ← degree[parent[oldroot]] − resulttable.degree
  for newrootvalue ← -1 to 1
    do for newrootexcess ← -range to range
    do for k ← 0 to resulttable.numvertices
      do
        minimum ← ∞
        startvalue ← max(0, k - table[oldroot].numvertices)
        stopvalue ← min(k, table[parent[oldroot]].numvertices)
        for j ← startvalue to stopvalue
          do
            for oldrootvalue ← -1 to 1
              do
                number ← degree[parent[oldroot]] - table[parent[oldroot]].degree-1
                if -number ≤ newrootexcess ≤ number
                  then
                    summand1 ← table[parent[oldroot]].
                    sum[newrootvalue, newrootexcess + oldrootvalue, j]

                    summand2 ← table[oldroot].sum[oldrootvalue,
                                    newrootvalue, k-j]
                    temp ← summand1 + summand2
                if (temp < minimum)
                  then minimum ← temp
        resulttable.sum[newrootvalue, newrootexcess, k] ← minimum
  table[parent[oldroot]] ← resulttable

for k ← 0 to n
  do output (k, min{table[1].sum[-1, 0, k], table[1].sum[0, 0, k], table[1].sum[1, 0, k]})
```

References

[1] I. Broere, J.E. Dunbar and J.H. Hattingh, Minus k-subdomination in graphs, *Ars Combin.* **50** (1998), 177–186.

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

[3] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, New York (1979).

[4] L. Harris and J.H. Hattingh, The algorithmic complexity of certain functional variations of total domination in graphs, *Australas. J. Combin.* **29** (2004), 143–156.

[5] J.H. Hattingh, A.A. McRae and E. Ungerer, Minus k-subdomination in graphs III, *Australas. J Combin.* **17** (1998), 69–76.

(Received 27 Apr 2005; revised 27 June 2006)