RWTH Aachen

Department of Computer Science
Technical Report

An optimal algorithm for the k-fixed-endpoint path cover on proper interval graphs

George B. Mertzios and Walter Unger

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2008-09

RWTH Aachen · Department of Computer Science · May 2008

An optimal algorithm for the k-fixed-endpoint path cover on proper interval graphs

George B. Mertzios and Walter Unger
Department of Computer Science
RWTH Aachen University
{mertzios, quax}@cs.rwth-aachen.de

Abstract. In this paper we consider the k-fixed-endpoint path cover problem on proper interval graphs, which is a generalization of the path cover problem. Given a graph G and a set T of k vertices, a k-fixed-endpoint path cover of G with respect to T is a set of vertex-disjoint simple paths that covers the vertices of G, such that the vertices of T are all endpoints of these paths. The goal is to compute a k-fixed-endpoint path cover of G with minimum cardinality. We propose an optimal algorithm for this problem with runtime $O(n)$, where n is the number of intervals in G. This algorithm is based on the *Stair Normal Interval Representation (SNIR) matrix* that characterizes proper interval graphs. In this characterization, every maximal clique of the graph is represented by one matrix element; the proposed algorithm uses this structural property, in order to determine directly the paths in an optimal solution.

1 Introduction

A graph G is called an *interval graph*, if its vertices can be assigned to intervals on the real line, such that two vertices of G are adjacent if and only if the corresponding intervals intersect. The set of intervals assigned to the vertices of G is called a *realization* of G. If G has a realization, in which no interval contains another one properly, then G is called a *proper interval graph*. Proper interval graphs arise naturally in biological applications such as the physical mapping of DNA [1]. Several linear-time recognition algorithms have been presented for both graph classes in the literature [2–5]. These classes of graphs have numerous applications to scheduling problems, biology, VLSI circuit design, as well as to psychology and social sciences [6, 7].

Several difficult optimization problems, which are NP-hard for general graphs [8], are solvable in polynomial time on interval and proper interval graphs. Some of them are the maximum clique, the maximum independent set [9, 10], the Hamiltonian cycle (HC) and the Hamiltonian path (HP) problem [11]. A generalization of the HP problem is the path cover (PC) problem. That is, given a graph G, the goal is to find the minimum number of vertex-disjoint simple paths that cover all vertices of G. Except graph theory, the PC problem finds many applications in the area of database design, networks, code optimization and mapping parallel programs to parallel arcitectures [12–15].

The PC problem is known to be NP-complete even on the classes of planar graphs [16], bipartite graphs, chordal graphs [17], chordal bipartite graphs, strongly chordal graphs [18], as well as in several classes of intersection graphs [19]. On the other hand, it is solvable in linear $O(n + m)$ time on interval graphs with n vertices and m edges [12]. For the greater class of circular-arc graphs there is an optimal $O(n)$-time approximation algorithm, given a set of n arcs with endpoints sorted [20]. The cardinality of the path cover found by this approximation algorithm is at most one more than the optimal one. Several variants of the HP and the PC problems are of great interest. The simplest of them are the 1HP and 2HP problems, where the goal is to decide whether G has a Hamiltonian path with one, or two fixed endpoints, respectively. Both problems are NP-hard for general graphs, as a generalization of the HP problem, while their complexity status remains open for interval graphs [21–23].

In this paper, we consider the k-fixed-endpoint path cover (kPC) problem, which generalizes the PC problem in the following way. Given a graph G and a set T of k vertices, the goal is to find a path cover of G with minimum cardinality, such that the elements of T are endpoints of these paths. Note that the vertices of $V \setminus T$ are allowed to be endpoints of these paths as well. For $k = 1, 2$, the kPC problem constitutes a direct generalization of the 1HP and 2HP problems, respectively. For the case, where the input graph is a cograph on n vertices and m edges, a linear $O(n + m)$ time algorithm for the kPC problem has been recently presented in [22].

We propose an optimal algorithm for the kPC problem on proper interval graphs with runtime $O(n)$, based on the zero-one *Stair Normal Interval Representation (SNIR) matrix* H_G that characterizes a proper interval graph G on n vertices [24]. In this characterization, every maximal clique of G is represented by one matrix element. It provides insight and may be useful for the efficient formulation and solution of difficult optimization problems. In most of the practical applications, the interval endpoints are sorted. Given such an interval realization of G, we construct first in $O(n)$ time a particular perfect ordering of the vertices of G [24], which complies with the ordering of the vertices in the SNIR matrix H_G.

