Optimal covering of cacti by vertex-disjoint paths

Shlomo Moran and Yaron Wolfstahl
Faculty of Computer Science, Technion, Israel Institute of Technology, Haifa, 32000, Israel

Communicated by G. Ausiello
Received March 1988
Revised April 1989

Abstract

A path cover (or in short: cover) of a graph G is a set of vertex-disjoint paths which cover all the vertices of G. An optimal cover of G is a cover of the smallest possible cardinality. Notable applications of graph covering are code optimization and mapping parallel programs to parallel architectures. The optimal covering problem is known to be NP-complete even for cubic 3-connected planar graphs. Motivated by the intractability of this problem, we develop an efficient optimal covering algorithm for cacti (i.e. graphs where no edge lies on more than one cycle). In doing so we generalize the results of Boesch, Gimpel, McHugh (1974) and of Pinter, Wolfstahl (1987) where optimal covering algorithms for trees and graphs where no two cycles share a vertex were presented.

1. Introduction

Let G = (V_G, E_G) be an undirected graph with no self loops or parallel edges. A path in G is either a single vertex v \in V_G or a sequence of distinct vertices (v_1, v_2, \ldots, v_k) where for 1 \leq i \leq k - 1, (v_i, v_{i+1}) \in E_G. A path cover (in short: cover) of G is a set of vertex-disjoint paths which cover all the vertices of G. An optimal cover of G is a cover of the smallest possible cardinality. The cardinality of such a cover is called the covering number of G, and is denoted by \pi(G).

The concept of graph covering has many practical applications. For example, in order to establish ring protocols [10], a computer network may be augmented by some auxiliary edges so as to make it Hamiltonian [5]. It is easily verified that the minimum number of additional edges needed to make a network Hamiltonian is identical to the covering number of the network. Other notable applications of graph covering are code optimization [3] and mapping parallel programs to parallel architectures [9].
The problem of finding an optimal cover is NP-complete, even for cubic 3-connected planar graphs [6]. There are, however, several results on optimal covering of restricted classes of graphs. Boesch, Chen and McHugh have derived in [2], among other things, an optimal covering algorithm for trees. Their result was generalized by Pinter and Wolfstahl [9], who developed an efficient optimal covering algorithm for graphs where no two cycles share a vertex. Boesch and Gimpel [3] have considered the related problem of covering a directed acyclic graph by directed paths.

The main result presented in this paper generalizes the above results of [2, 9]. Specifically, we develop a linear time covering algorithm for cacti, that is, graphs where no edge lies on more than one cycle [1, 8, 11] (see Fig. 1). We note that the class of cacti properly contains the graph classes considered in [2, 9]. The algorithm basically operates by applying two types of rules, namely, edge-deletion rules and a recursive decomposition rule. The edge-deletion rules characterize the edges that can be deleted from a given cactus without affecting its covering number. One such rule is used to bring a cactus to a state where each end-cycle (a concept to be defined) is of a certain type. The other edge-deletion rules are used to delete edges on these end-cycles, an operation we call opening end-cycles. The recursive decomposition rule provides a tool for constructing an optimal cover of a cactus by decomposing it into two components and covering each component separately.

![Fig. 1. A cactus (reproduced from [1]).](image)

The rest of this paper is organized as follows. Section 2 discusses the difference between covering trees and cacti. The edge-deletion rules are presented in Section 3. The recursive decomposition rule is presented in Section 4. The algorithm, developed in Section 5, specifies the order by which those rules are to be applied.
2. The difference between covering cacti and trees

In order to develop intuition to the optimal covering problem, as well as to motivate our proposed algorithmic approach, we next compare the cactus covering problem to the tree covering problem. In this context, it is worth noting that not every spanning tree $T$ of a cactus $G$ has the property $\pi(T) = \pi(G)$. Consider, for example, the cactus $G$ of Fig. 2. By deleting the edges $\{(2, 3), (5, 6)\}$ one obtains a tree with covering number 2, which is also the covering number of $G$. On the other hand, by deleting the edges $\{(1, 3), (4, 6)\}$ one obtains a tree with covering number 3.

![Fig. 2. The cactus G.](image)

The optimal tree covering algorithm of [Z] is based on a single edge-deletion rule. This rule is repeatedly applied to a forest $F$ which initially consists of the input tree $T$. The algorithm terminates when $F$ has reduced to a forest $\tilde{F}$, where each tree is a path. The above edge-deletion rule preserves the covering number of $T$, so $|\tilde{F}| = \pi(T)$. The rule is applicable to any vertex $v$ of $F$ such that $v$ has two neighbors $x_1$ and $y_1$, and there are two vertex-disjoint paths $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_m)$, where the degree of each vertex on these paths is 2 except for $x_n$ and $y_m$, whose degrees are 1. Specifically, the rule states that by deleting all the edges incident to $v$, except for $(v, x_1)$ and $(v, y_1)$, one obtains a forest whose covering number is $\pi(T)$. Figure 3 illustrates this rule.

Note that the above edge-deletion rule is actually a local rule, in the sense that it can be applied to any vertex satisfying the above requirement, regardless of the structure of the rest of the forest. One may be tempted to think that a similar approach can be used for optimally covering cacti by, say, using local edge-deletion rules to open the cycles. However, this is not the case, since in an optimal cover, the covering of a cycle may depend on the covering of the rest of the cactus. As an example, consider the cactus $G_1$ of Fig. 4(a). A seemingly reasonable edge-deletion approach for covering the vertices of the cycle $C$ is to delete the edges $(1, 2)$ and $(2, 3)$. In fact, if the rest of $G_1$ is as in Fig. 4(b), then these edges are not used by the optimal cover shown in the figure, and can indeed be deleted. However, if the rest of $G_1$ is as in Fig. 4(c), then these edges must be used by every optimal cover of $G_1$. We thus conclude that local edge-deletion operations do not suffice for covering cacti.
Fig. 3. The edge-deletion rules for trees: (a) before applying the edge-deletion rule to vertex $v$, (b) after the deletion.
3. Edge-deletion rules

The edge-deletion rules are presented in the lemmas below. First, we need some definitions.

