Available at
www.ElsevierMathematics.com
POWERED BY SCIENCE DIRECT
Discrete Applied Mathematics 136 (2004) 197–215
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

A linear algorithm for the Hamiltonian completion number of the line graph of a cactus

Paolo Detti[a], Carlo Meloni[b]

[a] Dipartimento di Ingegneria dell'Informazione, Università di Siena, Via Roma, 56, I-53100 Siena, Italy
[b] Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, Via E. Orabona, 4 I-70125 Bari, Italy

Received 25 June 2001; received in revised form 1 November 2002; accepted 19 February 2003

Abstract

Given a graph $G=(V,E), HCN(L(G))$ is the minimum number of edges to be added to its line graph $L(G)$ to make $L(G)$ Hamiltonian. This problem is known to be *NP*-hard for general graphs, whereas a $O(|V|)$ algorithm exists when G is a tree. In this paper a linear algorithm for finding $HCN(L(G))$ when G is a cactus is proposed.

Keywords: Algorithms; Hamiltonian path; Hamiltonian completion number; Cactus; Line graph

1. Introduction

In this paper a graph $G=(V,E)$ is called Hamiltonian if it has a Hamiltonian path, and the problem of finding the minimum number of edges which need to be added to G to make it Hamiltonian is considered. This problem is known in literature as the problem of finding the *Hamiltonian completion number* of a graph and will be denoted as $HCN(G)$. In particular, we investigate the problem restricted to a particular class of graphs, called *line graphs*. The line graph $L(G)$ of $G=(V,E)$ is a graph having $|E|$ nodes, each node of $L(G)$ being associated to an edge of G. There is an edge between two nodes of $L(G)$ if the corresponding edges of G are adjacent. Linear-time algorithms exist for recognizing a line graph $L(G)$ and obtain its *root* graph G [15,20].

Given a graph $G=(V,E)$, a *trail* is a sequence $w := (v_0, e_0, v_1, e_1, v_2, e_2, \ldots, e_{k-1}, v_k)$, where $(v_0, v_1, v_2, \ldots, v_k)$ are nodes of G, $(e_0, e_1, e_2, \ldots, e_{k-1})$ are distinct edges of G,

E-mail addresses: detti@dii.unisi.it (P. Detti), meloni@deemail.poliba.it (C. Meloni).

doi:10.1016/S0166-218X(03)00441-4

and v_i and v_{i+1} are the endpoints of e_i for $0 \leqslant i \leqslant k-1$. The trail is a *path* if its nodes $(v_0, v_1, v_2, \ldots, v_k)$ are distinct. In other words, a trail is a path that can pass more times for the same node. A path or a trail may consist of a single node.

A *dominating trail* D_T in G is a trail such that each edge of G has at least one endpoint belonging to it (i.e., a dominating trail *covers* all the edges of G). Note that a dominating trail may not exist. A *dominating trail set* Σ is a collection of edge-disjoint trails that altogether cover all the edges of G. A *minimum dominating trail set* (*MDTS*) is a dominating trail set of minimum cardinality.

Harary and Nash-Williams [11] link the problem of finding $HCN(L(G))$ and *MDTS* showing that the line graph $L(G)$ of a graph G has a Hamiltonian path if and only if G has a dominating trail. As consequence, if $HCN(L(G)) = k$ then the cardinality of *MDTS* of G is $k+1$.

Particular special conditions on G have been found that ensure the existence of a Hamiltonian path on $L(G)$ [24], and therefore $HCN(L(G))$=0. Agnetis et al. [1] showed the *NP*-hardness of the problem of finding $HCN(L(G))$ even when G is bipartite, and proposed, for this case, a heuristic approach.

When G is a tree or a forest the problem may be solved in linear time [9,14,17,21,22], while an approximate algorithm for the weighted version of the problem was proposed by Wu et al. [26].

When G is an interval graph [18], a circular-arc graph [4], a block graph [23,25,27], a bipartite permutation graph [23] or a cograph [16], it was shown that there exist polynomial time algorithms for finding $HCN(G)$.

Raychaudhuri [19] presented a $O(n^5)$ algorithm for finding $HCN(G)$ when G is the line graph of a tree, while Agnetis et al. [2] proposed a linear algorithm for this case.

For Cactus graphs $C = (V, E)$, i.e. graphs such that every edge is part of at most one cycle in C, Hedetniemi et al. [12] proposed a linear algorithm for finding a minimum dominating set (i.e., a minimum cardinality subset $V^* \subseteq V$ such that every vertex in $V \setminus V^*$ is adjacent to at least one vertex in V^*). However, literature does not report any specific algorithm for finding *MDTS* or $HCN(L(G))$ on cactus graphs.

The study of line graphs is strongly related to important graph invariants, i.e. the *interval number*, the *total interval number* and the *Wiener index* [10,13,19]. Moreover, finding a *MDTS* or *HCN* of line graphs is often required in routing, sequencing, graph searching and in updating data structures [1,8]. In particular, the case of cactus graphs has several applications in efficient organization of control and data structures [6,7].

In this paper, a linear algorithm for finding the *Hamiltonian completion number* of the line graph $L(C)$ (as well as a *MDTS* of C) of a cactus C is proposed. In Section 2 some notations, definitions and elementary graph transformations are considered. These transformations will play an important role in the theoretical foundation presented in Section 3. In Section 4 an algorithm for finding *MDTS* is reported.

2. Notations and elementary graph transformations

A *cut vertex* in a graph G is a vertex whose removal results in a disconnected graph. A block in a graph G is a maximal connected subgraph having no cut vertices.

A cactus is a graph in which each block is either an edge or a cycle. Thus, a tree is a cactus in which each block is an edge. An endblock of a cactus is a block containing at most one cut vertex. A cactus may be recognized in linear time [5].

Throughout the paper we use the following notation. Given a node i of the graph G, we call $ad(i)$ the set of nodes adjacent to i in G, and $\delta(i)$ the cardinality of $ad(i)$ (i.e., the degree of i). Clearly, when i is a leaf, $ad(i)$ contains a single node j. In this case we write $ad(i)=j$ (instead of $ad(i)=\{j\}$). Given an endblock $B=(V_B,E_B)$ of G, we indicate as $cv(B)$ the unique cut vertex of B. If Q is a block of G, let $cv(Q)$ be the set of the cut vertices of Q.

Starting from a cactus graph $C=(V,E)$, our approach repeatedly applies some elementary operations which reduce the size of the graph, until an empty graph is reached. To describe these transformations, we refer to the notation proposed by Agnetis et al. [2], in which two marking functions $\mu: V \to \{0,1\}$ and $v: E \to \{0,1\}$ have been introduced. A node i (an edge e) such that $\mu(i)=1$ ($v(e)=1$) is called *marked*. Marking an edge means that we want to find a trail set which does *not* need to dominate that edge of the current graph. Marking a node means that at least one element of the trail set *must* pass through that node of the current graph. In the following, the problem of finding a *minimum constrained dominating trail set* (*MCDTS*) is defined.

Definition 1. Given a triple (C,μ,v), a *constrained dominating trail set* Σ_c is a collection of disjoint trails $\{t_1,t_2,\ldots,t_r\}$ such that: (i) Σ_c dominates all the edges of C which are not marked; (ii) for each marked node i, a trail $t \in \Sigma_c$ containing i exists. A *minimum constrained dominating trail set* (*MCDTS*) is a constrained dominating trail set of minimum cardinality. Such a cardinality will be denoted as $S(C,\mu,v)$.

