Computing and Informatics, Vol. 27, 2008, 581–591

INDEPENDENCE AND DOMINATION IN PATH GRAPHS OF TREES

Ľudovít Niepel, Anton Černý

Department of Mathematics and Computer Science
Kuwait University
P. O. Box 5969, Safat, 13060, Kuwait
e-mail: {cerny, niepel}@sci.kuniv.edu.kw

Manuscript received 15 February 2006
Communicated by Hong Zhu

Abstract. The problems of determining the maximum cardinality of an independent set of vertices and the minimum cardinality of a maximal independent set of vertices of a graph are known to be NP-complete. We provide efficient algorithms for finding these values for path graphs of trees.

Keywords: Path graph, dominating set, independent set

1 INTRODUCTION

The problem of domination is one of the most studied problems in graph theory ([5]). It is well known and generally accepted that the problem of determining the domination number of an arbitrary graph is a difficult one. Since this problem has been shown to be NP-complete, it is generally thought to require exponential time in the order of the graph. Because of this the attention is turned to the study of classes of graphs for which the domination problem can be solved in polynomial time. For line graphs, the problem of finding the cardinality of maximal independent dominating set can be solved in polynomial time. On the other hand, finding the cardinality of the minimal dominating set in the line graph of an arbitrary graph is NP-complete ([6]).

In this paper we investigate the domination problem for path graphs. Path graphs were introduced by Broersma and Hoede in [2] as a natural generalization of

line graphs. A path graph is defined as follows. Let G be a graph, $k \geq 1$ and P_k be the set of all paths of length k (i.e., with $k+1$ vertices) in G. The vertex set of path graph $P_k(G)$ is the set P_k. Two vertices of $P_k(G)$ are joined by an edge if and only if their intersection is a path of length $k-1$, and their union forms either a cycle or a path of length $k+1$. In this paper we restrict ourselves to path graphs with $k = 2$. A characterization of P_2-path graphs was given in [10]. The connectivity and metric properties of P_2-path graphs were studied in [1, 7, 8]. The independence of P_2-path graphs and P_3-path graphs was studied in [9]. A survey of results in domination and independence on graphs can be found in the monograph [5].

A set S of vertices of a graph G is a dominating set if any vertex from $V(G)$ is in S or it is adjacent to some vertex in S. A set S of vertices is called independent if no two vertices of S are adjacent. It is clear that maximal independent set of vertices is also dominating. The maximum cardinality of an independent set of vertices is denoted as $\beta_0(G)$. The number $i(G)$ is the minimum cardinality of a maximal independent set; or, which is the same, the minimum cardinality of an independent dominating set. It is known that to find exact value of any of the parameters $\beta_0(G)$ and $i(G)$ is NP-hard for general graphs. In paper [9] the authors presented a randomized algorithm for finding the value $\beta_0(P(G))$ where P is a P_2 or P_3 path-graph operator. Exact values of $\beta_0(P(G))$ were found for some special classes of graphs. It was also shown that applying this algorithm with greedy strategy may give incorrect results for trees.

In the present paper we give a characterization of maximal independent sets in P_2-path graphs and then we provide efficient algorithms for finding $\beta_0(P_2(G))$ and $i(P_2(G))$ if G is any tree.

2 BASIC NOTIONS

For a finite set V, let $\binom{V}{2} = \{\{a, b\}; a, b \in V \text{ and } a \neq b\}$. A (*undirected*) *graph* is a pair $G = (V, E)$ where V is a finite set – the set of *vertices* of G and $E \subseteq \binom{V}{2}$ is the set of *edges* of G. The sets of vertices and edges of G will be alternatively denoted as $V(G)$ and $E(G)$, respectively. The undirected *edge* $\{a, b\}$ (where always $a \neq b$) will be denoted as ab; hence $ab = ba$. A *subgraph* of a graph G is a graph $G' = (V', E')$ where $V' \subseteq V$ and $E' \subseteq E$; we write $G' \subseteq G$. A subgraph G' of G is *induced* in G if $G' = (V(G'), \binom{V(G')}{2} \cap E(G))$. Let G_1, G_2 be two subgraphs of a graph G. Their *union* and *intersection* are the subgraphs $G_1 \cup G_2 = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2))$ and $G_1 \cap G_2 = (V(G_1) \cap V(G_2), E(G_1) \cap E(G_2))$ of G, respectively. A *path of length* $r \geq 0$ in G is a subgraph $(\{a_0, a_1, \ldots, a_r\}, \{a_0a_1, a_1a_2, \ldots, a_{r-1}a_r\})$ of G where the vertices $a_0, a_1, \ldots, a_r$ are pairwise distinct. This subgraph will be denoted as $a_0a_1 \cdots a_r$ (sometimes in parentheses); hence $a_0a_1 \cdots a_r = a_ra_{r-1} \cdots a_0$. This notation allows us to identify a path $(\{a\}, \varnothing)$ of length 0 with the vertex a and a path $(\{a, b\}, \{ab\})$ with the edge ab. For example, if $a \in V(G)$, $G - a$ denotes the graph $(V(G) - \{a\}, E(G) \cap \binom{V(G)-\{a\}}{2})$. A cycle of length $r \geq 2$ in G is a subgraph $(\{a_0, a_1, \ldots, a_{r-1}\}, \{a_0a_1, a_1a_2, \ldots, a_{r-2}a_{r-1}, a_{r-1}a_0\})$ of G where the