**Definition 3.1.** Let $S_G$ be a cover of a graph $G = (V_G, E_G)$. We say that $S_G$ employs an edge $e \in E_G$ if some path in $S_G$ includes $e$. The degree of a vertex $v \in V_G$ is denoted by $\text{deg}_G(v)$. Whenever the relevant graph is clear in the context, the subscript specifying the graph is omitted. A trail starting at $v_1 \in V_G$ is a path $(v_1, v_2, \ldots, v_k)$ containing two or more vertices, where $\text{deg}(v_i) = 2$ for $1 < i < k$ and $\text{deg}(v_k) = 1$. A vertex $v_1 \in V_G$ is a fork, if $\text{deg}(v_1) \geq 3$, at least one trail $(v_1, v_2, \ldots)$ starts at $v_1$, and $v_1$ is adjacent to a vertex $w \neq v_2$ of degree 1 or 2. A trimmed cactus is a cactus containing no forks.

The following proposition is often used in the sequel.

**Proposition 3.2.** Let $S$ be a cover of a graph $G = (V_G, E_G)$. Let $u, v$ and $w$ be vertices in $V_G$ where

1. $\{(u, v), (u, w)\} \subseteq E_G$,
2. $(u, w)$ is employed by $S$ but $(u, v)$ is not, and
3. $v$ is an end-vertex of some path in $S$.

Then there exists a cover of $G$, denoted by $\tilde{S}_G$, that employs $(u, v)$ but not $(u, w)$ and satisfies $|\tilde{S}_G| = |S_G|$.
Proof. Let $\overline{S}_G = (S_G - \{(u, w)\}) \cup \{(u, v)\}$. □

Lemma 3.3 below is easily proved using the above proposition.

**Lemma 3.3** (Pinter and Wolfstahl) [9] (Deletions due to forks). Let $G = (V_G, E_G)$ be a cactus. Let $v_1 \in V_G$ be a vertex of degree 3 or more which is the start-point of a trail $(v_1, v_2, \ldots, v_k)$ and is adjacent to a vertex $w \neq v_2$ of degree 1 or 2. Then $G' = (V_G, E'_G)$ where $E'_G = E_G - \{(x, v_1)\} \cup \{(x, v_1)\} \cup \{(x, v_i)\} \in E_{G}, x \notin \{v_2, w\}$ satisfies $\pi(G) = \pi(G')$.

**Definition 3.4.** A connected graph $G = (V_G, E_G)$ is said to have a separation vertex $v \in V_G$ if there exist vertices $a, b \in V_G$, $a \neq v, b \neq v$, such that all the paths connecting $a$ and $b$ contain $v$. A graph that has a separation vertex is called separable and one that has none is called nonseparable. Let $V' \subseteq V_G$. The subgraph induced by $V'$ is called a nonseparable component of $G$ if it is nonseparable and if for any larger $V''$, $V'' \subseteq V_G$, the subgraph induced by $V''$ is separable. Let $N(G)$ be the set of the nonseparable components of $G$. Observe that if $G$ is a cactus, then $N(G) = \Phi(G) \cup \mathcal{E}(G)$ where $\Phi(G)$ is a set of cycles and $\mathcal{E}(G)$ is a set of edges not on cycles. Let $S(G)$ be the set of the separating vertices in $G$. The superstructure of $G$ is the tree $T = (V_T, E_T)$, where $V_T = V_N \cup V_S$, $V_N = \{v_n | n_i \in N(G)\}$, $V_S = \{v_s | s_i \in S(G)\}$ and $E_T = \{(v_s, v_n) | s_i \in S(G), n_i \in N(G), s_i \text{ is a vertex of } n_i\}$. Figure 5 depicts a superstructure of a cactus $G$. Suppose that a vertex $r \in V_T$ is chosen as the root of $T$. Then, for any other vertex $v \in V_T$, let $f(v)$ denote the neighbor of $v$ which is on the unique path between $r$ and $v$. If $u = f(v)$ then $v$ is called the son of $u$. The transitive closure of the son relation is the descendant relation. By $d_T(x, y)$ we denote the distance in $T$ between two vertices $x, y \in V_T$.

**Definition 3.5.** A crown is a trimmed cactus containing a single cycle. Given a crown $C$, the cycle in $C$ is denoted by $C^c$.

Next, we define the concept of end-cycle which plays a key role in the development of our covering algorithm. In fact, the rest of the edge-deletion rules, as well as the recursive decomposition rule, are all applicable to end-cycles.

**Definition 3.6.** Let $G$ be a trimmed cactus containing cycles, and let $T = (V_T, E_T)$ be the superstructure of $G$. Assume that $G$ is not an isolated cycle. Choose an arbitrary $r \in V_S$ to be the root of $T$. Let $v_n \in V_N$ be a vertex of $T$ that corresponds to a cycle $n_i \in \Phi(G)$ and satisfies $d_T(r, v_n) = \max_{v_{n_i} \in \Phi(G)}(d_T(r, v_{n_i}))$. That is, $v_n$ is a vertex whose distance from $r$ is maximal, among all vertices corresponding to cycles in $G$. Note that $f(v_n) = v_{n_i}$ for some separating vertex $s_i$ in $G$. Let $T'$ denote the subgraph of $T$ that is induced by $v_n, v_{n_i}$ and the descendants of $v_{n_i}$ in $T$. Let $C$ be the subgraph of $G$ corresponding to $T'$. Then, by the choice of $v_n$, $C$ is a crown where $C^c = n_i$. This crown is said to be an end-cycle of $G$, denoted $C = G$. The vertex $v_n = f(v_{n_i})$ is called the anchor of $C$. Isolated cycles are also defined to be end-cycles, and the anchor of such an end-cycle $C$ is an arbitrary vertex on $C$. For example,
Optimal covering of cacti by vertex-disjoint paths

Fig. 5. A superstructure: (a) $G$, (b) $T$. 
the end-cycles in Fig. 6 are $C_1$, $C_2$ and $C_3$. Note that in an end-cycle $C$, all vertices except the anchor have the same degree $d \in \{2, 3\}$. An end-cycle $C \subseteq G$ is an end-cycle of order $d$, $d \in \{2, 3\}$, if each vertex $v$ on $C^\circ$, except perhaps for the anchor, satisfies $\deg(v) = d$. For example, crown $C_1$ in Fig. 6 is an end-cycle of order 2 while crown $C_2$ is an end-cycle of order 3.