Finding $HCN(L(C))$ can therefore be reformulated as the problem of finding $S(C,\mu,v)$, where $C=(V,E)$ is the original cactus, $\mu(i)=0$, $\forall i \in V$ and $v(e)=0$, $\forall e \in E$.

In the following some elementary transformations, employed in the proposed algorithm, are presented.

Definition 2. Given a triple (C,μ,v), let the edge $B=(i,j)$ be an endblock of C, i.e. node i is a leaf, and j is the cut vertex of B. By an *edge-shrink* of the endblock B we mean the transformation from (C,μ,v) to the triple (C',μ',v') defined as follows:

$$C'=(V',E')=(V\setminus i, E\setminus (i,j));$$

$$\mu'(q)=\mu(q), \quad \forall q \in (V'\setminus j);$$

$$\mu'(j)=1;$$

$$v'(e)=v(e), \quad \forall e \in E'.$$

In other words, given an edge endblock $B=(i,j)$, the *edge-shrink* transformation removes the leaf i and the edge (i,j) from the graph and marks the node j. A similar operation may be defined for cycle endblocks.

Definition 3. Given a triple (C, μ, v), let $B = (V_B, E_B)$ be a cycle endblock of C, and $cv(B)$ be the unique cut vertex of B. By a *cycle-shrink* of the endblock B we mean the transformation from (C, μ, v) to the triple (C', μ', v') defined as follows:

$$C' = (V', E') = (V \setminus (V_B \setminus cv(B)), E \setminus E_B);$$

$$\mu'(q) = \mu(q), \quad \forall q \in (V' \setminus cv(B));$$

$$\mu'(cv(B)) = 1;$$

$$v'(e) = v(e), \quad \forall e \in E'.$$

This operation removes a cycle endblock $B = (V_B, E_B)$, deleting from C all edges in E_B, and the nodes in $V_B \setminus cv(B)$ and marks $cv(B)$. In other words, a *cycle-shrink* operation collapses the cycle endblock in a single marked node. Note that, a path $p=(n_1, \ldots, n_k)$ on a triple (C', μ', v') resulting from the application of some *cycle-shrink* operations corresponds to a trail on the original triple (C, μ, v).

Another elementary operation is described by the following definition.

Definition 4. Given a triple (C, μ, v), $C = (V, E)$, let $(i, j) \in E$. The transformation that collapses the edge (i, j) in a single marked node, is called an *edge-collapse* of the edge (i, j).

In the first phase of the algorithm, the transformations *edge-shrink* and *edge-collapse* are employed in the following function.

function *preprocessing* $((C, \mu, v))$
begin
  while(a leaf i such that $\mu(i) = 0$ exists in C)
    edge-shrink $(i, ad(i))$;
  while(an edge $(i, j) \in E'$ exists such that $(\delta(i) \leqslant 2)$ **and** $(\delta(j) \leqslant 2)$)
    edge-collapse (i, j);
end

Function *preprocessing* consists of an iterative procedure, in which first all leaves i not marked are removed from C, and $\mu(ad(i))$ is set to 1. In fact, given a leaf i not marked, marking the node $ad(i)$ means that a trail t must pass in $ad(i)$. Hence, the edge $(i, ad(i))$ will be dominated by t, regardless if i belongs or not to t. In the second part of the function, an edge (i, j) is collapsed in a single marked node if i and j have both degree not greater than two. As consequence, if C is a path, this function transforms C in a single marked node. In Fig. 1, a cactus graph with marked nodes, and the resulting cactus after function *preprocessing* has been applied, are reported. Note that an optimal solution for the problem of finding $MCDTS$ on a triple (C, μ, v) transformed by operations of Definition 3, and by function *preprocessing* is also optimal for the original triple.

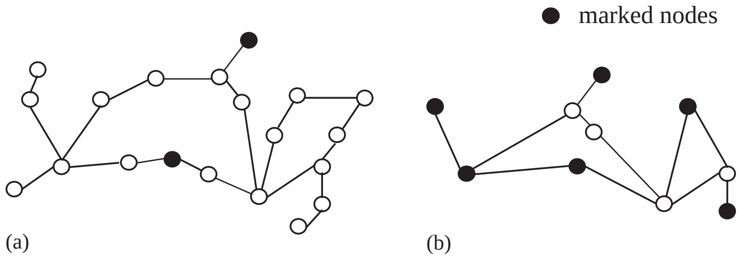

Fig. 1. (a) A cactus and (b) the cactus after the preprocessing phase.

3. Theoretical results

In this section the theoretical foundations of the algorithm reported in Section 4 are presented. We refer to the triple (C, μ, v), $C = (V, E)$, as the triple obtained applying the function *preprocessing* to the original triple (C_0, μ_0, v_0), $C_0 = (V_0, E_0)$, in which $\mu_0(i) = 0$, $\forall i \in V$ and $v_0(e) = 0$, $\forall e \in E_0$. Note that in C, all leaves are marked. In the following, we deal with the problem of finding $MCDTS$ on (C, μ, v).

The basic idea of the algorithm is to iteratively *process* the endblocks of the cactus C. An edge endblock $B = (i, j)$, with $j = cv(B)$, is *processed* by the following function *visit*.

function *visit* $((C, \mu, v), B = (i, j))$
begin
  edge-shrink(B);
  link j to i with a pointer.
end

This function removes the edge $B = (i, j)$, $j = cv(B)$, and the marked node i from (C, μ, v), marks the node j, and links the cut vertex j with a pointer to the removed node. A pointer from j to i means that a trail passing in i there exists in Σ_c. In the following, we say that a node j has a pointer if there exists a pointer from j to another node.

Since pointers are possibly associated by *visit* to some nodes, we classify the endblocks of a cactus into two sets $EB1$ and $EB2$. The set $EB1$ contains the endblocks $B = (V_B, E_B)$ whose nodes in $V_B \setminus cv(B)$ have no pointer. The second set $EB2$ contains endblocks $B = (V_B, E_B)$, in which nodes with pointers in $V_B \setminus cv(B)$ exist.

At each iteration of the algorithm, first the endblocks in $EB1$ are processed, and then the endblocks in $EB2$ are considered. In the following, results concerning endblocks of the set $EB2$ are presented. The Lemma 5 allows to build the trails of a minimum cardinality dominating trail set linking up two adjacent pointers that are associated to a node of an edge or a cycle endblock in $EB2$.

Lemma 5. *Consider a triple (C, μ, v), where C is a cactus not containing endblocks in the set $EB1$. Let $B = (V_B, E_B)$ be an endblock, let $n_1 \in V_B$ be a node having at* *least two pointers. Let l_1 and l_2 be two nodes pointed by n_1. Then, there exists a minimum constrained dominating trail set on (C, μ, v) containing a trail t starting in l_1 and ending in l_2.*

Proof. Let $\Sigma_c = \{t_1, \ldots, t_r\}$ be a constrained dominating trail set on (C, μ, v) of cardinality r, such that $t \notin \Sigma_c$. Since $\mu(l_1) = \mu(l_2) = 1$, then there exist in Σ_c two trails $t_q = (l_1, n_1, \ldots, n_q)$ and $t_k = (l_2, n_1, \ldots, n_k)$, starting in l_1 and l_2, respectively. Obviously, trails t_q and t_k have no edge in common. Removing t_q, t_k from Σ_c, and adding the trail $t = (l_1, n_1, l_2)$ and the trail $(n_q, \ldots, n_1, \ldots, n_k)$ to Σ_c, we obtain a new constrained dominating trail set Σ'_c with cardinality not greater than r. Hence, given any Σ_c, we can always find a Σ'_c containing t and such that $|\Sigma'_c| \leqslant |\Sigma_c|$. This is true also when Σ_c is a minimum cardinality dominating trail set and the thesis follows. □

From the previous lemma follows Corollary 6 that establishes how trails of a minimum constrained dominating trail set can be built, when at least three pointers are associated to a node of an endblock B.