vertices $a_0, a_1, \ldots, a_{r-1}$ are pairwise distinct. A graph G is *isomorphic* to a graph H if there is a bijection $f : V(G) \to V(H)$ such that $E(H) = \{f(a)f(b); ab \in E(G)\}$.

A *directed (oriented) graph* is a pair $G = (V, E)$ where V is a finite set – the set of *vertices* of G and $E \subseteq V \times V$ is the set of *directed (oriented) edges* of G. The directed *edge* $\langle a, b\rangle$ (where possibly $a = b$) will be denoted as $\overrightarrow{ab}$ or $\overleftarrow{ba}$. For a vertex $v \in V$, $in\deg v = |\{u \in V; \overrightarrow{uv} \in E\}|$ and $out\deg v = |\{u \in V; \overrightarrow{vu} \in E\}|$.

The path operator is a generalization of the well-known line operator. The result of the path operator P_r, $r \geq 1$, on a graph G is the graph $P_r(G)$ with the vertex set $V(P_r(G))$ consisting of all paths of length r in G and the edge set $E(P_r(G))$ consisting of all pairs p_1p_2 where $p_1 \cap p_2$ is a path of length $r-1$ and $p_1 \cup p_2$ is a path or a cycle of length $r+1$. A graph isomorphic to $P_r(G)$ for some graph G is called P_r-path graph.

3 MAXIMAL INDEPENDENT SETS OF VERTICES

In this part we describe a connection between maximal independent sets in P_2-path graphs and special orientations of original graphs. We will omit the subscript 2 in the notation of operator P_2. We shall use the notion of mixed graph for a graph that contains both oriented and non-oriented edges, such that there is at most one oriented or not oriented edge between any pair of vertices. A *mixed graph* is a triple $\widehat{G} = (V, E, \overrightarrow{E})$ such that (V, E) is an undirected graph and $(V, \overrightarrow{E})$ is a directed graph without loops. We denote $\widehat{E} = \{ab; ab \in E \text{ or } \overrightarrow{ab} \in \overrightarrow{E} \text{ or } \overrightarrow{ba} \in \overrightarrow{E}\}$. We will alternatively use the notation $V(\widehat{G}) = V, E(\widehat{G}) = E$, $\overrightarrow{E}(\widehat{G}) = \overrightarrow{E}$ and $\widehat{E}(\widehat{G}) = \widehat{E}$. For a vertex $v \in V$, $in\deg v = |\{u \in V; \overrightarrow{uv} \in \overrightarrow{E}\}|$ and $out\deg v = |\{u \in V; \overrightarrow{vu} \in \overrightarrow{E}\}|$. A *semi-oriented graph* is a mixed graph with the set $E \cup \overrightarrow{E}$ containing at most one of the edges $\overrightarrow{uv}, \overrightarrow{vu}, uv$ for each pair of vertices $u \neq v$. We will say that $\widehat{G}$ is a *semi-orientation* of the graph $(V(\widehat{G}), \widehat{E}(\widehat{G}))$ (*complete orientation* if $E(\widehat{G}) = \varnothing$).

A vertex in the graph $P(G)$ is a path abc in G. We will call b the *root* of this path. Let S be a set of vertices of the path graph $P(G)$ of a graph G. We will associate with S a mixed graph by assigning orientation to some edges in the following way: for each vertex $abc \in S$ we direct the edge ab as $\overrightarrow{ab}$ (and bc as $\overrightarrow{cb}$).

Lemma 1. Let $G = (V, E)$ be a graph and S an independent set of vertices in $P(G)$. Let $\widehat{G}$ be a mixed graph with $V(\widehat{G}) = V$, $\overrightarrow{E}(\widehat{G}) = \{\overrightarrow{ab}; abc \in S$ for some $c \in V\}$ and $E(\widehat{G}) = \{ab \in E; \overrightarrow{ab} \notin \overrightarrow{E}(\widehat{G})$ and $\overrightarrow{ba} \notin \overrightarrow{E}(\widehat{G})\}$. Then

1.1 $\widehat{G}$ is a semi-oriented graph.