Our optimal covering algorithm, a sketch of which is given below, basically operates by applying local edge-deletion rules (steps 1-2) and a recursive decomposition rule (step 3). The edge-deletion rules are first used to trim $G$ (step 1), and then to delete edges on certain end-cycles (step 2). The latter operation is called opening end-cycles, and may result in a nontrimmed cactus. Steps 1 and 2 are repeated until no edge-deletion rule is applicable. Then, the cactus is decomposed (step 3), and the algorithm is recursively applied to the decomposed cactus. The deletions/decompositions are done such that the covering number of the cactus is preserved. Eventually, the cactus reduces to a set of paths which constitutes an optimal cover of $G$. A more detailed outline of the algorithm is the following.

1. Trim $G$, adding the created paths to the cover $S_G$;
2. If there is an end-cycle $C \subseteq G$ where
   - $C$ is of order 2 or an isolated cycle, or
   - $C$ is of order 3 where $\deg(\text{anchor}) = 3$, or
   - $C$ is of order 3 where a trail starts at the anchor, or
   - $C$ is of order 3 where the number of trails starting on $C^\circ$ is odd, or
   - $C$ is of order 3 where the number of trails starting on $C^\circ$ is even, and the anchor is shared by another such end-cycle,
   then do open $C$; add the created paths to $S_G$; go to step 1 od

Fig. 6. End-cycles.
(3) If an end-cycle $C \preceq G$ exists \{C must be of order 3, the number of trails starting on $C^o$ is even, and $\deg(\text{anchor}) = 4\}

then do construct the graph $G[C]$; \{$G[C$ is yet to be defined$];
recursively apply the algorithm to $G[C$, thus finding a cover $S_{G[C]}$;
cover $G$, using information obtained from $S_{G[C]}$;

od

Having outlined the algorithm, we next show how the different end-cycles can be opened in step 3 while preserving the covering number of the cactus. The recursive decomposition rule will be developed in Section 3.

The following lemma is proved using Proposition 3.2.

Lemma 3.7 (Pinter and Wolfstahl [9]) (Opening end-cycles of order 2 or isolated cycles). Let $G = (V_G, E_G)$ be a cactus. Let $C$ be a subgraph of $G$ which is either an isolated cycle or an end-cycle of order 2. Let $v_1, v_2, \ldots, v_k$ be the vertices on $C^o$, starting from the anchor. Then $G' = (V_G, E'_G)$ where $E'_G = E_G - \{(v_1, v_2)\}$ satisfies $\pi(G') = \pi(G)$.

Suppose that neither Lemma 3.3 nor Lemma 3.7 is applicable to a cactus $G$. Then each end-cycle in $G$ is of order 3. The following lemmas are concerned with such end-cycles.

Definition 3.8. Let $G = (V_G, E_G)$ be a cactus. If exactly one trail starts at a vertex $v \in V_G$ then this trail is denoted by $t(v)$. Let $t(v) = (v, v_1, \ldots, v_k)$ be the single trail starting at a vertex $v \in V_G$. Then $t^{-1}(v)$ is the path $(v_k, \ldots, v_1, v)$. Let $p_1 = (v_1 \ldots v_k)$ and $p_2 = (u_1 \ldots u_l)$ be two paths in $G$, where $(v_k, u_1) \in E_G$. Then $p_1p_2$ is the path $(v_1 \ldots v_k, u_1 \ldots u_l)$.

The following useful lemma is thoroughly used in the sequel.

Lemma 3.9 (Switching edges in a cover of an end-cycle of order 3). Let $G = (V_G, E_G)$ be a cactus. Let $C \preceq G$ be an end-cycle of order 3 where $v_1, v_2, v_3, \ldots, v_k$ are the vertices on $C^o$, starting from the anchor. Let $S_G$ be a cover of $G$ which employs $(v_1, v_2)$ but not $(v_1, v_k)$. Then there exists an equal size cover of $G$, denoted by $\tilde{S}_G$, which employs $(v_1, v_k)$ but not $(v_1, v_2)$.

Proof. Let $p = p'p''$ be the path in $S_G$ which employs $e$, where $p'$ may be empty (i.e. contain no vertices) and $p'' = (v_1, v_2, \ldots)$. $\tilde{S}_G$ is defined as follows. All paths in $S_G$ that do not cover vertices in $C$ are also in $\tilde{S}_G$. Observe that since $C$ contains $k-1$ vertices of degree 1 and $p$ contains at most one of them, at least $\lceil k/2 \rceil$ additional paths are used by $S_G$ to cover the vertices in $C$ and in $p_1$. To prove the lemma, it suffices to show that $\tilde{S}_G$ uses exactly $\lceil k/2 \rceil$ paths to cover those vertices, employing $(v_1, v_k)$ but not $(v_1, v_2)$. This is established by having $\tilde{S}_G$ cover those vertices using
the paths $p_1, p_2, \ldots, p_{\lfloor k/2 \rfloor}$ as follows (see Fig. 7, where the bold edges are employed by $S_G$):

1. $p_1 = p'_G(v_1) \cdot \text{tr}(v_k)$.
2. For $1 < i < \lfloor k/2 \rfloor$, $p_i = \text{tr}^{-1}(v_{k-2i+3}) \cdot \text{tr}(v_{k-2i+2})$.
3. $p_{\lfloor k/2 \rfloor} = \text{tr}^{-1}(v_3) \cdot \text{tr}(v_2)$ if $k$ is even, and $p_{\lfloor k/2 \rfloor} = \text{tr}(v_2)$ if $k$ is odd. □

**Definition 3.10.** Let $G = (V_G, E_G)$ be a cactus. A bridged end-cycle of $G$ is an end-cycle of order 3 where the anchor is of degree three.