Corollary 6. *Consider a triple (C, μ, v), where C is a cactus not containing endblocks in the set $EB1$. Let $B = (V_B, E_B)$ be an endblock, let $n_1 \in V_B$ be a node having at least three pointers. Let l_1 and l_2 be two nodes pointed by n_1. Then, there exists a minimum constrained dominating trail set on (C, μ, v) containing the trail $t = (l_1, n_1, l_2)$.*

Proof. Let $\Sigma_c = \{t_1, \ldots, t_r\}$ be a constrained dominating trail set on (C, μ, v) of cardinality r, such that $t \notin \Sigma_c$. Note that, since n_1 has at least three pointers, $t \notin \Sigma_c$ and $\mu(l_1) = \mu(l_2) = 1$, then at least two trails starting in l_1 and l_2 exist in Σ_c. Hence, similar arguments employed in the proof of Lemma 5 can be used to show that it is always possible to build a new constrained dominating trail set Σ'_c containing t, with cardinality not greater than r. □

When a node of an endblock has at least three pointers, Corollary 6 allows to build the trails of an optimal solution Σ_c linking up two adjacent pointers, as described in the following definition.

Definition 7. Given a triple (C, μ, v), let $B = (V_B, E_B)$ be an endblock of C, in which node n_1 satisfies conditions of Corollary 6. We call *transform1* of the endblock B the transformation from (C, μ, v) to the triple (C', μ', v') defined as follows:

$$C' = (V', E') = (V, E);$$

$$\mu'(q) = \mu(q), \ \forall q \in (V' \setminus n_1); \quad \mu'(n_1) = 0;$$

$$v'(e) = v(e), \quad \forall e \in (E' \setminus \{(i, n_1) \colon i \in ad(n_1)\});$$

$$v'(e) = 1 \quad \forall e \in E' \text{ incident to } n_1.$$

Clearly, $S(C',\mu',v')=S(C,\mu,v)-1$, where (C',μ',v') is the triple modified by operation *transform1* and $\Sigma_c=\Sigma'_c\cup(l_1,n_1,l_2)$.

Consider now a cycle endblock $B\in EB2$, such that only one node having pointers exists in B. The following lemma holds.

Lemma 8. *Consider a triple (C,μ,v), where C is a cactus not containing endblocks of the set $EB1$. Let $B=(V_B,E_B)$ be a cycle endblock, in which only one node $n_1\in B$ having pointers exists in the set $V_B\setminus cv(B)$. If one of the following cases holds*:

i. *n_1 has only one pointer to a node l_1 and only a marked node $n_2\notin\{n_1,cv(B)\}$ exists in B*;
ii. *n_1 has only one pointer to a node l_1 and no marked node exists in $B\setminus\{n_1,cv(B)\}$.*
iii. *n_1 has only one pointer to a node l_1 and two marked nodes $n_2,n_3\notin\{n_1,cv(B)\}$ exist in B*;
iv. *n_1 has two pointers to nodes l_1 and l_2*;

Then, a minimum constrained dominating trail set Σ_c^* exists in each case, such that:
In case i. *a trail t containing the trail $(l_1,n_1,n_2,cv(B))$ exists in Σ_c^*.*
In case ii. *a trail t containing the trail $(l_1,n_1,\ldots,cv(B))$ exists in Σ_c^*.*
In case iii. *a trail $t=(l_1,n_1,n_2,cv(B),n_3)$ exists in Σ_c^*.*
In case iv. *a trail $t=(l_1,n_1,n_2,cv(B),n_3,n_1,l_2)$ exists in Σ_c^*.*

Proof. Since n_1 is the only node in $V_B\setminus cv(B)$ having an associated pointer, the endblock B is composed by a cycle with at most four nodes.

Case i: Figs. 2(i1) and 2(i2) report the two possible situations for this case. Since in both the situations the trail $(l_1,n_1,n_2,cv(B))$ dominates all the edges of B, then it is possible to remove $B\setminus cv(B)$ from (C,μ,v) and linking $cv(B)$ to l_1 with a pointer (Fig. 2(i3)).

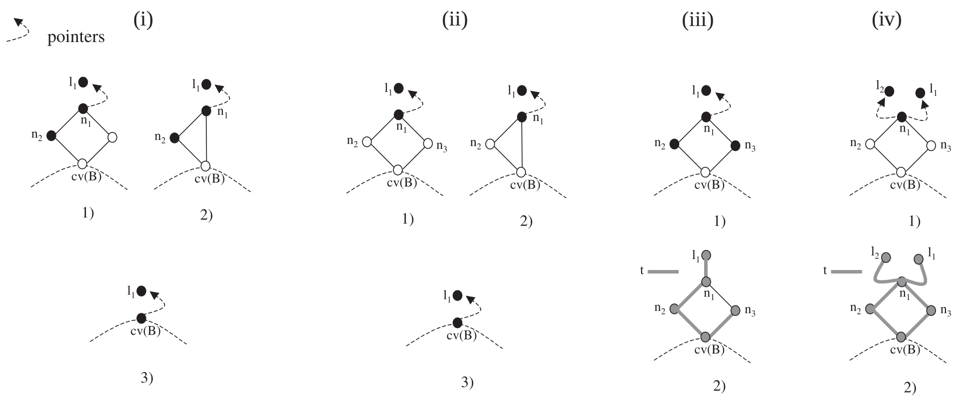

Fig. 2. Cases i–iv of Lemma 8.

Case ii: Similar arguments used in Case i. can be employed in this case. In Figs. 2(ii1) and 2(ii2) two possible situations of this case are reported. Note that since no marked nodes between n_1 and $cv(B)$ exists in B, in both the situations, two different trails dominating all the edges of B exist. Then it is possible to remove the $B \setminus cv(B)$ from (C,μ,v) and linking $cv(B)$ to l_1 with a pointer (Fig. 2(ii3)).

Case iii: Fig. 2(iii) shows this case and in Fig. 2(iii1) the dominating trail $t=(l_1,n_1,n_2,cv(B),n_3)$ is reported. Let $\Sigma_c=\{t_1,\dots,t_r\}$ be a constrained dominating trail set on (C,μ,v) of cardinality r, such that $t=(l_1,n_1,n_2,cv(B),n_3)\notin\Sigma_c$. Then, since nodes l_1, n_2 and n_3 are marked, at least two trails t_q and t_k there exist passing for these nodes in Σ_c. Without loss of generality we suppose $t_q=(l_1,n_1,\dots,n_q)$ and $t_k=(n_3,\dots,n_k)$. Suppose that a trail $t_i=(n_i,\dots,cv(B),\dots,n_j)\in\Sigma_c$ exists, such that $n_i\in B$ and $n_j\notin B$. We call it an *intersecting trail*. There are no more than two distinct intersecting trails in Σ_c, since each intersecting trail must pass either in n_2 or in n_3. Hence, only three cases are possible:

(a) t_q and t_k are not intersecting trails;
(b) either t_q or t_k is an intersecting trail;
(c) t_q and t_k are intersecting trails.