1.2 Let $v \in V(\widehat{G})$. Either there is no edge of the form $\overrightarrow{uv}$ in $\overrightarrow{E}(\widehat{G})$ or $\overrightarrow{E}(\widehat{G})$ contains at least two such edges

If S is maximal then

1.3 For each $uv \in E(\widehat{G})$, if $vw \in E(G)$ for some $w \in V - \{u\}$ then $\overrightarrow{vw} \in \overrightarrow{E}(\widehat{G})$.

Proof. The set S does not contain two adjacent vertices abc and bad (where, possibly, $d = c$), therefore $\overrightarrow{E}(\widehat{G})$ cannot contain an edge $\overrightarrow{ab}$ together with $\overrightarrow{ba}$; hence 1.1 is true. The condition 1.2 is clear from the construction of $\widehat{G}$, since $abc = cba$ and $\overrightarrow{E}(\widehat{G})$ contains the edge $\overrightarrow{cb}$ together with $\overrightarrow{ab}$. Let now S be maximal and let $uv \in E(\widehat{G})$. If, for a vertex $w \in V - \{u\}$, either $\overleftarrow{vw} \in \overrightarrow{E}(\widehat{G})$ or $vw \in E(\widehat{G})$ then $uvw \notin S$ and $S \cup \{uvw\}$ is an independent set. Indeed, if, for some $x \in V(\widehat{G})$, $xuv \in S$, then $\overrightarrow{vu} \in E(\widehat{G})$ and if $vwx \in S$ then $\overrightarrow{vw} \in E(\widehat{G})$. □

Lemma 2. Let $\widehat{G}$ be a semi-orientation of a graph $G = (V, E)$. Let $S_{\widehat{G}} = \{abc; \overrightarrow{ab} \in \overrightarrow{E}(\widehat{G})$ and $\overrightarrow{cb} \in \overrightarrow{E}(\widehat{G})\}$. Then $S_{\widehat{G}}$ is an independent set of vertices in $P(G)$. If $\widehat{G}$ satisfies the conditions 1.2 and 1.3 of Lemma 1 then $S_{\widehat{G}}$ is maximal.

Proof. The set $S_{\widehat{G}}$ cannot contain two adjacent vertices abc and bcd (where d is not necessarily distinct from a), since $\overrightarrow{E}(\widehat{G})$ then should contain both the edge $\overrightarrow{cb}$ and the edge $\overrightarrow{bc}$. Hence the set $S_{\widehat{G}}$ is independent. Assume that $\widehat{G}$ satisfies 1.2 and 1.3. To prove that $S_{\widehat{G}}$ is maximal, assume in contrary that there is a vertex $abc \in V(P(G)) - S_{\widehat{G}}$ such that $S_{\widehat{G}} \cup \{abc\}$ is an independent set in $P(G)$. Then at least one of the edges $\overrightarrow{ab}, \overrightarrow{cb}$ is not in $\overrightarrow{E}(\widehat{G})$; assume it is $\overrightarrow{ab}$. Then $\overrightarrow{ba}$ cannot be in $E(\widehat{G})$ since, following 1.2, there is $x \in V(G)$ such that $\overrightarrow{xa} \in E(\widehat{G})$ and the vertex $xab \in S_{\widehat{G}}$ adjacent to abc. On the other hand, ab cannot be in $E(\widehat{G})$, since then 1.3 implies $\overrightarrow{bc} \in E(\widehat{G})$ and, following 1.2, there is $x \in V(G)$ such that $\overrightarrow{xc} \in E(\widehat{G})$; the vertex $xcb \in S_{\widehat{G}}$ is then adjacent to abc. □

The size of $S_{\widehat{G}}$ from Lemma 2 can be expressed by counting the pairs of oriented edges headed in the same vertex v by the formula

$$|S_{\widehat{G}}| = \sum_{v \in V(\widehat{G})} \binom{in\deg v}{2}. \tag{1}$$

We will call the number $\binom{in\deg v}{2}$, being the number of vertices in $S_{\widehat{G}}$ rooted at vertex v, the *contribution of vertex* v. Hence finding, for a given graph G, a maximal independent vertex set in $P(G)$ of maximum/minimum size is equivalent to finding the maximum/minimum size of the set $S_{\widehat{G}}$ among all semi-orientations $\widehat{G}$ of G satisfying 1.2 and 1.3 of Lemma 1:

$$\begin{aligned} \beta_0(P(G)) &= \max_{\widehat{G}} \sum_{v \in V(\widehat{G})} \tbinom{in\deg v}{2} \\ i(P(G)) &= \min_{\widehat{G}} \sum_{v \in V(\widehat{G})} \tbinom{in\deg v}{2} \end{aligned} \tag{2}$$

As there exist exponentially many (with respect to the size of the graph) semi-orientations of G, the formulas (2) do not provide an efficient method for finding the values of $\beta_0(P(G))$ and $i(P(G))$. We will show that in the case when G is a tree, these values can be determined efficiently.