**Lemma 3.11** (Opening bridged end-cycles). Let $G = (V_G, E_G)$ be a cactus. Let $C \in G$ be a bridged end-cycle, where $v_1, v_2, \ldots, v_k$ are the vertices on $C^*$, starting from the anchor. Then $G' = (V_G, E_G')$ where $E'_G = E_G - \{(v_2, v_3)\}$ satisfies $\pi(G') = \pi(G)$.

**Proof.** Clearly, $\pi(G) \leq \pi(G')$. To prove the reverse inequality, we show that every optimal cover of $G$ defines an equal-size cover of $G'$. Let $S_G$ be an optimal cover of $G$. If $(v_2, v_3)$ is not employed by $S_G$ then we are done, since $S_G$ is also a cover of $G'$, so assume that $(v_2, v_3)$ is employed by $S_G$.

1. Suppose that $(v_1, v_2)$ is employed by $S_G$. Let $u$ be the second vertex on $\text{tr}(v_2)$. Since $(v_2, v_3)$ is employed by $S_G$, $(u, v_2)$ is not employed by $S_G$. By modifying $S_G$ to employ $(u, v_2)$ rather than $(v_2, v_3)$ one obtains, using Proposition 3.2, an equal-size cover of $G$ where $(v_2, v_3)$ is not employed. This cover is also a cover of $G'$.

2. Suppose that $(v_1, v_2)$ is not employed by $S_G$ but $(v_1, v_k)$ is. Then by Lemma 3.9, there exists an equal-size cover of $G$ where $(v_1, v_k)$ is not employed but $(v_1, v_2)$ is, and the argument of (1) above applies.

3. Suppose that neither $(v_1, v_2)$ nor $(v_1, v_k)$ is employed by $S_G$. In this case, $v_1$ is the end-vertex of some path $p \in S_G$ and $(u, v_2)$ is employed by $S_G$. By modifying $S_G$ to employ $(v_1, v_2)$ rather than $(v_2, v_3)$ one obtains, using Proposition 3.2, an equal-size cover of $G$ where $(v_2, v_3)$ is not employed. This cover is also a cover of $G'$. □
Definition 3.12. Let $G = (V_G, E_G)$ be a cactus. An anchor-trailed end-cycle of $G$ is an end-cycle of order 3 where the anchor is the start-point of a trail.

Lemma 3.13 (Opening anchor-trailed end-cycles). Let $G = (V_G, E_G)$ be a cactus. Let $C \subseteq G$ be an anchor-trailed end-cycle, where $v_1, v_2, \ldots, v_k$ are the vertices on $C^o$, starting from the anchor. Then $G' = (V_G, E'_G)$ where $E'_G = E_G - \{(v_1, v_2)\}$ satisfies $\pi(G') = \pi(G)$.

Proof. Clearly, $\pi(G) \leq \pi(G')$. To prove the reverse inequality (hence equality), we show that every optimal cover of $G$ defines an equal-size cover of $G'$. Let $S_G$ be an optimal cover of $G$.

1) If $(v_1, v_2)$ is not employed by $S_G$ then we are done, since $S_G$ is also a cover of $G'$.

2) If $(v_1, v_2)$ is employed by $S_G$ but $(v_1, v_k)$ is not, then by Lemma 3.9 there exists an equal-size cover of $G$ where $(v_1, v_2)$ is not employed but $(v_1, v_k)$ is. This cover is also a cover of $G'$.

3) Suppose that $(v_1, v_2)$ and $(v_1, v_k)$ are both employed by $S_G$. Let $u$ be the second vertex on a trail starting at $v_1$; observe that $(u, v_1)$ is not employed by $S_G$. By modifying $S_G$ to employ $(u, v_1)$ rather than $(v_1, v_2)$, one obtains, using Proposition 3.2, an equal-size cover of $G$ where $(v_1, v_2)$ is not employed. This cover is also a cover of $G'$.

Definition 3.14. Let $G = (V_G, E_G)$ be a cactus. An odd-trailed end-cycle of $G$ is an end-cycle $C$ of order 3, where $C$ is not anchor-trailed and the number of trails starting on $C^o$ is odd. An even-trailed end-cycle of $G$ is an end-cycle $C$ of order 3, where $C$ is not anchor-trailed and the number of trails starting on $C^o$ is even. Note that an odd-trailed (even-trailed) end-cycle $C$ has an even (odd) number of vertices on $C^o$.

Lemma 3.15 (Opening odd-trailed end-cycles). Let $G = (V_G, E_G)$ be a cactus. Let $C \subseteq G$ be an odd-trailed end-cycle where $v_1, v_2, \ldots, v_k$ are the vertices on $C^o$, starting from the anchor ($k$ is even). Then $G' = (V_G, E'_G)$ where $E'_G = E_G - \{(v_1, v_2)\}$ satisfies $\pi(G') = \pi(G)$.

Proof. Clearly $\pi(G) \leq \pi(G')$. To prove the reverse inequality, we show that every optimal cover of $G$ defines an equal-size cover of $G'$. Let $S_G$ be an optimal cover of $G$.

1) If $(v_1, v_2)$ is not employed by a path in $S_G$ then we are done, since $S_G$ is also a cover of $G'$.

