A linear algorithm for connected domination in partial k-trees*

RADOSŁAW ZIEMANN
University of Gdańsk
Faculty of Mathematics, Physics and Informatics
Wita Stwosza 57, 80-308 Gdańsk
Poland
rziemann@inf.ug.edu.pl

Abstract: In this paper we claim that the Corollary 2 in [V. Pinciu, Dominating sets for outerplanar graphs, *WSEAS Transactions on Mathematics* 1(3), 2004, pp. 55–58] is false. In particular, we present a linear-time algorithm for partial k-trees that solves the problem.

Key–Words: Algorithm, connected domination, linear-time, partial k-trees.

1 Introduction

The connected domination is a one of the well-known NP-hard problems, introduced by Sampathkumar and Walikar in [14]. It is a natural model for some network issues and has been widely studied in literature, see for example [5], [7] and [9]. In [11] the author claims that the problem of finding the minimum connected dominating set is NP-hard even for outerplanar graphs (Corollary 2). In this paper, in two ways, we prove that this corollary is incorrect: in Section 2 we express the problem as a logical formula solvable in linear-time, while in Section 3 present a combinatorial algorithm based on dynamic programming.

We generally use standard graph terminology [6]. Let $G = (V_G, E_G)$ be a simple finite connected graph. A *connected dominating* set D of G is a subset of V_G such that every vertex not in D is adjacent to at least one vertex in D, and the subgraph $G[D]$ induced by D is connected. The *connected domination number* $\gamma_c(G)$ of G is the cardinality of a minimum connected dominating set of G. A graph G is *outerplanar* if it has a crossing-free embedding in the plane such that all vertices are on the boundary of its outer face. If addition of a single edge in G results in a graph that is not outerplanar, then G is called a *maximal outerplanar* graph. Furthermore, the *open neighbourhood* of a vertex u of a graph G, denoted by $N_G(u)$, is the set of all vertices adjacent to u. The *closed neighbourhood* of u, denoted by $N_G[u]$, is the set $N_G(u) \cup \{u\}$. Similarly, for a subset $X \subseteq V_G$ of vertices, the *open neighbourhood* of X, denoted by $N_G(X)$, is defined to be $\bigcup_{u \in X} N_G(u)$, and the *closed neighbourhood* of X, denoted by $N_G[X]$, is the set $N_G(X) \cup X$.

*Supported by the Grant 2015/17/B/ST6/01887 (National Science Centre, Poland)

2 Complexity

In [13] Robertson and Seymour introduced the concept of the graph parameter, called treewidth, playing an important role in algorithmic graph theory. Many problems which are NP-hard on arbitrary graphs can be solved in polynomial-time on graphs of bounded treewidth [3]. Moreover, Courcelle's theorem states that every decision or optimization problem definable in the monadic second order logic (MSO) has a linear-time algorithm on graphs with this property. The graph problem in question may be defined in terms of sets of vertices of the given graph and the binary adjacency relation between the vertices (see [4] for more details).

We recall that outerplanar graphs constitute a subclass of partial 2-trees, graphs with the treewidth of at most 2 (see comprehensive characterization in [15]). Furthermore, the connected domination problem in graph $G = (V_G, E_G)$, where D is a connected dominating set, can be expressed in MSO by the following formula:

$$\bigwedge_{u \in V_G} \bigvee_{v \in V_G} (u \in D \vee v \in D \wedge adj(u, v)) \qquad (1)$$

and

$$\bigwedge_{S \subseteq V_G} \Big[\bigvee_{u,v \in D} (u \in S \wedge v \notin S) \Rightarrow \Rightarrow \bigvee_{x,y \in D} (x \in S \wedge y \notin S \ \wedge \ adj(x, y)) \Big], \qquad (2)$$

where adj$(\cdot)$ is an adjacency relation. The expression ensures that every vertex in V_G is dominated (1) and $G[D]$ is connected (2).

Corollary 1. *There exists a linear-time algorithm for finding a minimum connected dominating set in outerplanar graphs.* ■

Unfortunately, this approach can be inpractical, even for graphs with treewidth equal to one, because of hidden constants and complex implementation [8]. Therefore, in the next section, we give a simple combinatorial linear-time algorithm solving that problem in partial k-trees, a supeclass of outerplanar graphs.