4 THE ALGORITHMS

Starting from now we will consider a fixed non-empty tree T. We chose randomly a vertex r (the *root* of T).

Our algorithms for computing the values $\beta_0(P_2(G))$ and $i(P_2(G))$ will apply the standard dynamic programming technique on trees. We will compute auxiliary values $\overrightarrow{M}_v, \overleftarrow{M}_v$ in the former case and $m_v, \overrightarrow{m}_v, \overleftarrow{m}_v$ in the latter case, for a semi-oriented tree rooted in vertex v, using the corresponding values for subtrees of an internal vertex v, while the values for leaves will be obtained in a straight way. The auxiliary values will depend on the orientation of the edge connecting v to its parent. To be able to use for r the same formula as for other vertices v, we add a new (*dummy*) vertex d and the edge rd. We denote the resulting new tree as T'. For each vertex v from T there is now a unique parent u in T' being the vertex preceding v on the path from d to v; the vertex v will be called the *child* of u. If a vertex is named v, its parent will always be named u. We denote as T_v the subtree of T' induced by the set of vertices consisting of u and all vertices, which can be reached from u by a path starting by uv (Figure 1). For a semi-orientation $\widehat{T}$ of T_v we partition the set $N_v = \{w \in V(T); wv \in E(T), w \neq u\}$ of children of v into three subsets $N_v^{\widehat{T}} = \{w \in N_v; wv \in E(\widehat{T})\}$, $\overrightarrow{N}_v^{\widehat{T}} = \{w \in N_v; \overrightarrow{wv} \in \overrightarrow{E}(\widehat{T})\}$ and $\overleftarrow{N}_v^{\widehat{T}} = \{w \in N_v; \overleftarrow{wv} \in \overrightarrow{E}(\widehat{T})\}$,

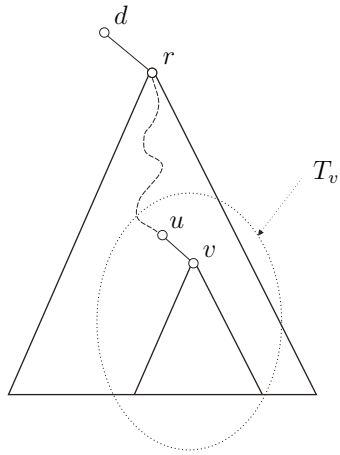

Fig. 1. The tree T_v

We will first consider the problem of finding $\beta_0(P(T))$. We start by a general note. Assume, for a graph G, a maximal independent set S in $P(G)$ of maximum size and the semi-orientation $\widehat{G}$ of G corresponding to S as in Lemma 1. If we add orientation to the edges from $E(\widehat{G})$ in an arbitrary way, still the independent set from Lemma 2 will be S. Therefore to find the maximum in the first formula of (2),

it is enough to consider only complete orientations of G. Moreover, we need not limit ourselves by conditions 1.2 and 1.3, since we may look for a maximum independent set in a wider class of independent sets than just the maximal ones.

In the following formulas defining the parameters $\overrightarrow{M}_v$ and $\overleftarrow{M}_v$ for the tree T_v the maximum is taken for all complete orientations $\widehat{T}_{\overrightarrow{uv}}$, $\widehat{T}_{\overleftarrow{uv}}$ of T_v containing the edge $\overrightarrow{uv}, \overleftarrow{uv}$, respectively.

$$\overrightarrow{M}_v = \max_{\widehat{T}_{\overrightarrow{uv}}} |S_{\widehat{T}_{\overrightarrow{uv}}}| \quad \overleftarrow{M}_v = \max_{\widehat{T}_{\overleftarrow{uv}}} |S_{\widehat{T}_{\overleftarrow{uv}}}|. \tag{3}$$

Clearly, $\beta_0(P(T)) = \overleftarrow{M}_r$. The following lemma follows from the formula (1). The combinatorial number in each formula in the lemma is the contribution of the vertex v. The orientation of the edge uv into v participates in this contribution.