2) If $(v_1, v_2)$ is employed by $S_G$ but $(v_1, v_k)$ is not, then by Lemma 3.9 there exists an equal-size cover of $G$ where $(v_1, v_2)$ is not employed but $(v_1, v_k)$ is. This latter cover is also a cover of $G'$.
(3) Suppose that \((v_1, v_2)\) and \((v_1, v_k)\) are both employed by \(S_G\). Clearly, every path of \(S_G\) either covers only vertices in \(C\) or only vertices not in \(C\). In this case, an equal-size cover of \(G\), denoted by \(\tilde{S}_G\), can be constructed as follows: All paths in \(S_G\) that do not cover vertices in \(C\) are also in \(\tilde{S}_G\). Observe that at least \(k/2\) additional paths are used by \(S_G\) to cover the vertices in \(C\). To prove that \(\tilde{S}_G\) is an optimal cover of \(G^\prime\), it suffices to show that it uses exactly \(k/2\) paths to cover those vertices, without employing \((v_1, v_2)\). This is established by having \(S_G\) cover those vertices using the paths \(p_1, p_2, \ldots, p_{k/2}\), where \(p_i = v_1 \cdot \text{tr}(v_k)\) and for \(1 < i \leq k/2\), 
\[p_i = \text{tr}^{-1}(v_{k-2i+2}) \cdot \text{tr}(v_{k-2i+3}). \]

**Definition 3.16.** Let \(G\) be a cactus and let \(C = (V_C, E_C)\) be an even-trailed end-cycle of \(G\), where \(v_1, v_2, \ldots, v_k\) are the vertices on \(C\), starting from the anchor. Then \(C^\prime\) is defined to be the graph induced on \(V_C - \{v_1\}\). Define \(\Delta(C)\) to be a cover of \(C\) using \((k - 1)/2\) paths as follows:

(1) \(p_1 = \text{tr}^{-1}(v_2) \cdot \text{tr}(v_k)\).

(2) If \(k > 3\), then for \(1 < i \leq (k-1)/2\), \(p_i = \text{tr}^{-1}(v_{i-1}) \cdot \text{tr}(v_i)\).

Define \(\Lambda(C^\prime)\) to be a cover of \(C^\prime\) using \((k - 1)/2\) paths as follows: for \(1 \leq i \leq (k-1)/2\), 
\[p_i = \text{tr}^{-1}(v_{i-2}) \cdot \text{tr}(v_{i+1}). \]

**Lemma 3.17 (Covering even-trailed end-cycles using \(\Lambda\) and \(\Delta\)).** Let \(G = (V_G, E_G)\) be a cactus. Let \(C = (V_C, E_C)\) be an even-trailed end-cycle where \(v_1, v_2, \ldots, v_k\) are the vertices on \(C\), starting from the anchor. Let \(S_G\) be an optimal cover of \(G\).

(1) If \(S_G\) employs neither \((v_1, v_2)\) nor \((v_1, v_k)\), then \(S_G\) uses \(\Lambda(C^\prime)\) to cover \(C^\prime\).

(2) If \(S_G\) employs both \((v_1, v_2)\) and \((v_1, v_k)\), then \(S_G\) uses \(\Delta(C)\) to cover \(C\).

**Proof.** (1) Let \(S\) be the set of paths used by \(S_G\) to cover \(C^\prime\). If \(S \neq \Lambda(C^\prime)\), then there exists a path \(p = (u_1, \ldots, u_k) \in S\) where at least one vertex from \(\{u_1, u_k\}\) is not the end-vertex of a trail. Since there are \(k-1\) trails starting on \(C^\prime\), \(S - \{p\}\) must cover at least \(k-2\) vertices of degree one, using at least \([((k-2)/2)] = (k-1)/2\) paths. Thus,
\[
|S| = 1 + |S - \{p\}| = 1 + \frac{k-1}{2} = \frac{k+1}{2}.
\]

Since \(|\Lambda(C^\prime)| = (k-1)/2\), a contradiction to the optimality of \(S_G\) arises.

(2) The proof for this case is similar to that former case, and is omitted. \(\square\)

**Lemma 3.18 (Opening even-trailed end-cycles that share a vertex).** Let \(G = (V_G, E_G)\) be a cactus. Let \(X = \{C_1, C_2, \ldots, C_n\}\) \((n > 1)\) be a set of even-trailed end-cycles in \(G\), all sharing the anchor \(v_1\). Let \(v_1, v_2, \ldots, v_k\) be the vertices on \(C_1\), starting from the anchor. Then \(G' = (V_G, E_G)\) where \(E_G = E_G - \{(v_1, v_2)\}\) satisfies \(\pi(G') = \pi(G)\).

**Proof.** Clearly \(\pi(G) \leq \pi(G')\). To prove the reverse inequality, we show that every optimal cover of \(G\) defines an equal-size cover of \(G'\). Let \(S_G\) be an optimal cover of \(G\).
(1) If \((v_1, v_2)\) is not employed by a path in \(S_G\) then we are done, since \(S_G\) is also a cover of \(G'\).

(2) If \((v_1, v_2)\) is employed by \(S_G\) but \((v_1, v_k)\) is not, then by Lemma 3.9 there exists an equal-size cover of \(G\) that employs \((v_1, v_k)\) but not \((v_1, v_2)\). This latter cover is also a cover of \(G'\).