We introduce the notion of a *singular point* in a proper interval graph G on n vertices. An arbitrary vertex of G is called singular point, if it is the unique common vertex of two consecutive maximal cliques. Due to the special structure of H_G, we need to compute only $O(n)$ of its entries, in order to capture the complete information of this matrix. Based on this structure, the proposed algorithm detects the singular points of G in $O(n)$ time and then it determines *directly* the paths in an optimal solution, using only the positions of the singular points. Namely, it turns out that every such path is a Hamiltonian path of a particular subgraph $G_{i,j}$ of G with two specific endpoints. Here, $G_{i,j}$ denotes the induced subgraph of the vertices $\{i, \ldots, j\}$ in the vertex ordering of H_G. Since any algorithm for this problem has to visit at least all n vertices of G, this runtime is optimal.

Recently, while writing this paper, it has been drawn to our attention that another algorithm has been independently presented for the kPC problem on proper interval graphs with runtime $O(n + m)$ [23], where m is the number of edges of the input graph. This algorithm uses a greedy approach to augment the already constructed paths with connect/insert operations, by distinguishing whether these paths have already none, one, or two endpoints in T. The main advantage of the here proposed algorithm, besides its runtime optimality, is that an optimal solution is constructed directly by the positions of the singular points, which is a structural property of the investigated graph. Given an interval realization of the input graph G, we do not need to visit all its edges, exploiting the special structure of the SNIR matrix. Additionally, the representation of proper interval (resp. interval) graphs by the SNIR (resp. NIR) matrix [24] may lead to efficient algorithms for other optimization problems, such as the 1HP, 2HP, or even kPC problem on interval graphs [21, 22].

The paper is organized as follows. In Section 2 we recall the SNIR matrix of a proper interval graph. Furthermore, in Section 3 we present an algorithm for the 2HP, based on the SNIR matrix. This algorithm is used in Section 4, in order to derive an algorithm for the kPC problem on proper interval graphs with runtime $O(n)$. Finally, we discuss some conclusions and open questions for further research in Section 5.

2 The SNIR matrix

An arbitrary proper interval graph G with n vertices $\{1, \ldots, n\}$ can be characterized by the *SNIR matrix* H_G, which has been introduced in [24]. This is the lower portion of the adjacency matrix of G, which uses a particular ordering of its vertices. In this ordering, the vertex with index i corresponds to the i^{th} diagonal element of H_G. All diagonal elements of H_G are zero, i.e. $H_G(i,i) = 0$ for every $i \in \{1,\ldots,n\}$. Every diagonal element has a (possibly empty) chain of consecutive ones immediately below it, while the remaining entries of this column are zero. These chains are ordered in such a way that H_G has a stair-shape, as it is illustrated in Figure 2(a). We recall now the definitions of a stair and a pick of the SNIR matrix H_G [24].

Definition 1. *Consider the SNIR matrix H_G of the proper interval graph G. The matrix element $H_G(i,j)$ is called a* pick *of H_G, iff:*

1. *$i \geq j$,*
2. *if $i > j$ then $H_G(i,j) = 1$,*
3. *$H_G(i,k) = 0$, for every $k \in \{1,2,\ldots,j-1\}$, and*
4. *$H_G(l,j) = 0$, for every $l \in \{i+1,i+2,\ldots,n\}$.*

Definition 2. *Given the pick $H_G(i,j)$ of H_G, the set*

$$\mathcal{S} = \{H_G(k,\ell) : j \leq \ell \leq k \leq i\} \tag{1}$$

of matrix entries is called the stair *of H_G, which corresponds to this pick.*

Lemma 1 ([24]). *Any stair of H_G corresponds bijectively to a maximal clique of G.*

A stair of H_G can be recognized in Figure 2(a), where the corresponding pick is marked with a circle. Given an interval realization of G with sorted endpoints, the ordering of vertices in H_G can be computed in $O(n)$ time [24]. Furthermore, the picks of H_G can be also computed in $O(n)$ time during the construction of the ordering of the vertices, since every pick corresponds to the right endpoint of an interval in G [24]. Due to its stair-shape, the matrix H_G is uniquely determined by its $O(n)$ picks.