In each case we will show that a constrained dominating trail set Σ_c' exists containing t and such that $|\Sigma_c'|\leqslant|\Sigma_c|$.

In *case* (a), t_q and t_k are fully contained in t. Removing from Σ_c t_q,t_k and all the other trails strictly contained in t (if any) and adding the trail t, we obtain a new constrained dominating trail set Σ_c' with cardinality not greater than the cardinality of Σ_c.

In *case* (b), either $t_q=(l_1,n_1,n_2,cv(B),\dots,n_q)$, with $n_q\notin B$, or $t_k=(n_3,cv(B),\dots,n_k)$, with $n_k\notin B$, is the intersecting trail. If $t_q=(l_1,n_1,n_2,cv(B),\dots,n_q)$ is the intersecting trail, removing all the trails strictly contained in t and the trail t_q from Σ_c, and adding t and the trail $(cv(B),\dots,n_q)$ to Σ_c, we obtain a new constrained dominating trail set with cardinality not greater than r. If $t_k=(n_3,cv(B),\dots,n_k)$ is the intersecting trail, removing all the trails strictly contained in t and the trail t_k from Σ_c, and adding t and the trail $(cv(B),\dots,n_k)$ to Σ_c, we obtain a new constrained dominating trail set with cardinality not greater than r.

In the *case* (c), let $t_q=(l_1,n_1,n_2,cv(B),\dots,n_q)$ and $t_k=(n_3,cv(B),\dots,n_k)$ be the intersecting trails. Removing t_q,t_k from Σ_c, and adding t and the trail $(n_q,\dots,cv(B),\dots,n_k)$, we obtain a new constrained dominating trail set Σ_c' with cardinality not greater than r.

Case iv: If $t=(l_1,n_1,n_2,cv(B),n_3,n_1,l_2)\notin\Sigma_c$, since nodes l_1 and l_2 are marked, then at least two trails there exist passing for these nodes in Σ_c. Hence, similar arguments employed in *case* iii can be used also in this case. Note that nodes n_2 and n_3 can be marked or not. Fig. 2(iv) shows the situation of *case* iv, and in Fig. 2(iv1) the dominating trail $t=(l_1,n_1,n_2,cv(B),n_3,n_1,l_2)$ is reported.

Hence, given any Σ_c, we can always find a Σ_c' containing t and such that $|\Sigma_c'|\leqslant|\Sigma_c|$. This is true also when Σ_c is a minimum cardinality dominating trail set and the thesis follows. Note that in all the four cases, the lemma is still valid without regarding whether n_1 and/or $cv(B)$ are marked or not. □

In force of Lemma 8, the following transformation can be considered.

Definition 9. Given a triple (C,μ,v), let $B=(V_B,E_B)$ be a cycle endblock of C of Lemma 8. By *transform2* of the endblock B we mean the transformation from (C,μ,v) to the triple (C',μ',v') defined as follows:

in *cases* i and ii of Lemma 8 then:

$$C'=(V',E')=(V\setminus(V_B\setminus cv(B)),E\setminus E_B);$$

$$\mu'(q)=\mu(q),\quad \forall q\in V';$$

link $cv(B)$ with a pointer to l_1;

in *cases* iii and iv of Lemma 8 then:

$$C'=(V',E')=(V\setminus(V_B\setminus cv(B)),E\setminus E_B);$$

$$\mu'(q)=\mu(q),\ \forall q\in(V'\setminus cv(B));\quad \mu'(cv(B))=0;$$

$$v'(e)=v(e),\quad \forall e\in(E'\setminus\{(i,cv(B)):\ i\in ad(cv(B))\});$$

$$v'(e)=1,\quad \forall e\in E' \text{ incident to } cv(B).$$

Note that *transform2* can be applied also when $B=(n_1,cv(B))$ is an edge endblock, in which n_1 has one or to two pointers. It is easy to see that these cases are similar to *case* i and *case* iv, respectively, of Lemma 8. In the algorithm, we will apply *transform2* also to these situations. Clearly, an optimal solution for $MCDTS$ on a triple (C',μ',v') modified by operation *transform2*, is also optimal for the original triple in *cases* i and ii; while $S(C',\mu',v')=S(C,\mu,v)-1$ in *cases* iii and iv.

Let us consider now a cycle endblock $B=(V_B,E_B)$ in which more than one node has pointers. The following three cases will be considered:

(α) a path $p=(n_1,n_2,n_3)$ exists in B, in which both nodes n_1 and n_3 have at least one pointer, node n_2 is marked and has no pointers, $n_1,n_2,n_3\neq cv(B)$;

(β) a marked node without pointers adjacent to $cv(B)$ exists in B;

(γ) no node without pointers in $V_B\setminus cv(B)$ is marked.

In *case* α, Lemma 10 shows how trails of an optimal solution can be found.

Lemma 10. *Consider a triple (C,μ,v), where C is a cactus not containing endblocks of set $EB1$. Let $B=(V_B,E_B)$ be a cycle endblock of case α. Let (n_1,n_2,n_3) be a trail in B, in which nodes n_1 and n_3 have at least one pointer, and node $n_2\neq cv(B)$ is marked. Let l_1 and l_3 the nodes pointed by n_1 and n_3, respectively. Then, there exists a minimum constrained dominating trail set on (C,μ,v) containing the trail $t=(l_1,n_1,n_2,n_3,l_3)$.*

Proof. In Fig. 3(1) the cycle endblock B and the trail $t=(l_1,n_1,n_2,n_3,l_3)$ is shown. Let $\Sigma_c=\{t_1,\dots,t_r\}$ be a constrained dominating trail set on (C,μ,v) of cardinality r, such that $t\notin\Sigma_c$. Suppose that a trail $t_i\in\Sigma_c$ exists, which contains at least one edge of t and at least one edge of C that is not in t. We call t_i *intersecting trail.*

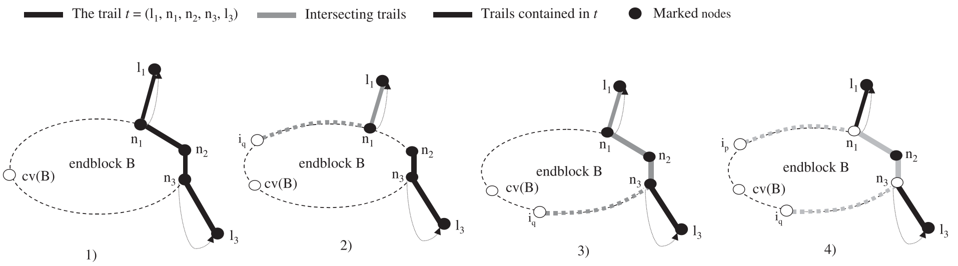

Fig. 3. The trail $t=(l_1,n_1,n_2,n_3,l_3)$, and intersecting trails of t.

There are no more than two distinct intersecting trails in Σ_c, since each intersecting trail must pass either in n_1 or in n_3. Hence, only three cases are possible:

(i) there are no intersecting trails in Σ_c;
(ii) there is exactly one intersecting trail in Σ_c;
(iii) there are two intersecting trails in Σ_c.