3 Algorithm

Our algorithm is based on the standard dynamic programming method for partial k-trees [16], being graphs with the treewidth of at most k, and on new ideas showed in [10]. First, we find an ordered tree decomposition [13] corresponding to the input graph $G = (V_G, E_G)$. Then we traverse it by a bottom-up technique calculating and storing proper values for each node.

Let $H = (V_H, E_H)$ be a k-tree, where $V_H = V_G$ and $E_G \subseteq E_H$, see Fig 1, and let $(v_1, ..., v_n)$ be some perfect elimination ordering (peo) of the vertices in H, where the set $V_i = \{v_i, ..., v_j\}$ $(i < j)$ induces a $(k+1)$-vertex clique in H under peo, $i = 1, ..., n-k$. We create the directed tree $\mathcal{T}$ with the vertex set $\{V_1, ..., V_{n-k}\}$ rooted at V_{n-k}, where the parent of the vertex V_i in $\mathcal{T}$ is the vertex V_p such that $V_i \cap V_p = k$ and $p > i$ is as small as possible, see Fig 2. The subgraph of G corresponding to the subgraph of $\mathcal{T}$ rooted at V_i is denoted by G^i. It is known that finding a tree decomposition of a graph requires a linear-time [2], as well as finding a perfect elimination ordering [12].

We divide our problem into two natural subproblems: domination and connectivity, giving some necessary definitions. Let X be a subset of V_{G^i} and let $F_{X,i} : V_{G^i} \longrightarrow \{-1, 0, 1\}$ be the function corresponding to the domination problem, expressed by the following formula:

$$F_{X,i}(u) = \begin{cases} 1 & \text{if } u \in X \\ 0 & \text{if } u \in N_{G^i}(X) \\ -1 & \text{if } u \notin N_{G^i}[X]\,. \end{cases}$$

Notice that $|\bigcup_{X \subseteq V_{G^i}} \{F_{X,i}\}| = 2^{|V_{G^i}|}$. The function $f_{X,i} : V_i \longrightarrow \{-1, 0, 1\}$ such that $f^{-1}_{X,i}(1) \subseteq F^{-1}_{X,i}(1), f^{-1}_{X,i}(0) \subseteq F^{-1}_{X,i}(0)$ and $f^{-1}_{X,i}(-1) \subseteq F^{-1}_{X,i}(-1)$ is called the *partial* function of $F_{X,i}$ and we say that two partial functions $f_{Y,i}, f_{Z,i}$ are *equivalent* if and only if $f_{Y,i}(u) = f_{Z,i}(u)$ for each vertex u in V_i and some $Y, Z \subseteq V_{G^i}$. Next, let $\mathcal{C}_{F_{X,i}} = \{A_1, ..., A_p\}$ be the family of vertex sets of

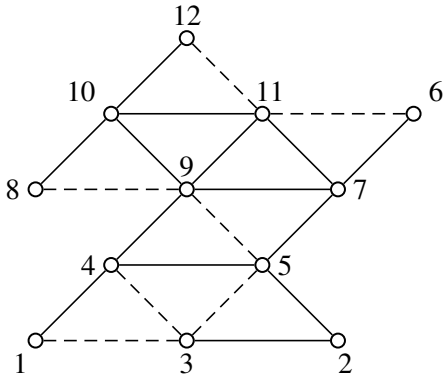

Figure 1: An outerplanar graph G (partial 2-tree) and the maximal outerplnar graph H (2-tree) obtained by adding (dashed) edges with peo (1,2,3,4,5,6,7,8,9,10,11,12).

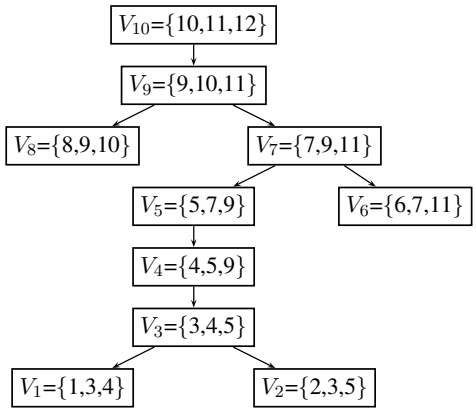

Figure 2: The tree $\mathcal{T}$ of H (Fig 1).