(3) Suppose that \((v_1, v_2)\) and \((v_1, v_k)\) are both employed by \(S_G\). Then in some other even-trailed end-cycle \(C_i \in X\), neither of the edges incident to \(v_1\) is employed. By Lemma 3.17, \(S_G\) covers \(C\) and \(C_i\) using \(\Delta(C)\) and \(\Delta(C_i)\), respectively. Observe that an equal-size cover of \(G\) is given by

\[
((S_G - \Delta(C)) - \Delta(C_i)) \cup \Delta(C_i) \cup \Delta(C).
\]

This latter cover does not employ \((u_1, v_2)\), and is thus a cover of \(G'\). □

4. A recursive decomposition rule

In this section we consider end-cycles to which neither of the above edge-deletion rules is applicable. Let \(G = (V_G, E_G)\) be a trimmed cactus that properly contains a cycle. Then \(G\) contains an end-cycle. Moreover, all end-cycles in \(G\) are either of order 2 or of order 3. Assume that no end-cycle of \(G\) is anchor-trailed, bridged, odd-trailed or an even-trailed that shares its anchor with other even-trailed end-cycles. Then any end-cycle \(C \in G\) must be even-trailed where the degree of the anchor is 4. The recursive decomposition rule applies to such end-cycles.

**Definition 4.1.** Let \(G = (V_G, E_G)\) be a cactus. A final even-trailed end-cycle of \(G\) is an even-trailed end-cycle where the anchor is of degree 4. Let \(C = (V_C, E_C)\) be a final even-trailed end-cycle in \(G\), where \(v_1, v_2, \ldots, v_k\) are the vertices on \(C\), starting from the anchor. Let \(u\) and \(w\) be the vertices adjacent to \(v_1\) that are not on \(C^\circ\). Then \(G|C\) is defined to be the graph \(G|C = (V_G - V_C, E_G)\) where \(E_G' = (E_G - E_C) \cup \{u, w\}\) (see Fig. 8).

**Lemma 4.2** (Recursive decomposition rule for final even-trailed end-cycles, part I). Let \(G = (V_G, E_G)\) be a cactus. Let \(C \in G\) be a final even-trailed end-cycle where \(v_1, v_2, \ldots, v_k\) are the vertices on \(C^\circ\) starting from the anchor. Then

\[
\pi(G|C) \leq \pi(G) - \frac{k - 1}{2}.
\]

**Proof.** Let \(S_G\) be an optimal cover of \(G\). It is proved that \(S_G\) defines a cover \(S_{G|C}\) of \(G|C\) such that \(|S_{G|C}| = |S_G| - (k - 1)/2\). In the sequel, let \(u\) and \(w\) be the vertices adjacent to \(v_1\) that are not on \(C^\circ\).

(1) Suppose that neither \((u, v_1)\) nor \((v_1, w)\) is employed by \(S_G\). In this case, it is easily verified that \(S_G\) uses \(\Delta(C)\) to cover \(C\). Observe that for each edge \(e \in E_G\), if
Fig. 8. Definition of $G|C$: (a) $G$, (b) $G|C$.

$e$ is employed by $S_G - \Delta(C)$ then $e$ is an edge in $G|C$. Thus, $S_{G|C} = S_G - \Delta(C)$ is a cover of $G|C$. Its size is given by $|S_G| - |\Delta(C)| = |S_G| - (k - 1)/2$.

(2) Suppose that exactly one edge from $\{(u, v_1), (v_1, w)\}$, say $e = (u, v_1)$, is employed by $S_G$. Let $p = p_1*(u, v_1)*p_2$ be the path in $S_G$ which employs $e$, where $p_1$ and $p_2$ may be empty. Using Proposition 3.2 and the fact that $C$ is even-trailed, the reader can verify that an equal-size cover of $G$ which employs neither $(u, v_1)$ nor $(v_1, w)$ but both $(v_1, v_2)$ and $(u, v_k)$ can be obtained by replacing $p$ by $p_1*(u)$ and using $\Delta(C)$ to cover $C$. From here, the argument of (1) above applies.

(3) Suppose that both $(u, v_1)$ and $(v_1, w)$ are employed by $S_G$. Then neither $(v_1, v_2)$ nor $(v_1, v_k)$ is employed by $S_G$. By Lemma 3.17, $S_G$ uses $\Delta(C^v)$ to cover $C$. Let $p \in S_G$ be the path employing $(u, v_1)$ and $(v_1, w)$, and let $p'$ be the path obtained by deleting $v_1$ from $p$. Then

$$S_{G|C} = ((S_G - \Delta(C^v)) - \{p\}) \cup \{p'\}$$

is a cover of $G|C$, and its size is $|S_G| - |\Delta(C^v)| = |S_G| - (k - 1)/2$. □

**Lemma 4.3** (Recursive decomposition rule for final even-trailed end-cycles, part II). Let $G = (V_G, E_G)$ be a cactus. Let $C \subseteq G$ be a final even-trailed end-cycle where $v_1, v_2, \ldots, v_k$ are the vertices on $C^v$, starting from the anchor. Let $u$ and $w$ be the vertices adjacent to $v_1$ that are not on $C^v$, and let $S_{G|C}$ be an optimal cover of $G|C$. 
(1) Suppose that some path \( p \in S_{G|C} \) employs \((u, w)\). Let \( \tilde{p} \) be the path obtained from \( p \) by inserting \( v \) between \( u \) and \( w \). Then \( S_G = (S_{G|C} - \{ p \}) \cup \tilde{p} \cup A(C^v) \) is an optimal cover of \( G \).

(2) If \( S_{G|C} \) does not employ \((u, w)\), then \( S_G = S_{G|C} \cup \Delta(C) \) is an optimal cover of \( G \).

**Proof.** In both cases, \( |S| = |S_{G|C}| - (k - 1)/2 \). Combining this fact with Lemma 4.2, we conclude that \( S_G \) is optimal. \( \square \)

5. The algorithm

In this section we present a first version of our algorithm for optimal covering of cacti. This version is called Algorithm A. In developing Algorithm A, we focus on simplicity rather than efficiency. An efficient (and more complicated) algorithm is described later.