In each case we will show that a constrained dominating trail set Σ_c' exists containing t such that $|\Sigma_c'| \leqslant |\Sigma_c|$.

In the *case* (i), since l_1 and l_3 are two pointed nodes, and $\mu(n_2)=1$, in Σ_c there must be at least one trail fully contained in t. Removing from Σ_c all trails fully contained in t and adding the trail t, we obtain a new constrained dominating trail set Σ_c' with cardinality not greater than $|\Sigma_c|$.

In the *case* (ii), let $(i_p,\ldots,i_q)$ be the unique intersecting trail. Obviously, it cannot contain both l_1 and l_3. Since l_1 and l_3 are two pointed nodes, and $\mu(n_2)=1$, without loss of generality, we have three possible intersecting trails:

(1) $p_1=(l_1,n_1,\ldots,i_q),\ i_q\notin t$;
(2) $p_2=(l_1,n_1,n_2,n_3,\ldots,i_q),\ i_q\notin t$;
(3) $p_3=(i_p,\ldots,n_1,n_2,n_3,\ldots,i_q),\ i_p,i_q\notin t$;

In Figs. 3.2, 3.3 and 3.4, the intersecting trails p_1, p_2 and p_3 are, respectively, reported. In *case* (1), since l_3 is a pointed node and $\mu(n_2)=1$, then at least another trail $p_q\subset t$ there exists passing in n_2 and in l_3 in Σ_c. Removing from Σ_c the trails p_1 and p_q, and adding t and the trails $(n_1,\ldots,i_q)$ to Σ_c, we obtain a new constrained dominating trail set containing t, with cardinality not greater than r.

In the *case* (2), since l_3 is a pointed node, then at least another trail $p_q\subset t$ there exists passing in l_3 in Σ_c. Removing from Σ_c the trails p_2 and p_q, and adding t and the trail $(n_3,\ldots,i_q)$ to Σ_c, we obtain a new constrained dominating trail set with cardinality not greater than r.

In *case* (3), since l_1 and l_3 are two pointed nodes, then two trails p_h and p_k there exist in Σ_c, passing in l_1 and l_3, respectively. Removing from Σ_c the trails p_3, p_h and p_k, and adding t and the trails $(i_p,\ldots,n_1)$ and $(n_3,\ldots,i_q)$ to Σ_c, we obtain a new constrained dominating trail set with cardinality not greater than r.

If two intersecting trails exist in Σ_c (*case* (iii)), since l_1 and l_3 are two pointed nodes and $\mu(n_2)=1$, without loss of generality, we may have three possible intersecting trails pairs:

(a) $a_1=(l_1,n_1,\ldots,i_q)$, $a_2=(l_3,n_3,\ldots,i_t)$, $i_q,\ i_t\notin t$;
(b) $b_1=(l_1,n_1,\ldots,i_q)$, $b_2=(n_2,n_3,\ldots,i_t)$, $i_q,i_t\notin t$;
(c) $c_1=(n_2,n_1,\ldots,i_q)$, $c_2=(l_3,n_3,\ldots,i_t)$, $i_q,i_t\notin t$;

In *case* (a), since $\mu(n_2)=1$, then another trail $p_q\subset t$ there exists passing in n_2 in Σ_c. Removing from Σ_c the trails a_1,a_2 and p_q, and adding t and the trails $(n_1,\ldots,i_q)$ and $(n_3,\ldots,i_t)$ to Σ_c, we obtain a new constrained dominating trail set with cardinality not greater than r.

In *case* (b), since l_3 is a pointed node, then another trail p_q there exists passing in l_3 in Σ_c. Removing from Σ_c the trails b_1, b_2 and p_q, and adding t and the trails $(n_1,\ldots,i_q)$ and $(n_3,\ldots,i_t)$ to Σ_c, we obtain a new constrained dominating trail set with cardinality not greater than r.

The *case* (c) is symmetric to *case* (b).

Hence, given any Σ_c, we can always find a Σ'_c containing t and such that $|\Sigma'_c|\leqslant|\Sigma_c|$. This is true also when Σ_c is a minimum cardinality dominating trail set and the thesis follows. □

If conditions of Lemma 10 holds, a *MCDTS* Σ_c containing the trail t exists. The trail t can be removed from the triple (C,μ,v) and added to Σ_c. As consequence, the endblock B results *opened*. Note that this *opened* endblock can be considered as a tree rooted in the node $cv(B)$. More formally, the following transformation can be defined.

Definition 11. Given a triple (C,μ,v), let $B=(V_B,E_B)$ be a cycle endblock and $t=(l_1,n_1,n_2,n_3,l_3)$ the trail of C of Lemma 10. By *open1* of the endblock B we mean the transformation from (C,μ,v) to the triple (C',μ',v') defined as follows:

$$C'=(V',E')=(V\setminus n_2,E\setminus\{(n_1,n_2),(n_2,n_3)\});$$

$$\mu'(q)=\mu(q),\ \forall q\in(V'\setminus\{n_1,n_3\});\qquad \mu'(n_1)=\mu'(n_3)=0;$$

$$v'(e)=v(e),\quad \forall e\in(E'\setminus\{(n_1,n_2),(n_2,n_3)\});$$

$$v'(e)=1,\quad \forall e\in E' \text{ incident to } n_1 \text{ and } n_3.$$

By Definition 11 follows $S(C',\mu',v')=S(C,\mu,v)-1$ and $\Sigma_c=\Sigma'_c\cup t$, where (C',μ',v') is the triple after the Transformation *open1* has been applied.

Lemma 12. *Consider a triple (C,μ,v), where C does not contain endblocks of set $EB1$. Let $B=(V_B,E_B)$ be a cycle endblock of case β. Let $(n_1,n_2,cv(B))$ be a trail in B, in which the node n_1 has a pointer to the node l_1, n_2 is marked and has no pointer. Then, there exists a minimum constrained dominating trail set Σ_c on (C,μ,v), in which a trail $t\in\Sigma_c$ that contains $p=(l_1,n_1,n_2,cv(B))$ exists.*

Proof. Let $\Sigma_c = \{t_1, \dots, t_r\}$ be a constrained dominating trail set on (C, μ, v) of cardinality r not containing a trail $t \supset p$. Let $p_1 = (l_1, n_1)$ be the trail connecting n_1 to the pointed node l_1. Only two cases are possible:

(i) the trail $p_1 \in \Sigma_c$;
(ii) a trail containing p_1 exists in Σ_c. In each case, we will show that a constrained dominating trail set Σ'_c containing t and such that $|\Sigma'_c| \leqslant |\Sigma_c|$ exists.

In *case* (i), let $t_y = (n_k, \dots, n_2, \dots, n_q)$ of Σ_c be the trail passing in the node n_2. Without loss of generality, suppose that $cv(B) \in (n_2, \dots, n_q)$. If the node $n_1 \in t_y$, then $n_1 \in (n_k, \dots, n_2)$. In this case, removing p_1 and t_y from Σ_c, and adding to Σ_c the trail $(l_1, n_1, n_2, \dots, n_q)$ and the trail $(n_k, \dots, n_1)$, we obtain a new constrained dominating trail set Σ'_c with cardinality not greater than r. Note that trail $(n_k, \dots, n_1)$ can be empty (i.e., when $(n_k = n_1)$). Otherwise, if $n_1 \notin t_y$ then n_2 necessarily is an endpoint of t_y. Hence, $n_2 = n_k$. Removing p_1 and t_y from Σ_c, and adding the trail $(l_1, n_1, n_2, \dots, n_q)$, we obtain a new constrained dominating trail set Σ'_c with cardinality $r - 1$.