the connected components of $G^i[X]$ and let $\mathcal{C}_{f_{X,i}} = \{B_1, ..., B_p\}$ be the family of sets such that $B_1 \subseteq A_1, ..., B_p \subseteq A_p$ and $\bigcup_{j=1}^{j=p} B_j = V_i \cap X$ (notice that B_j may be the empty set). Now, let $\mathcal{P}_i$ be the set of all possible pairs $(f_{X,i}, \mathcal{C}_{f_{X,i}})$ of $V_i \in \mathcal{T}$, taken over all $X \subseteq V_{G^i}$. One can observe that the cardinality of the set of the all non equivalent partial functions for V_i is at most 3^{k+1} and the number of different sets $\mathcal{C}_{f_{X,i}}$ for the partial function $f_{X,i}$ is at most B_k, where B_k is the k-th Bell number (the number of the possible partitions of a set). Thus, we have $|\mathcal{P}_i| \leq 3^{k+1} \cdot B_k$ for each $V_i \in \mathcal{T}$. Observe that $D \subseteq V_G$ is a connected dominating set of G if and only if $F_{D,r}(u) \neq -1$ for each vertex $u \in V_G$ and $G[F^{-1}_{D,r}(1)]$ is connected, where $r = n - k$. Hence, similarly as in algorithm for convex domination number proposed in [10], we say that

a pair is *feasible* if it fulfills the following three rules:

The domination rule
Recall that if $D \subseteq V_G$ is a dominating set of G then $N_G[v_i] \cap D \neq \emptyset$ for each $i \in \{1, ..., n-k\}$. Consequently, if $(f_{X,i}, \mathcal{C}_{f_{X,i}})$ for $X \subseteq D$ is feasible then we must have $f_{X,i}(v_i) \neq -1$: v_i is dominated by itself or by a vertex in V_{G^i}, $N_G[v_i] \cap (V_G \setminus V_{G^i}) = \emptyset$.

The connectivity rules
If $D \subseteq V_G$ is a connected dominating set of G, $X \subseteq D$ and $i \in \{1, ..., n-k\}$ then the following two properties must be satisfied:
1. if $v_i \in D$ then vertex v_i must be reachable from any other node belonging to D, especially to $D \cap V_i$. Therefore, if $f_{X,i}(v_i) = 1$, $|V_i \cap X| > 1$ and $(f_{X,i}, \mathcal{C}_{f_{X,i}})$ is feasible then there must exist at least one vertex u in V_i sharing the same (connected) component with v_i ($\{v_i, u\} \subseteq B, B \in \mathcal{C}_{f_{X,i}}$).
2. if V_i is a child of V_j in $\mathcal{T}$, then we must have $|V_i \cap V_j \cap X| > 0$ (otherwise the graph $G \setminus (V_i \cap V_j)$ is not connected). Accordingly, for each feasible pair belonging to $\mathcal{P}_i$ ($\mathcal{P}_j$, respectively) we must have $f_{X,i}(u) = 1$ ($f_{X \cup (D \cap V_j),j}(u) = 1$, respectively) for at least one vertex u in $V_i \cap V_j$.

The set of all feasible pairs for $V_i \in \mathcal{T}$ is denoted by $\mathcal{F}_i$, $\mathcal{F}_i \subseteq \mathcal{P}_i$. Furthermore, let $M : \mathcal{P}_j \times \mathcal{F}_i \longrightarrow \mathcal{P}_j$ be the function $M((g_{Y,j}, \mathcal{C}_{g_{Y,j}}), (h_{Z,i}, \mathcal{C}_{h_{Z,i}})) = (f_{X,j}, \mathcal{C}_{f_{X,j}})$ matching two pairs, one for vertex V_j in $\mathcal{T}$, second for its child V_i, where $X = Y \cup Z$. Therefore, $f^{-1}_{X,j}(1) = g^{-1}_{Y,j}(1)$, $g^{-1}_{Y,j}(1) \setminus \{x\} = h^{-1}_{Z,i}(1) \setminus \{v_i\}$ and $g^{-1}_{Y,j}(0) \cup (h^{-1}_{Z,i}(0) \setminus \{v_i\}) = f^{-1}_{X,j}(0)$, where $x \in V_j \setminus V_i$. Moreover, if $A_1, ..., A_p$ are the vertex sets of the connected components in the graph $G^j[X]$, then $\mathcal{C}_{f_{X,j}} = \{B_1 \subseteq A_1, ..., B_p \subseteq A_p\}$ and $\bigcup_{i=1}^{i=p} B_i = V_j \cap X$. Finally, let $D_{(f_{X,j}, \mathcal{C}_{f_{X,j}})}$ be the cardinality of a minimum set X such that all vertices in $((G^j \setminus V_j) \cup \{v_j\}) \setminus X$ have at least one neighbour in X and $(f_{X,j}, \mathcal{C}_{f_{X,j}})$ fulfills the domination and connectivity rules.