For an arbitrary vertex w of G, denote by $s(w)$ and $e(w)$ the adjacent vertices of w with the smallest and greatest index in this ordering, respectively. Due to the stair-shape of H_G, the vertices $s(w)$ and $e(w)$ are the uppermost and lowermost diagonal elements of H_G, which belong to a common stair with w. Denote now the maximal cliques of G by $Q_1, Q_2, \ldots, Q_m$, $m \leq n$ and suppose that the corresponding pick to Q_i is the matrix element $H_G(a_i, b_i)$, where $i \in \{1,\ldots,m\}$. Since the maximal cliques of G, i.e. the stairs of H_G, are linearly ordered, it holds that $1 \leq a_1 \leq \ldots \leq a_m \leq n$ and $1 \leq b_1 \leq \ldots \leq b_m \leq n$. Denote for simplicity $a_0 = b_0 = 0$ and $a_{m+1} = b_{m+1} = n+1$. Then, Algorithm 1 computes the values $s(w)$ and $e(w)$ for all vertices $w \in \{1,\ldots,n\}$, as it is illustrated in Figure 1. Since $m \leq n$, the runtime of Algorithm 1 is $O(n)$.

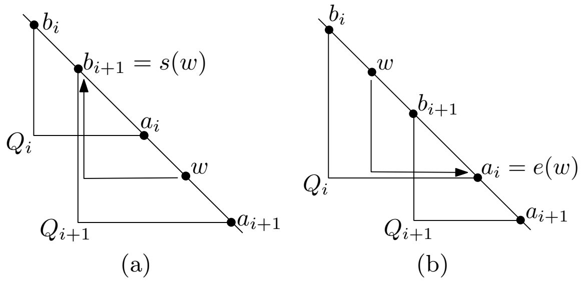
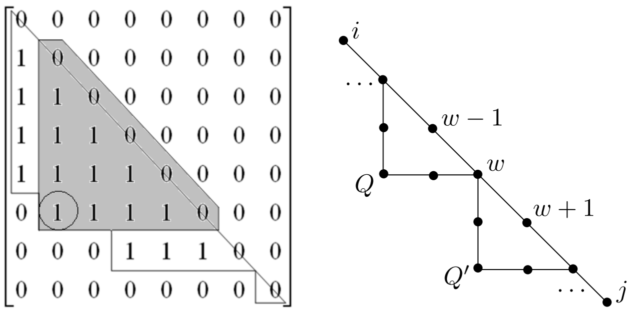


Fig. 1. The computation of $s(w)$ and $e(w)$.

The vertices $\{i,\ldots,j\}$ of G, where $i \leq j$, constitute a submatrix $H_{i,j}$ of H_G, which is equivalent to the induced subgraph $G_{i,j}$ of these vertices. Since the proper interval graphs are hereditary, this subgraph remains a proper interval graph as well. In particular, $H_{1,n} = H_G$ is equivalent to $G_{1,n} = G$.

Algorithm 1 Compute $s(w)$ and $e(w)$ for all vertices w

```
for i = 0 to m do
    for w = a_i + 1 to a_{i+1} do
        s(w) ← b_{i+1}
    for w = b_i to b_{i+1} − 1 do
        e(w) ← a_i
```



(a) (b)

Fig. 2. (a) The SNIR matrix H_G, (b) a singular point w of $G_{i,j}$.

Definition 3. *A vertex w of $G_{i,j}$ is called* singular point *of $G_{i,j}$, if there exist two consecutive cliques Q, Q' of $G_{i,j}$, such that*

$$|Q \cap Q'| = \{w\} \tag{2}$$

Otherwise, w is called regular point *of $G_{i,j}$. The set of all singular points of $G_{i,j}$ is denoted by $S(G_{i,j})$.*

Proposition 1. *For every* singular point *w of $G_{i,j}$, it holds $i + 1 \leq w \leq j - 1$.*

Proof. Since w is a singular point of $G_{i,j}$, there exist two consecutive maximal cliques Q, Q' of $G_{i,j}$ with $Q \cap Q' = \{w\}$. Then, as it is illustrated in Figure 2(b), both Q and Q' contain at least another vertex than w, since otherwise one of them would be included in the other, which is a contradiction. It follows that $i + 1 \leq w \leq j - 1$.

Definition 4. *Consider a connected proper interval graph G and two indices $i \leq j \in \{1, \ldots, n\}$. The submatrix $H_{i,j}$ of H_G is called* two-way matrix*, if all vertices of $G_{i,j}$ are regular points of it. Otherwise, $H_{i,j}$ is called* one-way matrix.