In *case* (ii), let $t_y = (l_1, n_1, \dots, n_q)$ of Σ_c be the trail containing the trail p_1. If $n_2 \notin t_y$, since $\mu(n_2) = 1$, a trail t_z containing node n_2 there exists. Removing t_y and t_z from Σ_c, and adding the trail $p_1 \cup t_z$ and the trail $(n_1, \dots, n_q)$, we obtain a new constrained dominating trail set Σ'_c with cardinality not greater than r.

Let us now suppose $n_2 \in t_y$. Since $t \notin \Sigma_c$, if the edge $e = (n_2, cv(B))$ is contained in a trail $t_e \neq t_y$ of Σ_c then e is an extreme edge of this trail ($t_e = (n_2, cv(B), \dots, n_w)$). Removing t_y and t_e from Σ_c, and adding the trail $t_y \cup e = p$ and the trail $(cv(B), \dots, n_w)$, we obtain a new constrained dominating trail set Σ'_c with cardinality not greater than r. Otherwise, if the edge $e \notin t_i$, $\forall t_i \in \Sigma_c$, removing t_y from Σ_c, and adding the trail $t_y \cup e = p$, we obtain a new constrained dominating trail set Σ'_c with cardinality not greater than r.

Hence, given any Σ_c, we can always find a Σ'_c containing the trail t and such that $|\Sigma'_c| \leqslant |\Sigma_c|$. This is true also when Σ_c is a minimum cardinality dominating trail set and the thesis follows. □

In force of Lemma 12, the trail $p = (l_1, n_1, n_2, cv(B))$ can be replaced with a pointer from $cv(B)$ to l_1. This transformation is described in the following definition.

Definition 13. Given a triple (C, μ, v), let $B = (V_B, E_B)$ be a cycle endblock and $p = (l_1, n_1, n_2, cv(B))$ the trail considered in Lemma 12. By *open2* of the endblock B we mean the transformation from (C, μ, v) to the triple (C', μ', v') defined as follows:

$$C' = (V', E') = (V \setminus (p \setminus \{l_1, cv(B)\}), E \setminus \{(n_1, n_2), (n_2, cv(B))\});$$

$$\mu'(q) = \mu(q), \quad \forall q \in V';$$

$$v'(e) = v(e), \quad \forall e \in E';$$

link $cv(B)$ with a pointer to l_1.

An optimal solution for *MCDTS* on a triple (C', μ', v') modified by operation *open2*, is also optimal for the original triple (C, μ, v). Note that, also in this case, the endblock B is transformed in a tree rooted in $cv(B)$.

Let B be a cycle endblock of *case* γ, containing nodes n_i, $i=1,\dots,k$, having pointers, and let l_i be a node pointed by n_i. Since $\mu(l_i)=1$, trails starting in l_i, $i=1,\dots,k$, there exist in the *MCDTS* Σ_c. Hence, trails that join together two of this pointed nodes (say, l_i and l_k) by means of edges of the endblock B can be considered. In particular if B has ρ nodes labeled with a pointer, then $\rho/2$ is the minimum numbers of trails that we need to dominate B, when ρ is even; otherwise $\rho/2+1$ trails occur. The following lemma shows how these trails can be constructed.

Lemma 14. *Given a triple* (C, μ, v), *where* C *is a cactus containing no endblocks of set* $EB1$. *Let* $B=(V_B, E_B)$ *be a cycle endblock of case* γ *containing at least two nodes having pointers. Let* $n_1, n_2 \in B$ *be two of these nodes such that either* (*case* (a)) n_1 *and* n_2 *are adjacent or* (*case* (b)) *a node* n_x *without pointers exists in* B *adjacent to* n_1 *and* n_2 (*i.e., such that* B *contains the trail* $p_x=(n_1, n_x, n_2)$). *Let* l_1 *and* l_2 *be nodes pointed respectively by nodes* n_1 *and* n_2, *and let* $p_1=(l_1, n_1)$ *and* $p_2=(l_2, n_2)$.

Then, a minimum constrained dominating trail set Σ_c *on* (C, μ, v) *exists, that does not contain*:

(1) *both the trails* p_1 *and* p_2;

or

(2) *the trail* $t_1=p_1$ *and a trail* $t_2=(l_2,\dots,n_1,\dots,n_q)$, *in which* $n_q \neq l_1$, *or, vice versa, the trail* $t_2=p_2$ *and a trail* $t_1=(l_1,\dots,n_2,\dots,n_q)$, *in which* $n_q \neq l_2$.

Proof. Let $\Sigma_c=\{t_1,\dots,t_r\}$ be a constrained dominating trail set on (C, μ, v) of cardinality r containing the trails $t_1=p_1$ and $t_2=p_2$ as in (1).

Let $t_3=(n_q,\dots,n_1,\dots,n_2,\dots,n_k) \notin \{t_2, t_1\}$ be the trail of Σ_c (if exists) containing the edge (n_1, n_2) in *case* (a), or the edges (n_1, n_x) and (n_x, n_2) in *case* (b). Note that may be either $n_q \equiv n_1$ or $n_k \equiv n_2$. Then, we may construct a new constrained dominating trail set Σ_c' by removing t_1, t_2 and t_3 (if exists) from Σ_c, and adding to Σ_c the trail $p_t=(l_1, n_1, n_2, l_2)$ in *case* (a), $p_t=(l_1, n_1, n_x, n_2, l_2)$ in *case* (b), and the trails $t_4=(n_q,\dots,n_1)$ and $t_5=(n_2,\dots,n_k)$ (if t_3 exists). Clearly, $|\Sigma_c'| \leqslant |\Sigma_c|$.

Let now $\Sigma_c=\{t_1,\dots,t_r\}$ be a constrained dominating trail set on (C, μ, v) of cardinality r containing the trails t_1 and t_2 as in (2). Without loss of generality, let $t_1=p_1$ and $t_2=(l_2,\dots,n_1,\dots,n_q)$. Then, we may construct a set Σ_c' removing t_1 and t_2 from Σ_c, and adding the trails $(l_2, n_2,\dots,n_1, l_1)$ and $(n_1,\dots,n_q)$. Clearly, $|\Sigma_c'|=|\Sigma_c|$. □

The previous lemma allows to consider a *MCDTS* for an endblock $B=(V_B, E_B)$ of *case* γ (i.e., having no marked nodes in $V_B \setminus cv(B)$), in which dominating trails are obtained joining a pair of nodes, as n_1 and n_2 (i.e. l_1 and l_2) of Lemma 14. Let $\Sigma_B=\{t_1,\dots,t_k\}$ be the set of trails that dominates B. In order to construct a *MCDTS* for the whole triple (C, μ, v), it is useful that one trail of Σ_B passes in $cv(B)$. This is always possible according to Lemma 14. Hence, the following corollary trivially holds.