Now we outline the main steps of our algorithm for currently visited vertex V_j in $\mathcal{T}$. For simplicity and clarity of presentation we only show how to calculate the connected domination number of the input graph G, but it is easy to modify our approach to obtain the relevant minimum connected dominating set. Accordingly, we use the symbols f, g, h instead of the full denotations of partial functions, keeping in mind the definitions.

1. First, we determine all of the feasible pairs $(f, \mathcal{C}_f) \in \mathcal{F}_j$, where V_j is a leaf vertex in $\mathcal{T}$, and set $D_{(f,\mathcal{C}_f)}$ to zero for each pair. Therefore, we have $|\mathcal{F}_j| < |\bigcup_{X \subseteq V_j} \{F_{X,j}\}| = 2^{k+1}$.

2. For a non-leaf vertex $V_j \in \mathcal{T}$, taking into account the previous calculated subsolutions for all the children of V_j, we compute the set $\mathcal{F}_j$ and the value $D_{(f,\mathcal{C}_f)}$ for each $(f, \mathcal{C}_f) \in \mathcal{F}_j$, step by step. Let $\mathcal{P}'_j$ and $D'_{(f,\mathcal{C}_f)}$ be the set of all possible pairs for V_j, fulfilling the connectivity rule 2, taken over all $X \subseteq V_j$ (notice that $|\mathcal{P}'_j| < 2^{k+1}$) and the value corresponding to $(f, \mathcal{C}_f) \in \mathcal{P}'_j$, initially set to zero, respectively. We also use auxillary variables $\mathcal{P}''_j$ and $D''_{(f,\mathcal{C}_f)}$ initially set to $\mathcal{P}'_j$ and $D'_{(f,\mathcal{C}_f)}$. Moreover, the subset of feasible pairs of (updated) $\mathcal{P}'_j$ we denote by $\mathcal{F}'_j$.

For each child V_i of V_j, in a sequence, first we expand the set $\mathcal{P}'_j$ by adding new pairs:

$$\bigwedge_{\substack{(f,\mathcal{C}_f) \notin \mathcal{P}'_j \\ (f,\mathcal{C}_f) = M((g,\mathcal{C}_g),(h,\mathcal{C}_h)) \\ (g,\mathcal{C}_g) \in \mathcal{P}''_j \\ (h,\mathcal{C}_h) \in \mathcal{F}_i}} : \quad \mathcal{P}'_j := \mathcal{P}'_j \cup \{(f, \mathcal{C}_f)\}.$$

Then we set $\mathcal{P}''_j$ to the (new) $\mathcal{P}'_j$ and we update the value $D'_{(f,\mathcal{C}_f)}$ for pair $(f, \mathcal{C}_f) \in \mathcal{P}'_j$ using the following formula:

$$D'_{(f,\mathcal{C}_f)} = \min_{\substack{(f,\mathcal{C}_f) = M((g,\mathcal{C}_g),(h,\mathcal{C}_h)) \\ (g,\mathcal{C}_g) \in \mathcal{P}'_j \\ (h,\mathcal{C}_h) \in \mathcal{F}_i}} (D''_{(g,\mathcal{C}_g)} + D_{(h,\mathcal{C}_h)} + h(v_i)).$$