Lemma 3.

$$\begin{aligned} \overrightarrow{M}_v &= \sum_{w \in \overrightarrow{N}_v^{\widehat{T}}} \overleftarrow{M}_w + \sum_{w \in \overleftarrow{N}_v^{\widehat{T}}} \overrightarrow{M}_w + \binom{|\overrightarrow{N}_v^{\widehat{T}}|+1}{2} \\ \overleftarrow{M}_v &= \sum_{w \in \overrightarrow{N}_v^{\widehat{T}}} \overleftarrow{M}_w + \sum_{w \in \overleftarrow{N}_v^{\widehat{T}}} \overrightarrow{M}_w + \binom{|\overrightarrow{N}_v^{\widehat{T}}|}{2} \end{aligned} \tag{4}$$

where $\widehat{T}$ denotes the complete orientation of T_v containing the edge $\overrightarrow{uv}$ (in the case of $\overrightarrow{M}_v$) or $\overleftarrow{uv}$ (in the case of $\overleftarrow{M}_v$), for which the maximum is achieved in (3).

The formulas from (4) can be rewritten as

$$\begin{aligned} \overrightarrow{M}_v &= \sum_{w \in N_v} \overleftarrow{M}_w + \sum_{w \in \overleftarrow{N}_v^{\widehat{T}}} (\overrightarrow{M}_w - \overleftarrow{M}_w) + \binom{|\overrightarrow{N}_v^{\widehat{T}}|+1}{2} \\ \overleftarrow{M}_v &= \sum_{w \in N_v} \overleftarrow{M}_w + \sum_{w \in \overleftarrow{N}_v^{\widehat{T}}} (\overrightarrow{M}_w - \overleftarrow{M}_w) + \binom{|\overrightarrow{N}_v^{\widehat{T}}|}{2}. \end{aligned} \tag{5}$$

Our algorithm will perform the depth-first search of the tree T' starting from the dummy vertex d. Each vertex $v \neq d$ will be processed at the last visit during the search. The processing consists of the computation of the two parameters $\overrightarrow{M}_v$ and $\overleftarrow{M}_v$ In the moment of processing of v the values of all these two parameters will be available for all children of v. However, we must find, separately for each of the two parameters, which choice of orientations of the edges wv provides the maximum value of the formulas in (5). For example, in the case of $\overrightarrow{M}_v$ we have to maximize $\sum_{w \in \overrightarrow{N}_v^{\widehat{T}}} (\overrightarrow{M}_w - \overleftarrow{M}_w) + \binom{|\overrightarrow{N}_v^{\widehat{T}}|+1}{2}$ since $\sum_{w \in N_v} \overleftarrow{M}_w$ is a constant not depending on the actual orientation of the edges vw. To avoid checking all possible orientations, we will sort the sequence of differences $\overrightarrow{M}_w - \overleftarrow{M}_w$ and use the fact that if $k = |\overrightarrow{N}_v^{\widehat{T}}|$ is fixed, then $\sum_{w \in \overrightarrow{N}_v^{\widehat{T}}} (\overrightarrow{M}_w - \overleftarrow{M}_w)$ reaches the maximum if it involves the k largest differences $\overrightarrow{M}_w - \overleftarrow{M}_w$. Then we find the maximum of the particular expression from (5)

among all possible values $1 \le k \le \deg v - 1$. The resulting value $\overleftarrow{M}_r$ is then equal to $\beta_0(P(T))$.

Algorithm 1. Input: A tree T. Output: $\beta_0(P_2(T))$.
1. Choose any vertex r in T. Add a dummy vertex d and the edge rd to T.
2. Call the recursive procedure *Process(r,d)*
3. Return $\overleftarrow{M}_r$.

Procedure *Process(v,u)*
1. $N := \{w \in V(T) | vw \in E(T), v \ne u\}; p := |N|;$
2. if $p = 0$ then $\{\overrightarrow{M}_v := 0;\ \overleftarrow{M}_v := 0;$ return$\}$
3. for each $w \in N$ call *Process(w,v)*
4. $s := \sum_{w \in N_v} \overleftarrow{M}_w;$
5. sort the differences $d_w = \overleftarrow{M}_w - \overrightarrow{M}_w$, $w \in N$, resulting in a sequence $d_1 \ge d_2 \ge \cdots \ge d_p$.
6. $\overrightarrow{M}_v := s + \max_{0 \le k \le p} (\binom{k+1}{2} + \sum_{i=1}^{k} d_i);\ \overleftarrow{M}_v := s + \max_{0 \le k \le p} (\binom{k}{2} + \sum_{i=1}^{k} d_i);$
7. return