Corollary 15. *Consider a triple* (C,μ,v), *where* C *is a cactus not containing endblocks of set* $EB1$. *Let* $B=(V_B,E_B)$ *be a cycle endblock of case* γ. *Let* $p_h=(l_1,n_1,\ldots,c_v(B))$ *be a trail, in which* $n_1\in B$ *has a pointer to the node* $l_1, c_v(B)$ *is the cut vertex of* B, *and such that between nodes* n_1 *and* $cv(B)$ *at most one node of* B *with no pointers exists. Then, there exists a minimum constrained dominating trail set* Σ_c *on* (C,μ,v), *such that a trail* $t\in\Sigma_c$ *containing the trail* p_h *exists.*

From Lemma 14 and Corollary 15, the Transformation *open3* can be defined as follows.

Definition 16. Given a triple (C,μ,v), let $B=(V_B,E_B)$ be a cycle endblock and $p_h=(l_1,n_1,\ldots,c_v(B))$ be the trail considered Corollary 15. By *open3* of the endblock B we mean the transformation from (C,μ,v) to the triple (C',μ',v') defined as follows:

$$C'=(V',E')=(V\setminus\{n_1,\ldots,c_v(B)\},E\setminus\{e:\ e\in\{p_h\setminus(l_1,n_1)\}\});$$

$$\mu'(q)=\mu(q),\quad \forall q\in V';$$

$$v'(e)=v(e),\quad \forall e\in E';$$

link $cv(B)$ with a pointer to l_1.

An optimal solution for *MCDTS* on a triple (C',μ',v') modified by Transformation *open3*, is also optimal for the original triple (C,μ,v). The endblock B results open and is transformed in a tree rooted in $cv(B)$.

4. Finding a *MDTS* of a cactus *C* and *HCN*(*L*(*C*)) in linear time

In this section, a linear algorithm for *MDTS* on a cactus is presented. The algorithm, called DOMCACTUS, is reported in Fig. 4 and consists of four different phases.

In the first phase, the function *preprocessing* is applied on (C,μ,v), where $C=(V,E)$ is the original cactus graph, $\mu(i)=0\ \forall i\in V$, $v(i,j)=0\ \forall(i,j)\in E$. Note that, after the preprocessing, all leaves of the cactus are marked. The first phase is applied once in the algorithm and requires a linear time.

In the second phase, the blocks of the cactus are individuated, employing the linear algorithm proposed by Aho et al. [3] (addressed as *AHU procedure* in what follows) for finding the biconnected components of a graph. This algorithm uses a stack structure to store biconnected components of a graph. When applied to a cactus graph C, the *AHU* procedure provides an endblock B of C at the top of the stack. On the removal of B from C and from the stack, the *AHU* procedure behaves exactly as it would on the graph $C'=C\setminus B$, and another endblock $B'\in C'$ appears at the top of the stack structure. The *AHU* procedure allows to easily obtain all information about blocks and

Algorithm DOMCACTUS

Input: A cactus graph $C = (V, E)$;

Output: A minimum cardinality trail set Σ;

Initialize triple (C, μ, v): $\mu(i) = 0\ \forall i \in V,\ v(i,j) = 0\ \forall (i,j) \in E$
$\Sigma = \emptyset$;
(phase 1)
preprocessing (C, μ, v);
(phase 2)
Find the set A of blocks of C employing the *AHU* procedure.
Let $EB \subset A$ be the endblock set of C.
$EB1 = EB$, $EB2 = \emptyset$
while $((C, \mu, v) \neq \emptyset)$ **do**
begin
(phase 3)
  while ($EB1$ contains a cycle endblock $B = (V_B, E_B)$) **do**
  begin
    if (the edges of B are all marked) **and** (no marked nodes are contained in $V_B \setminus cv(B)$)
    then $C = C \setminus (B \setminus cv(B))$;
    else *cycle-shrink*(B);
    update $EB1$ and $EB2$;
  end
  while ($EB1$ contains an edge endblock B) **do**
  begin
    Let $B = (i,j)$; let $cv(B) = j$;
    visit(B);
    update $EB1$ and $EB2$;
  end
(phase 4)
  if ($EB2$ is not empty) **do**
  begin
    Let $B \in EB2$;
    Dominate-EB2$((C, \mu, v), B, \Sigma)$;
    update $EB1$ and $EB2$;
  end
  if (C is a single marked node i) **then**
  begin
    the last dominating trail $t = (i)$ has been found;
    $\Sigma = \Sigma \cup t$; $C = C \setminus i$;
  end
end

Fig. 4. Algorithm DOMCACTUS.

cut vertex of a cactus. These data will play an important role in the DOMCACTUS algorithm.

Phases 3 and 4 are iteratively performed until an empty graph is obtained.

In Phase 3, first the *cycle-shrink* operation introduced in Definition 3 is applied on all the cycle endblocks of the set $EB1$, then the edge endblocks are processed, by the *visit* procedure introduced in Section 3. During this phase, new endblocks that could be inserted in the sets $EB1$ and $EB2$ could generated. In the algorithm, the updating operation of these sets is indicated as *update EB1 and EB2*. This instruction, that will be described in the following, removes from $EB1$ and $EB2$ the processed endblocks, and possibly adds new endblocks that have been generated.

In Phase 4, either an edge endblock or a cycle endblock of $EB2$ is analyzed according to Lemmas 5, 8, 10, 12 and 14 and Corollaries 6 and 15. Phase 4 calls the function *Dominate-EB2*, reported in Fig. 5. In *Dominate-EB2*, if $B=(i,j), cv(B)=j$, is an edge endblock such that the node i has pointers then *transform1* is applied, until at most one pointer remains. Then *transform2* is employed. Note that edge (i,j) is dominated and can be eliminated from C. If B is an cycle endblock, *transform1* is applied until no node with more than three pointers exists in B. Then either *transform1* is employed or the cycle is opened by *open1*, *open2* or *open3*. In particular, if B is opened, the linear algorithm presented by Agnetis et al. [2] is used to dominate the tree rooted in $cv(B)$. Note that *Dominate-EB2* may generate new endblocks, and hence the sets $EB1$ and $EB2$ must be updated at the end of this function.

The whole algorithm DOMCACTUS can be implemented to run in linear time. At this aim, the following data structure allows to use the AHU procedure once in the overall algorithm. Each block has associated the number of its cut vertices. The blocks are partitioned in three sets $EB1$, $EB2$ and other blocks. Set $EB1$ is stored in a list partitioned in two subsets containing the cycle endblocks and the edge endblocks, respectively. Note that, unless C is empty, $EB1 \cup EB2 \neq \emptyset$.

We consider a vector CV dedicated to vertices, in which a component $CV(i)$, $i \in V$, contains the number of blocks having i as cut vertex ($CV(i)=0$ if i is not a cut vertex). Another vector EBL is associated to edges. Every component $EBL(e)$, $e \in E$, indicates the block B in which the edge e is contained. The vectors CV and EBL are initialized in Phase 2, starting from the stack structure provided by the AHU procedure. This initialization requires linear time.

Every time that an endblock B is removed from the current graph C, say k its unique cut vertex, CV is updated as follows: $CV(k)=CV(k)-1$. If $CV(k)=0$, k is not even a cut vertex. In this case, we can find the unique block Q which contains k by using the vector EBL. Let $PT(k)$ be the number of pointers of the node k, and let $PT(Q)$ be the number of pointers associated to the nodes of $Q \setminus cv(Q)$, if $CV(k)=0$ we set $PT(Q)=PT(Q)+PT(k)$. If the block Q has only one cut vertex, it is immediately put in the set $EB1$ or $EB2$ according to the number of its pointers (the rest of stored data remains the same). Hence, every time an endblock is removed the vector CV and the sets $EB1$ and $EB2$ can be updated in constant time.