Notice that this values can be computed by going through all pairs $(g, \mathcal{C}_g) \in \mathcal{P}'_j$ (matched with proper pairs $(h, \mathcal{C}_h) \in \mathcal{F}_i$) and storing the temporary (for just visited $(g, \mathcal{C}_g)$) minimum of $D'_{(f,\mathcal{C}_f)}$. Next, we remove the pairs $(f, \mathcal{C}_f)$, which do not belong to the set of values of function M (there does not exist proper pair $(h, \mathcal{C}_h)$ in $\mathcal{F}_i$), from the set $\mathcal{P}'_j$, and at last we set $D''_{(f,\mathcal{C}_f)}$ to the (new) $D'_{(f,\mathcal{C}_f)}$. Observe now that after all updates for all children of V_j, we have $\mathcal{F}_j \subseteq \mathcal{P}_j = \mathcal{F}'_j \subseteq \mathcal{P}'_j$ and $D_{(f,\mathcal{C}_f)} = D'_{(f,\mathcal{C}_f)}$ for each $(f, \mathcal{C}_f) \in \mathcal{F}_j$.

4. Finally, when reaching the root T_r, where $r = n - k$, we compute the connected domination number $\gamma_c(G)$ as follows:

$$\gamma_c(G) = \min_{\substack{(f,\mathcal{C}_f) \in \mathcal{F}_r \\ |\mathcal{C}_f| = 1}} (D_{(f,\mathcal{C}_f)} + \sum_{i=r}^{n} f(v_i)).$$

To get better understanding of above steps see the following example reffering to Fig 1 and 2.

Example for visited vertex V_3
Let $\mathcal{F}_1 = \bigcup_{i=1}^{i=4} \{(h_i, \mathcal{C}_{h_i})\}$ be the set of already calculted feasible pairs for leaf vertex V_1, where:

$$h_1(1) = 0, h_1(3) = -1, h_1(4) = 1, \mathcal{C}_{h_1} = \{\{1\}\};$$

$h_2(1) = h_2(4) = 1, h_2(3) = -1, \mathcal{C}_{h_2} = \{\{1,4\}\}$;

$h_3(1) = 0, h_3(3) = h_3(4) = 1, \mathcal{C}_{h_3} = \{\{3\},\{4\}\}$;

$h_4(1) = h_4(3) = h_4(4) = 1, \mathcal{C}_{h_4} = \{\{1,4\},\{3\}\}$,

and $\mathcal{F}_2 = \bigcup_{i=5}^{i=10}\{(h_i, \mathcal{C}_{h_i})\}$ be the set of already calculted feasible pairs for leaf vertex V_2, where:

$h_5(2) = 0, h_5(3) = 1, h_5(5) = -1, \mathcal{C}_{h_5} = \{\{3\}\}$;

$h_6(2) = 0, h_6(3) = -1, h_6(5) = 1, \mathcal{C}_{h_6} = \{\{5\}\}$;

$h_7(2) = h_7(3) = 1, h_7(5) = 0, \mathcal{C}_{h_7} = \{\{2,3\}\}$;

$h_8(2) = h_8(5) = 1, h_8(3) = 0, \mathcal{C}_{h_8} = \{\{2,5\}\}$;

$h_9(2) = 0, h_9(3) = h_9(5) = 1, \mathcal{C}_{h_9} = \{\{3\},\{5\}\}$;

$h_{10}(2) = h_{10}(3) = h_{10}(5) = 1, \mathcal{C}_{h_{10}} = \{\{2,3,5\}\}$.

Next, $\mathcal{F}_3$ is calculated in three steps. Initially $\mathcal{P}_3' = \bigcup_{i=1}^{i=4}\{(g_i, \mathcal{C}_{g_i})\}$. Accordingly, let:

$g_1(3) = g_1(4) = g_1(5) = 1, \mathcal{C}_{g_1} = \{\{3\},\{4,5\}\}$;

$g_2(3) = g_2(4) = 1, g_2(5) = 0, \mathcal{C}_{g_2} = \{\{3\},\{4\}\}$;

$g_3(3) = g_3(5) = 1, g_3(4) = 0, \mathcal{C}_{g_3} = \{\{3\},\{5\}\}$;

$g_4(3) = -1, g_4(4) = g_4(5) = 1, \mathcal{C}_{g_4} = \{\{4,5\}\}$.