Let us now turn to the problem of finding $i(P(T))$. Let v be vertex in T. We will say that a semi-orientation of T_v is *good* if it satisfies 1.2 (except for vertex u) and 1.3 of Lemma 1. Exempting u from condition 1.2 allows us to consider the orientation $\overleftarrow{uv}$ of the edge uv, which may be possible if T_v is considered within the whole tree T, and the orientation $\overrightarrow{rd}$ in the tree $T' = T_r$. We have to find the minimum size of a maximal independent set of vertices in $P(T)$ corresponding to a good semi-orientation of the tree T' containing $\overrightarrow{rd}$. In the following formulas defining the parameters m_v, $\overrightarrow{m}_v$ and $\overleftarrow{m}_v$ for the tree T_v, the minimum is taken for all good semi-orientations $\widehat{T}_{uv}$, $\widehat{T}_{\overrightarrow{ut}}$, $\widehat{T}_{\overleftarrow{uv}}$ of T_v containing the edge $uv, \overrightarrow{ut}, \overleftarrow{uv}$, respectively.

$$m_v = \min_{\widehat{T}_{uv}} |S_{\widehat{T}_{uv}}| \quad \overrightarrow{m}_v = \min_{\widehat{T}_{\overrightarrow{uv}}} |S_{\widehat{T}_{\overrightarrow{uv}}}| \quad \overleftarrow{m}_v = \min_{\widehat{T}_{\overleftarrow{uv}}} |S_{\widehat{T}_{\overleftarrow{uv}}}|. \tag{6}$$

If no semi-orientation of T_v containing the particular edge exists, the minimum in (6) takes by default the value ∞. Clearly, $i(P(T)) = \overleftarrow{m}_r$.

We will first identify cases when the parameters from (6) take the value ∞. We denote $N_v^{\infty} = \{w \in N_v; m_w = \infty\}$, $\overrightarrow{N}_v^{\infty} = \{w \in N_v; \overleftarrow{m}_w = \infty\}$ and $\overleftarrow{N}_v^{\infty} = \{w \in N_v; \overrightarrow{m}_w = \infty\}$.

Lemma 4. Let $v \in V(T)$. Then 1. $\overleftarrow{m}_v < \infty$, 2. $m_v < \infty$ iff $\overleftarrow{N}_v^{\infty} = \varnothing$, 3. $\overrightarrow{m}_v < \infty$ iff v is not a leaf.

Proof.

1. We have to prove that there always exists a good semi-orientation of the tree T_v containing $\overrightarrow{vu}$. We will prove it by induction based on the height h of the tree T_v.

If $h = 1$ then v is a leaf. The semi-orientation containing only $\overrightarrow{vu}$ is good. Let us now assume that the height of T_v is $h \geq 2$ and the assertion is true for all trees T_w of height smaller than h. If v has only one child w then either w is a leaf or w has children. If w is a leaf then there is a good semi-orientation of T_v consisting of wv and $\overrightarrow{vu}$, else take for each child z of w the good semi-orientation of T_z containing $\overrightarrow{zw}$ and add to the union of these semi-orientations the edges $\overrightarrow{vw}$ and $\overrightarrow{vu}$. The resulting mixed graph is a good semi-orientation of T_v. If v has at least two children, then a good semi-orientation of T_v is obtained as follows: take for each child z of v the good semi-orientation of T_z containing $\overrightarrow{zv}$ and add to the union of these semi-orientations the edge $\overrightarrow{vu}$.

2. The assertion follows from condition 1.3 of Lemma 1.
3. If v is a leaf then 1.2 of Lemma 1 implies that $\overrightarrow{m}_v = \infty$. If v is not a leaf, then a good semi-orientation can be constructed by taking for each child z of v the good semi-orientation of T_z containing $\overrightarrow{zv}$ (which exists, following 1.) and adding the edge $\overrightarrow{uv}$.

□

Corollary 5. Let $v \in V(T)$. Then $\overrightarrow{N}_v^{\infty} = \varnothing$.

As in the case of Lemma 3, the following Lemma 6 follows from the formula (1). The sum of zero terms is by default 0. The condition 1.3 implies that $|N_v^{\widehat{T}}| \leq 1$ and if uv is not oriented as $\overleftarrow{uv}$ then $|N_v^{\widehat{T}}| = \varnothing$. If the edge uv is not oriented then $\overleftarrow{N}_v^{\widehat{T}} = N_v$.

Lemma 6. Let $v \in V(T)$.