We next review some definitions that were given in the previous sections, to be used by the algorithm. Let \( G = (V_G, E_G) \) be a cactus. A vertex \( v_1 \in V_G \) is a **fork** if \( \deg(v_1) \geq 3 \), exactly one trail \((v_1, v_2, \ldots)\) starts at \( v_1 \), and \( v_1 \) is adjacent to a vertex \( w \neq v_2 \) of degree 1 or 2. An **anchor-trailed end-cycle** is an end-cycle of order 3 where the anchor is the start-point of a trail. A **bridged end-cycle** is an end-cycle of order 3 where the anchor is of degree 3. An **odd-trailed end-cycle** is an end-cycle \( C \) of order 3, where the number of trails starting on \( C^o \) is odd. An **even-trailed end-cycle** is an end-cycle \( C \) of order 3, where the number of trails starting on \( C^e \) is even. A **final even-trailed end-cycle** is an even-trailed end-cycle where the anchor is of degree 4.

We are now able to present Algorithm A.

**Algorithm A**

**Input:** A cactus \( G = (V_G, E_G) \).

**Output:** A set of paths \( S_G \), comprising an optimal cover of \( G \).

**Procedure used:**

**Procedure** Transfer-Paths;

**do**

Add the isolated paths in \( G = (V_G, E_G) \) to \( S_G \).

\( V_G \leftarrow V_G - \{ v \in V_G \mid \text{some path in } S_G \text{ covers } v \} \).

\( E_G \leftarrow E_G - \{ e \in E_G \mid \text{some path in } S_G \text{ employs } e \} \).

**od**

**Method:**

(1) **Initialize** \( S_G \leftarrow \emptyset \).

(2) **Transfer-Paths.**

(3) **While** \( G = (V_G, E_G) \) contains forks,

(3.1) **Choose** a fork \( v \).

(3.2) **Apply** Lemma 3.3 to \( v \).

(3.3) **Transfer-Paths.**
(* Comment: At this point, G is a union of trimmed cacti, and the end-cycles in G are of order 2 or 3. *)

(4) If G = (V, E_G) contains a subgraph C which is either an isolated cycle or an end-cycle of order 2, then
   (4.1) Apply Lemma 3.7 to C.
   (4.2) Transfer-Paths.
   (4.3) Go to step 3.
   (* Comment: At this point, all end-cycles in G are of order 3. *)

(5) If G = (V, E_G) contains a bridged end-cycle C, then
   (5.1) Apply Lemma 3.11 to C.
   (5.2) Go to step 3.

(6) If G = (V, E_G) contains an anchor-trailed end-cycle C, then
   (6.1) Apply Lemma 3.13 to C.
   (6.2) Go to step 3.

(7) If G = (V, E_G) contains an odd-trailed end-cycle C, then
   (7.1) Apply Lemma 3.15 to C.
   (7.2) Go to step 3.
   (* Comment: At this point, all end-cycles in G are even-trailed end-cycles. *)

(8) If G = (V, E_G) contains two or more even-trailed end-cycles that share a vertex v, then
   (8.1) Choose an even-trailed end-cycle C from these sharing v.
   (8.2) Apply Lemma 3.18 to C.
   (8.3) Go to step 3.
   (* Comment: At this point, all end-cycles in G are final even-trailed end-cycles. *)

(9) If G = (V, E_G) contains a final even-trailed end-cycle C, then
   (9.1) Let v be the anchor of C. Let u and w be the vertices adjacent to v that are not on C^v.
   (9.2) Recursively apply the algorithm to G|C, resulting in an optimal cover S_{G|C}.
   (9.3) If (u, w) is employed by a path p ∈ S_{G|C}, then
         (9.3.1) Let p' the path obtained from p by inserting v between u and w.
         (9.3.2) S_G ↔ S_G ∪ ((S_{G|C} - {p}) ∪ p' ∪ Δ(C^v))
   (9.4) If (u, w) is not employed by S_{G|C}, then S_G ↔ S_G ∪ S_{G|C} ∪ Δ(C).

(10) Stop.

Theorem 5.1 (Correctness of Algorithm A). Given a cactus G = (V_G, E_G), Algorithm A produces an optimal path cover of G.

Proof. Whenever the algorithm returns to step 3, the size of E_G is strictly smaller than it was in the previous execution of step 3. Thus, the algorithm eventually terminates, since none of the conditions tested in steps 3–10 holds when E_G = ∅.
Upon termination, $G$ contains no forks, isolated cycles, or end-cycles. Hence, $G$ contains no nonisolated cycles. Also, $G$ contains no isolated paths upon termination, for such paths, which are generated only by applying Lemmas 3.3 and 3.7, are immediately transferred to $G$. It follows that upon termination $V_G = \emptyset$, so $S_G$ is a cover of $G$.

The algorithm deletes edges from $G$ only by applying the edge-deletion rules. The decomposition rule ensures that the construction of an optimal cover, upon return from each recursive invocation of the algorithm, is properly done. We conclude that when the algorithm terminates, $|S_G| = \pi(G)$. 

Using the fact that the number of cycles in a cactus $G = (V_G, E_G)$ is $O(|E_G|)$, the reader can verify that Algorithm A can be implemented in $O(|E_G|^2) = O(|V_G|^2)$ time. However, a better bound is in fact achievable by Algorithm B below. Algorithm B is based on the DFS (depth first search) algorithm ([7], see also [4]), with which we assume the reader is familiar.

**Definition 5.2.** An EDFS is a DFS extended to identify forks upon backtracking from such. Recall that DFS generates a directed tree, where edges not in the tree are called back-edges. Assume that an EDFS is applied to a cactus $G = (V_G, E_G)$, and that $e \in E_G$ is a back-edge of the EDFS tree. Then $C(e)$ is defined to be the unique cycle in $G$ which contains $e$. The source of a cycle $C$ with respect to the EDFS is the first vertex on $C$ which was discovered by the EDFS. Note that if $v$ is a source of a cycle, $C$, then there is a back-edge $(u \rightarrow v)$ on $C$ entering $v$. Let $G = (V_G, E_G)$ be a graph, and let $v \in V_G$ be a separation vertex of $G$. We say that $v$ separates a connected subgraph $A = (V_A, E_A)$ from $G$ if $v \in V_A$, and for all $u \in V_G - V_A$ and $w \in V_A$, all the paths connecting $u$ and $w$ pass through $v$. Assume that $v$ separates a tree $T$ from $G$. An elimination of $T$ from $G$ is an EDFS traversal of $T$, starting from $v$, where Lemma 3.3 is applied to each fork $u \in V_G - \{v\}$ upon backtracking from $u$ (if $T$ is a trail, then the elimination has no effect).