For V_1 we remove one pair: $\mathcal{P}_3' := \mathcal{P}_3' \setminus \{(g_3, \mathcal{C}_{g_3})\}$, while visiting V_2 we are adding two new pairs: $\mathcal{P}_3' := \mathcal{P}_3' \cup \{(g_5, \mathcal{C}_{g_5})\} \cup \{(g_6, \mathcal{C}_{g_6})\}$, where

$(g_5, \mathcal{C}_{g_5}) = M((g_1, \mathcal{C}_{g_1}), (h_{10}, \mathcal{C}_{h_{10}}))$;

$(g_6, \mathcal{C}_{g_6}) = M((g_4, \mathcal{C}_{g_4}), (h_8, \mathcal{C}_{h_8}))$,

and

$g_5(3) = g_5(4) = g_5(5) = 1, \mathcal{C}_{g_5} = \{\{3,4,5\}\}$;

$g_6(3) = 0, g_6(4) = g_6(5) = 1, \mathcal{C}_{g_6} = \{\{4,5\}\}$.

Finally, the set $\mathcal{F}_3 = \{(g_5, \mathcal{C}_{g_5}), (g_6, \mathcal{C}_{g_6})\}$ and $D_{(g_5, \mathcal{C}_{g_5})} = D_{(g_6, \mathcal{C}_{g_6})} = 1$.

In conclusion, the correctness of the algorithm directly follows from the induction on tree $\mathcal{T}$. Namely, we use the following invariant: after each step of the algorithm, partial solutions for so handled vertex in $\mathcal{T}$ have stored proper values, fulfilling the domination and connectivity rules. We omit the details. Next, the running time mostly depends on the number of nodes in $\mathcal{T}$. For each vertex, we compute at most $3^{k+1} \cdot B_k$ subsolutions and perform some additional calculations depending on the constant k. This gives a linear-time complexity (for fixed k), which can be inefficent for large values of k, even if the number of subsolutions for each node is much smaller in practice. However, our approach is effective for outerplanar graphs.

References:

[1] R. E. Bellman, S. E. Dreyfus, Applied Dynamic Programming, Princeton University Press, 1962.

[2] H. L. Bodlaender, A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* 25(6), 1996. pp. 1305-1317.

[3] H. L. Bodlaender, Dynamic programming on graphs with bounded treewidth, *International Colloquium on Automata, Languages, and Programming*, 1988, pp. 105–118.

[4] B. Courcelle, The monadic second-order logic of graphs. I. Recognizable sets of finite graphs, *Inf. Comput.* 85(1), 1990, pp. 12–75.

[5] W. J. Desormeaux, T. W. Haynes, M. A. Henning, Bounds on the connected domination number of a graph, *Discrete Appl. Math.* 161(18), 2013, pp. 2925–2931.

[6] R. Diestel, Graph Theory, Springer, Heidelberg, 2012.

[7] W. Duckworth, B. Mansb, Connected domination of regular graphs, *Discrete Math.* 309(8), 2009, pp. 2305–2322.

[8] M. Frick, M. Grohe, The complexity of first-order and monadic second-order logic revisited, *Ann. Pure Appl. Logic* 130(13), 2004, pp. 3–31.

[9] H. Karami, A. Khodkar, S. M. Sheikholeslami, D. B. West, Connected domination number of a graph and its complement, *Graphs Combin.* 28(1), 2012, pp. 123–131.

[10] M. Lemańska, E. Rivera-Campo, R. Ziemann, R. Zuazua, P. Żyliński, Convex dominating sets in maximal outerplanar Graphs, *Discrete Appl. Math.* 265, 2019, pp. 142–157.

[11] V. Pinciu, Dominating sets for outerplanar graphs, *WSEAS Transactions on Mathematics* 1(3), 2004, pp. 55–58.

[12] D. Rose, G. Lueker, R. E. Tarjan, Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.* 5(2), 1976, pp. 266–283.

[13] N. Robertson, P. D. Seymour, Graph minors. II. Algorithmic aspects of treewidth, *J. Algorithms* 7, 1986, pp. 309–322.

[14] E. Sampathkumar, H. B. Walikar, The connected domination number of a graph, *J. Math. Phys. Sci.* 13(6), 1979, pp. 607–613.

[15] M. Sysło, Characterizations of outerplanar graphs, *Discrete Math.* 26(1), 1979, pp. 47–53.

[16] J. A. Telle, A. Proskurowski, Practical algorithms on partial k-trees with an application to domination-like problems, *Workshop on Algorithms and Data Structures*, 1993, pp. 610–621.