1. If $\overleftarrow{N}_v^{\infty} = \varnothing$ then $m_v = \sum_{w \in N_v} \overrightarrow{m}_w$.
2. If v is a leaf then $\overrightarrow{m}_v = \infty$, else $\overrightarrow{m}_v = \sum_{w \in \overrightarrow{N}_v^{\widehat{T}}} \overleftarrow{m}_w + \sum_{w \in \overleftarrow{N}_v^{\widehat{T}}} \overrightarrow{m}_w + \binom{|\overrightarrow{N}_v^{\widehat{T}}|+1}{2}$ (in this case $\overleftarrow{N}_v^{\infty} \subseteq \overrightarrow{N}_v^{\widehat{T}}$).
3. $$\overleftarrow{m}_v = \begin{cases} m_{w_0} + \sum_{w \in \overleftarrow{N}_v^{\widehat{T}}} \overrightarrow{m}_w & \text{if } N_v^{\widehat{T}} = \{w_0\} \\ \text{(in this case } \overleftarrow{N}_v^{\infty} \subseteq \{w_0\}) & \\ \sum_{w \in \overrightarrow{N}_v^{\widehat{T}}} \overleftarrow{m}_w + \sum_{w \in \overleftarrow{N}_v^{\widehat{T}}} \overrightarrow{m}_w + \binom{|\overrightarrow{N}_v^{\widehat{T}}|}{2} & \text{if } N_v^{\widehat{T}} = \varnothing \\ \text{(in this case } \overleftarrow{N}_v^{\infty} \subseteq \overrightarrow{N}_v^{\widehat{T}}). & \end{cases}$$

where $\widehat{T}$ denotes the semi-orientation of T_v containing the edge uv (in the case of m_v), $\overrightarrow{uv}$ (in the case of $\overrightarrow{m}_v$) or $\overleftarrow{uv}$ (in the case of $\overleftarrow{m}_v$), for which the minimum

is achieved in (6). In the second case for $\overrightarrow{m}_v$ and in the case of $\overleftarrow{m}_v$ the condition is fulfilled.

The algorithm for finding the value $i(P(T))$ is based on similar ideas as Algorithm 1. This time we are looking for minimum instead of the maximum and we must take into consideration that no edge can be oriented from a vertex from $\overleftarrow{N}_v^{\infty}$ into v. Therefore the algorithm will sort just those differences $\overrightarrow{m}_w - \overleftarrow{m}_w$ where w belongs to $\overleftarrow{N}_v^{OK} = N_v - \overleftarrow{N}_v^{\infty}$. The first case in 2. of Lemma 6 may take place if w_0 is from $N_v^{OK} = \{w \in N_v; m_w < \infty\}$ and $N_v - \{w\} \subseteq \overleftarrow{N}_v^{OK}$.

Algorithm 2. Input: A tree T. Output: $i(P_2(T))$.
1. Choose any vertex r in T. Add a dummy vertex d and the edge rd to T.
2. Call the recursive procedure *Process(r,d)*
3. Return $\overleftarrow{m}_r$.

Procedure *Process(v,u)*
1. $N := \{w \in V(T) | vw \in E(T), v \neq u\}; p := |N|;$
$\overleftarrow{N}^{OK} := \{w \in N; \overrightarrow{m}_w < \infty\}; \overleftarrow{N}^{\infty} := \{w \in N; \overrightarrow{m}_w = \infty\}; \overleftarrow{p}^{OK} := |N^{OK}|;$
$N^{OK} := \{w \in N; m_w < \infty\}; p^{OK} := |N^{OK}|;$
2. if $p = 0$ then $\{m_v := 0; \overrightarrow{m}_v := \infty; \overleftarrow{m}_v := 0;$ return$\}$
3. for each $w \in N$ call *Process(w,v)*
4. $s := \sum_{w \in N^{OK}} \overleftarrow{m}_w$; $t := \sum_{w \in \overleftarrow{N}^{OK}} \overrightarrow{m}_w$
5. sort the differences $d_w = \overrightarrow{m}_w - \overleftarrow{m}_w$, $w \in \overleftarrow{N}^{OK}$, resulting in a sequence $d_1 \leq d_2 \leq \cdots \leq d_{\overleftarrow{p}^{OK}}$.
6. $m_v := \infty; \overrightarrow{m}_v := s + \min_{0 \leq k \leq \overleftarrow{p}^{OK}} (\binom{k+1}{2} + \sum_{i=1}^{k} d_i)$; $\overleftarrow{m}_v := s + \min_{0 \leq k \leq \overleftarrow{p}^{OK}, k \neq 1} (\binom{k}{2} + \sum_{i=1}^{k} d_i)$;
7. if $\overleftarrow{p}^{OK} \geq p - 1$
then $\{$if $\overleftarrow{N}^{\infty} = \{w_0\}$ then $q := m_{w_0} + t$
else $\{ q := \min_{w \in N}(m_w + t - \overrightarrow{m}_w); m_v := t\}$
$\overleftarrow{m}_v := \min(\overleftarrow{m}_v, q)\}$
8. return

Note: The value ∞ may be assigned to the variable q in Step 7.