The previous data structure allows to efficiently implement the algorithm DOMCACTUS (in particular, the operation *update EB1 and EB2*) and the function *Dominate-EB2*. Note that, processing a block $B=(V_B,E_B)$ by *Dominate-EB2* requires $O(|E_B|)$ time. Since *Dominate-EB2* removes B from the graph and each block is processed once, *Dominate-EB2* runs in $O(|E|)$, in the overall algorithm. Then the DOMCACTUS algorithm finds a $MDTS$ on a cactus C and the $HCN(L(C))$ in linear time.

```
function Dominate-EB2((C, μ, ν),B,Σ)
begin
    if (B = (i,j), cv(B) = j) then
    begin
        while (i has at least three pointers (Corollary 6)) do transform1 B;
        if (node i has one or two pointers) then transform2 B;
        eliminate edge (i,j) from C;
    end
    if (B is a cycle endblock) then
    begin
        while(B falls in Corollary 6) do transform1 B;
        if (B falls in Lemma 8) then transform2 B;
        if (B falls in case of Lemma 10) then open1 B;
        if (B falls in case of Lemma 12) then open2 B;
        if (B falls in cases of Lemma 14 and Corollary 15) then open3 B;
        if (B has been opened) then
        begin
            Let T = (V_T, E_T) the tree rooted in cv(B);
            Let (T, μ, ν) be the triple associated to T.
            Find a minimum dominating path set Σ_T = {p_1, ..., p_k} on (T, μ, ν);
            if (a path p_i = (n_1, ..., n_2) passing in cv(B) exists in Σ_T) then
            begin
                Σ = Σ ∪ (Σ_T \ p_i);
                link cv(B) with a pointer to n_1 if n_1 ≠ cv(B);
                link cv(B) with a pointer to n_2 if n_2 ≠ cv(B);
            end
        end
    end
    while (cv(B) has at least three pointers (Corollary 6)) do
    begin
        let l_1 and l_2 be two nodes pointed by cv(B);
        a dominating trail t = (l_1, cv(B), l_2) has been found;
        remove the pointers to l_1 and l_2 from cv(B);
        Σ = Σ ∪ t;
        mark as dominated the edges incident to cv(B);
    end
end
```

Fig. 5. The function Dominate-EB2.

Acknowledgements

We would like to thank prof. Z. Skupień for his personal comments and suggestions on this work and two anonymous referees for their helpful comments.

References

[1] A. Agnetis, P. Detti, C. Meloni, D. Pacciarelli, Set-up coordination between two stages of a supply chain, Ann. Oper. Res. 107 (2001) 15–32.
[2] A. Agnetis, P. Detti, C. Meloni, D. Pacciarelli, A linear algorithm for the Hamiltonian completion number of the line graph of a tree, Inform. Process. Lett. 79 (2001) 17–24.
[3] A.H. Aho, J.E. Hopcroft, J.D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, 1974.
[4] M.A. Bonuccelli, D.P. Bovet, Minimum node disjoint path covering for circular-arc graphs, Inform. Process. Lett. 8 (4) (1979) 159–161.
[5] A. Brandstädt, V.B. Le, J.P. Spinard, Graph classes, a survey, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 1999.
[6] A. De Vitis, The cactus representation of all minimum cuts in weighted graph, Technical Report, 454, IASI-CNR, Roma, 1997.
[7] L. Fleischer, Building chain and cactus representations of all minimum cuts from Hao–Orlin in the same asymptotic run time, J. Algorithms 33 (1) (1999) 51–72.
[8] F.V. Fomin, P.A. Golovach, Graph searching and interval completion, SIAM J. Discrete Math. 13 (4) (2000) 454–464.
[9] S.E. Goodman, S.T. Hedetniemi, P.J. Slater, Advances on the Hamiltonian completion problem, J. ACM 22 (3) (1975) 352–360.
[10] I. Gutman, Buckley-type relations for Wiener-type structure-descriptors, J. Serbian Chem. Soc. 63 (7) (1998) 491–496.
[11] F. Harary, C.St.J.A. Nash-Williams, On Eulerian and Hamiltonian graphs and line-graphs, Canad. Math. Bull. 8 (1965) 701–709.
[12] S.T. Hedetniemi, R.C. Laskar, J. Pfaff, A linear algorithm for finding a minimum dominating set in a cactus, Discrete Appl. Math. 13 (1986) 287–292.
[13] S. Klavzar, I. Gutman, Wiener number of vertex-weighted graphs and a chemical application, Discrete Appl. Math. 80 (1) (1997) 73–81.
[14] S. Kundu, A linear algorithm for the Hamiltonian completion number of a tree, Inform. Process. Lett. 5 (1976) 55–57.
[15] P.G.H. Lehot, An optimal algorithm to detect a line graph and output its root graph, J. ACM 21 (1974) 569–575.
[16] R. Lin, S. Olariu, G. Pruesse, An optimal path cover algorithm for cographs, Comput. Math. Appl. 30 (8) (1995) 75–83.
[17] J. Misra, R.E. Tarjan, Optimal chain partitions of trees, Inform. Process. Lett. 4 (1) (1975) 24–26.
[18] S. Rao Arikati, C. Pandu Rangan, Linear algorithm for optimal path cover problem on interval graphs, Inform. Process. Lett. 35 (1990) 149–153.
[19] A. Raychaudhuri, The total interval number of a tree and the Hamiltonian completion number of its line graph, Inform. Process. Lett. 56 (1995) 299–306.
[20] N.D. Roussopoulos, A max $\{m, n\}$ algorithm for determining the graph H from its line graph G, Inform. Process. Lett. 2 (1973) 108–112.
[21] Z. Skupień, Path Partitions of Vertices and Hamiltonity of Graphs, in: M. Fiedler (Ed.), Recent Advances in Graph Theory (Proceedings of the 2nd Czechoslovakian Symposium on Graph Theory, Prague 1974), Akademia, Praha, 1975, pp. 481–491.
[22] Z. Skupień, Hamiltonian shortage, path partitions of vertices, and matchings in a graph, Colloq. Math. 36 (2) (1976) 305–318.

[23] R. Srikant, R. Sundaram, K.S. Singh, C. Pandu Rangan, Optimal path cover problem on block graphs and bipartite permutation graphs, Theoret. Comput. Sci. 115 (1993) 351–357.

[24] H.J. Veldman, A result of Hamiltonian line graphs involving restrictions on induced subgraphs, J. Graph Theory (12) 3 (1988) 413–420.

[25] P.K. Wong, Optimal path cover problem on block graphs, Theoret. Comput. Sci. 225 (1999) 163–169.

[26] Q.S. Wu, C.L. Lu, R.C.T. Lee, An approximate algorithm for the weighted Hamiltonian path completion problem on a tree, in: D.T. Lee, S.H. Teng (Eds.), Proceedings of Eleventh Annual International Symposium on Algorithms and Computation (ISAAC 2000), Lecture Notes in Computer Science, Springer, Berlin, 1969, pp. 156–167.

[27] J.H. Yan, G.J. Chang, The path-partition problem in block graphs, Inform. Process. Lett. 52 (1994) 317–322.