The intuition resulting from Definition 4 is the following. If $H_{i,j}$ is an one-way matrix, then $G_{i,j}$ has at least one singular point w. In this case, no vertex among $\{i, \ldots, w-1\}$ is connected to any vertex among $\{w+1, \ldots, j\}$, as it is illustrated in Figure 2(b). Thus, every Hamiltonian path of $G_{i,j}$ passes only once from the vertices $\{i, \ldots, w-1\}$ to the vertices $\{w+1, \ldots, j\}$, through vertex w. Otherwise, if $H_{i,j}$ is a two-way matrix, a Hamiltonian path may pass more than once from $\{i, \ldots, w-1\}$ to $\{w+1, \ldots, j\}$ and backwards, where w is an arbitrary vertex of $G_{i,j}$. The next corollary follows directly from Proposition 1.

Corollary 1. *An arbitrary vertex w of G is a regular point of the subgraphs $G_{i,w}$ and $G_{w,j}$, for every $i \leq w$ and $j \geq w$.*

3 The 2HP problem on proper interval graphs

3.1 Necessary and sufficient conditions

In this section we solve the 2HP problem on proper interval graphs. In particular, given two fixed vertices u, v of a proper interval graph G, we provide necessary and sufficient conditions for the existence of a Hamiltonian path in G with endpoints u and v. An algorithm with runtime $O(n)$ follows directly from these conditions, where n is the number of vertices of G.

Denote by 2HP(G, u, v) this particular instance of 2HP on G. Since G is equivalent to the SNIR matrix H_G and since this matrix specifies a particular ordering of its vertices, we identify w.l.o.g. the vertices of G with their indices in this ordering. Observe at first that if G is not connected, then there is no Hamiltonian path at all in G. Also, if G is connected with only two vertices u, v, then there exists trivially a Hamiltonian path with u and v as endpoints. Thus, we assume in the following that G is connected and $n \geq 3$. The next Theorems 1 and 2 provide necessary and sufficient conditions for the existence of a Hamiltonian path with endpoints u, v in a connected proper interval graph G.

Theorem 1. *Let G be a connected proper interval graph and u, v be two vertices of G, with $v \geq u+2$. There is a Hamiltonian path in G with u, v as endpoints if and only if the submatrices $H_{1,u+1}$ and $H_{v-1,n}$ of H_G are two-way matrices.*

Proof. Suppose that $H_{1,u+1}$ is an one-way matrix. Then, due to Definition 4, $G_{1,u+1}$ has at least one singular point w. Since $G_{1,u+1}$ is connected as an induced subgraph of G, Proposition 1 implies that $2 \leq w \leq u$.

In order to obtain a contradiction, let P be a Hamiltonian path in G with u and v as its endpoints. Suppose first that for the singular point w it holds $w < u$. Then, due to the stair-shape of H_G, the path P has to visit w in order to reach the vertices $\{1, \ldots, w-1\}$. On the other hand, P has to visit w again in order to reach v, since $w < v$. This is a contradiction, since P visits w exactly once. Suppose now that $w = u$. The stair-shape of H_G implies that u has to be connected in P with at least one vertex of $\{1, \ldots, u-1\}$ and with at least one vertex of $\{u+1, \ldots, n\}$. This is also a contradiction, since u is an endpoint of P. Therefore, there exists no such path P in G, if $H_{i,u+1}$ is an one-way matrix. Similarly, we obtain that there exists again no such path P in G, if $H_{v-1,n}$ is an one-way matrix. This completes the necessity part of the proof.

For the sufficiency part, suppose that both $H_{1,u+1}$ and $H_{v-1,n}$ are two-way matrices. Then, Algorithm 2 constructs a Hamiltonian path P in G having u and v as endpoints, as follows. In the while-loop of the lines 2-4 of Algorithm 2, P starts from vertex u and reaches vertex 1 using sequentially the uppermost diagonal elements, i.e. vertices, of the visited stairs of H_G. Since $H_{1,u+1}$ is a two-way matrix, P does not visit any two consecutive diagonal elements until it reaches vertex 1. In the while-loop of the lines 5-10, P continues visiting all unvisited vertices until vertex $v-1$. Let t be the actual visited vertex of P during these lines. Since P did not visit any two consecutive diagonal elements until it reached vertex 1 in lines 2-4, at least one of the vertices $t+1$, $t+2$ has not been visited yet. Thus, always one of the lines 7 and 10 is executed.