Algorithm B, whose time complexity is $O(|V_G|)$, is outlined below. It is based on an EDFS traversal of the input cactus. In the course of the EDFS, the edge-deletion rules, as well as the recursive decomposition rule, are applied to $G$, resulting in a properly smaller graph. Specifically, these rules are applied whenever the algorithm backtracks from a fork that is not on a cycle, or from a source of a cycle. The isolated paths created by applying the edge-deletion rules are transferred to set $S_G$, which eventually constitutes an optimal cover of $G$. Whenever a final even-trailed end-cycle $C$ is detected, the two vertices adjacent to the anchor of $C$ are connected to form $G[C]$; $C$ is then pushed onto a stack, to be covered when $G[C]$ is fully covered.

**Algorithm B**

Initialize $S_G - \emptyset$. Starting from an arbitrary vertex, traverse $G$ using EDFS. Immediately before backtracking from a vertex $v$, invoke procedure Backtrack-From($v$), described below.
**Procedure Backtrack-From(v)**

```plaintext
do
  (1) Record the father of v.
  (2) If v is not on a cycle and is a fork, apply Lemma 3.3 to v.
  (3) If v is the source of a cycle, perform the following:
      (3.1) Let $B_v$ be the set of EDFS back-edges entering v. Temporarily suspend
           the EDFS, and for each $e = (u \rightarrow v) \in B_v$, re-traverse
           the remaining edges of $C(e)$ — backtracking the EDFS tree-edges,
           starting from u. Stop the re-traversal of $C(e)$ upon discovering
           a fork. If none exists — count the number of trails starting on $C(e)$.
      (3.2) Let $S = \{ e \in B_v \mid C(e) \text{ contains a fork} \}$. For each $e \in S$,
           apply Lemma 3.3 to a fork on $C(e)$, transferring the isolated
           paths thus created to $S_G$ (tracing the fork can be done
           by re-traversing $C(e)$ once again).
      (3.3) The remaining edges of the cycles of $S$ induce a tree which
           is separated from $G$ by v. Eliminate this tree from $G$,
           transferring the isolated paths thus created to $S_G$.
      (3.4) While $B_v$ contains an edge $e$ such that no edge on $C(e)$ was deleted,
           perform the following:
           (3.4.1) If $C(e)$ is an isolated cycle or is contained in an end-cycle $C$
                   that satisfies the requirements of some edge-deletion lemma,
                   perform the following:
                   Apply that lemma to $C$. The remaining edges of $C$ now induce
                   a tree which is separated from $G$ by v. Eliminate this tree from
                   $G$, transferring the isolated paths thus created to $S_G$.
           (3.4.2) If $C(e)$ underlies a final even-trailed end-cycle $C$
                   perform the following:
                   Let $u$ and $w$ be the vertices adjacent to $v$ that are not
                   on $C$. Delete all the vertices of $C$ from $G$, and connect $u$ and $w$
                   to form $G|C$. Push $C$ onto the stack of the yet-uncovered
                   final even-trailed end-cycles.
  od
```

On completion of Backtrack-From(v), resume the EDFS from the father of v. When the EDFS is done, pop and cover the even-trailed end-cycles stored in the stack, using the recursive decomposition rule.

**Theorem 5.3** (Correctness and Complexity of Algorithm B). Given a cactus $G = (V_G, E_G)$, Algorithm B produces an optimal cover of $G$ in $O(|V_G|)$ time.

**Proof (Outline).** The theorem can be established using the following claims, which, in turn, can be verified by standard techniques for proving the correctness of DFS-based algorithms. In the sequel, we say that $u$ is a descendant of $v$ at a given time within the execution of the algorithm, if, at that time, $u$ is reachable from $v$ by a sequence of tree edges $(v \rightarrow x_1), (x_1 \rightarrow x_2), \ldots, (x_{n-1} \rightarrow x_n), (x_n \rightarrow u)$. 

(1) Upon invoking $\text{Backtrack-From}(v)$, $v$ is on a cycle if and there is a back-edge entering $v$ or lowpoint($v$) $\leq k(v)$ (see [4]). Hence, checking if $v$ is on a cycle can be done in constant time.

(2) Upon invoking $\text{Backtrack-From}(v)$, if $v$ is not on a cycle then no descendant of $v$ is a fork or on a cycle.

(3) Upon invoking $\text{Backtrack-From}(v)$, if $v$ is the source of cycles $C_1, C_2, \ldots, C_n$, then no descendant of $v$, except perhaps for those on some $C_i$ ($1 \leq i \leq n$), is a fork or on a cycle. Hence, for $1 \leq i \leq n$, $C_i$ is either an end-cycle or contains a fork.

(4) If $v$ is the source of cycles $C_1, C_2, \ldots, C_n$ upon invoking $\text{Backtrack-From}(v)$, then on completion of $\text{Backtrack-From}(v)$, no descendant of $v$ is a fork or on a cycle.

(5) Each edge $e \in E_G$ is scanned a constant number of times: Twice by the EDFS, at most twice by the re-traversal, and at most twice by either the elimination process or the covering of the final even-trailed end-cycles.

\section*{Acknowledgment}

The authors wish to thank an anonymous referee, whose remarks have improved the presentation of this paper.

\section*{References}