Analysis of Algorithms 1 and 2 We will analyze Algorithm 2 only. The time and space complexity of Algorithm 1 is clearly not greater than that of Algorithm 2. Let the tree T consist of n vertices. The space used by Algorithm 2 is $O(n)$. The total running time of the algorithm can be computed as the sum of the times spent by processing each node (steps 1, 4, 5, 6, 7). Let p_v be the number computed in step 1 of the call of *Process(v, u)*. All the sets in step 1 can be constructed in the time $O(p_v)$. The computation of the sums in step 4 and the minimum in step 7 take time $O(p_v)$, as well. The sums in step 6 can be computed by increments with increasing k; computing the minimum takes time $O(p_v)$. Therefore each of the steps 1, 4, 6 and 7

requires time $O(p_v)$. The complexity of sorting in step 5 depends on the algorithm used. Sorting by comparison requires time $O(p_v \log p_v)$. Using algorithms like Radix Sort or Bucket Sort ([3]) will not improve the performance, since p_v can be as large as n and $\log n$ bits are necessary for its encoding. However, since p_v is limited by n, sorting may be performed here using the van Emde Boas trees ([4]), in time $O(p_v \log\log p_v)$. Hence processing the node v can be done in time $O(p_v \log\log p_v)$. Then the total time spent by the algorithm is $O(\sum_{v\in V(T)} p_v \log\log p_v)$. Since p_v is the number of children of v in the depth-first search tree and each vertex is a child of exactly one vertex, we have $\sum_{v\in V(T)} p_v = n$. Then $\log\log p_v < \log\log n$ and the time complexity of the algorithm is $O(\sum_{v\in V(T)} p_v \log\log p_v) = O((\sum_{v\in V(T)} p_v)\log\log n) = O(n\log\log n)$. The result is summarized in the following theorem.

Theorem 7. Let T be a tree consisting of n vertices. The values $\beta_0(P_2(T))$ and $i(P_2(T))$ can be found in time $O(n\log\log n)$, using the space $O(n)$.

Acknowledgement

The authors are grateful to an unknown referee for his helpful comments, which led to an improvement of the result. The work of the first author was supported by Kuwait University Research Grant No. SM01/05.

REFERENCES

[1] Balbuena, C.—Ferrero, D.: Edge-Connectivity and Super Edge-Connectivity of P_2-Path Graphs. Discrete Mathematics, Vol. 269, 2003, pp. 13–20.

[2] Broersma, H. J.—Hoede, C.: Path Graphs. Graph Theory, Vol. 13, 1989, pp. 427–444.

[3] Cormen, T. H.—Leiserson, C. E.—R. L.: Introduction to Algorithms. 2nd ed. Cambridge, MIT Press, 2001.

[4] Peter van Emde Boas, P.—Kaas, R.—Zijlstra, E.: Design and Implementation of an Efficient Priority Queue. Mathematical Systems Theory, Vol. 10, 1977, pp. 99–127.

[5] Haynes, T. W.—Hedetniemi, S. T.—Slater, P. J.: Fundamentals of Domination in Graphs. 1st ed. New York: Marcel Dekker, 1998.

[6] Haynes, T. W.—Hedetniemi, S. T.—Slater, P. J.: Domination in Graphs: Advanced Topics. 1st ed. New York: Marcel Dekker, 1998.

[7] Knor, M.—Niepel, Ľ.: Diameter in Iterated Path Graphs. Discrete Mathematics, Vol. 233, 2001, pp. 151–161.

[8] Knor, M..—Niepel, Ľ.: Centers in Path Graphs. JCISS, Vol. 24, 1999, pp. 79–86.

[9] Knor, M..—Niepel, Ľ.: Independence Number in Path Graphs. Computing and Informatics, Vol. 23, 2004, pp. 179–187.

[10] Li, H.—Lin, Y.: On the Characterization of Path Graphs. J. Graph Theory, Vol. 17, 1993, pp. 463–466.

Ľudovít Niepel received the master degree in mathematics from Comenius University, Bratislava, in 1974, and the Ph. D. degree in geometry and topology from Comenius University in 1980. Currently, he is an associate professor at Kuwait University. His research interests include graph theory and computer graphics.

Anton Černý received the master degree in mathematics from Comenius University, Bratislava, in 1976, and the Ph. D. degree in computer science from Czechoslovak Academy of Sciences, Prague, in 1985. Currently, he is an associate professor at Kuwait University. His research interests include combinatorics on words and graph theory.