Next, in the while-loop of the lines 11-13, P starts from vertex $v-1$ and reaches vertex n using sequentially the lowermost diagonal elements of the visited stairs of H_G. During the execution of lines 11-13, since $H_{v-1,n}$ is a two-way matrix, P does not visit any two consecutive diagonal elements until it reaches vertex n. Finally, in the while-loop of the lines 14-18, P continues visiting all unvisited vertices until v. Similarly to the lines 5-10, let t be the actual visited vertex of P. Since P did not visit any two consecutive diagonal elements until it reached vertex n in lines 11-13, at least one of the vertices $t-1$, $t-2$ has not been visited yet. Thus, always one of the lines 16 and 18 is executed. Figure 3(a) illustrates the construction of such a Hamiltonian path by Algorithm 2 in a small example.

Theorem 2. *Let G be a connected proper interval graph and u be a vertex of G. There is a Hamiltonian path in G with $u, u+1$ as endpoints if and only if H_G is a two-way matrix and either $u \in \{1, n-1\}$ or the vertices $u-1$ and $u+2$ are adjacent.*

Proof. Suppose that H_G is an one-way matrix. Then, at least one of the matrices $H_{1,u+1}$ and $H_{u,n}$ is one-way matrix. Similarly to the proof of Theorem 1, there is no Hamiltonian path in G having as endpoints the vertices u and $v = u + 1$.

Suppose now that H_G is a two-way matrix and let $u \in \{2, \ldots, n-2\}$. Then, both vertices $u-1$ and $u+2$ exist in G. Since the desired path P starts at u and ends at $u+1$, at least one vertex in $\{1, \ldots, u-1\}$ has to be connected to at least one vertex in $\{u+2, \ldots, n\}$. Thus, due to the stair-shape of H_G, it follows that the vertices $u-1$ and $u+2$ are connected. This completes the necessity part of the proof.

For the sufficiency part, suppose that the conditions of Theorem 2 hold. Then, Algorithm 2 constructs a Hamiltonian path P in G having u and $u+1$ as endpoints. The only differences from the proof of Theorem 1 about the correctness of Algorithm 2 are the following. If $u = 1$, the lines 2-10 are not executed at all. In this case, P visits all vertices of G during the execution of lines 11-18, exactly as in the proof of Theorem 1. If $u \geq 2$, none of the lines 7 and 10 of Algorithm 2 is executed when P visits vertex $t = u - 1$, since in this case it holds that $t + 1 = u \in P$ and $t + 2 = u + 1 \in P \cup \{u+1\}$. If $u + 1 = n$, then P visits the last vertex $u+1$ in lines 12 and 13. Otherwise, if $u + 1 < n$, the vertices $u-1$ and $u+2$ are adjacent, due to the conditions of Theorem 2. In this case, P continues visiting all the remaining vertices of G, as in the proof of Theorem 1. Figure 3(b) illustrates the construction of such a Hamiltonian path by Algorithm 2 in a small example.

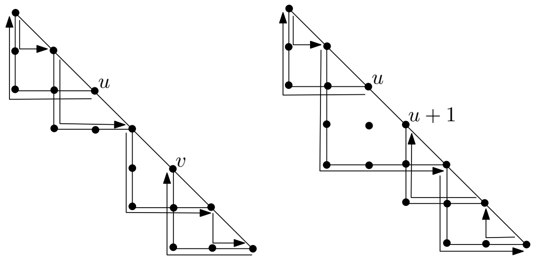
Algorithm 2 Construct a Hamiltonian path P in G with u, v as endpoints

1: $t \leftarrow u$; $P \leftarrow \{u\}$
2: **while** $t > 1$ **do**
3:     $p \leftarrow s(t)$ {the adjacent vertex of t with the smallest index}
4:     $P \leftarrow P \circ p$; $t \leftarrow p$
5: **while** $t < v - 1$ **do**
6:     **if** $(t+1) \notin P$ **then**
7:         $P \leftarrow P \circ (t+1)$; $t \leftarrow t + 1$
8:     **else**
9:         **if** $(t+2) \notin P \cup \{v\}$ **then**
10:             $P \leftarrow P \circ (t+2)$; $t \leftarrow t + 2$
11: **while** $t < n$ **do**
12:     $p \leftarrow e(t)$ {the adjacent vertex of t with the greatest index}
13:     $P \leftarrow P \circ p$; $t \leftarrow p$
14: **while** $t > v$ **do**
15:     **if** $(t-1) \notin P$ **then**
16:         $P \leftarrow P \circ (t-1)$; $t \leftarrow t - 1$
17:     **else**
18:         $P \leftarrow P \circ (t-2)$; $t \leftarrow t - 2$
19: **return** P

If the conditions of Theorems 1 and 2 are satisfied, Algorithm 2 constructs a Hamiltonian path with endpoints u, v, as it is described in the proofs of these theorems. Algorithm 2 operates on every vertex of G at most twice. Thus, since all values $s(t)$ and $e(t)$ can be computed in $O(n)$ time, its runtime is $O(n)$ as well. Figure 3 illustrates the construction of such a Hamiltonian path by Algorithm 2 in a small example, for both cases $v \geq u + 2$ and $v = u + 1$.

3.2 The decision of 2HP in $O(n)$ time

We can use now the results of Section 3.1 in order to decide in $O(n)$ time whether a given proper interval graph G has a Hamiltonian path P with two specific endpoints u, v and to construct P,



(a) (b)

Fig. 3. The construction of the HP with endpoints u, v where (a) $v \geq u+2$, (b) $v = u+1$.

if it exists. The values $s(w)$ and $e(w)$ for all vertices $w \in \{1, \ldots, n\}$ can be computed in $O(n)$ time. Due to the stair-shape of H_G, the graph G is not connected if and only if there is a vertex $w \in \{1, \ldots, n-1\}$, for which it holds $e(w) = w$ and thus, we can check the connectivity of G in $O(n)$ time. If G is not connected, then it has no Hamiltonian path at all. Finally, a vertex w is singular if and only if $e(w-1) = s(w+1) = w$ and thus, the singular points of G can be computed in $O(n)$.

Since the proper interval graphs are hereditary, the subgraphs $G_{1,u+1}$ and $G_{v-1,n}$ of G remain proper interval graphs as well. Thus, if G is connected, we can check in $O(n)$ time whether these graphs have singular points, or equivalently, whether $H_{1,u+1}$ and $H_{v-1,n}$ are two-way matrices. On the other hand, we can check in constant time whether the vertices $u-1$ and $u+2$ are adjacent. Thus, we can decide in $O(n)$ time whether there exists a Hamiltonian path in G with endpoints u, v, due to Theorems 1 and 2. In the case of non-existence, we output "NO", while otherwise Algorithm 2 constructs in $O(n)$ time the desired Hamiltonian path.

4 The kPC problem on proper interval graphs

4.1 The algorithm

In this section we present Algorithm 3, which solves in $O(n)$ the k-fixed-endpoint path cover (kPC) problem on a proper interval graph G with n vertices, for any $k \leq n$. This algorithm uses the characterization of the 2HP problem of the previous section. We assume that for the given set $T = \{t_1, t_2, \ldots, t_k\}$ it holds $t_1 < t_2 < \ldots < t_k$. Denote also for simplicity $t_{k+1} = n+1$.

Algorithm 3 Compute $C(G, T)$ for a proper interval graph G

if $G = \emptyset$ **then**
  return $\emptyset$
Compute the values $s(w)$ and $e(w)$ for every vertex w
$w \leftarrow 1$
while $w < n$ **do**
  if $e(w) = w$ **then** $\{G$ is not connected$\}$
    $T_1 \leftarrow T \cap \{1, 2, \ldots, w\}$; $T_2 \leftarrow T \setminus T_1$
    return $C(G_{1,w}, T_1) \cup C(G_{w+1,n}, T_2)$
  $w \leftarrow w + 1$
if $k \leq 1$ **then**
  call Algorithm 4
if $t_1 \in S(G)$ **then**
  $P_1 \leftarrow 1 \circ \ldots \circ t_1$
  return $\{P_1\} \cup C(G_{t_1+1,n}, T \setminus \{t_1\})$
call Algorithm 5

Algorithm 3 computes an optimal path cover $C(G,T)$ of G. In lines 4-9, it checks the connectivity of G. If it is not connected, the algorithm computes in lines 7-8 recursively the optimal solutions of the first connected component and of the remaining graph. It reaches line 10 only if G is connected. In the case $|T| = k \leq 1$, Algorithm 3 calls Algorithm 4 as subroutine.

In lines 12-14, Algorithm 3 considers the case, where G is connected, $|T| \geq 2$ and t_1 is a singular point of G. Then, Proposition 1 implies that $2 \leq t_1 \leq n-1$. Since no vertex among $\{1, \ldots, t_1 - 1\}$ is connected to any vertex among $\{t_1 + 1, \ldots, n\}$ and since $t_1 \in T$, an optimal solution must contain at least two paths. Thus, it is always optimal to choose in line 13 a path that visits sequentially the first t_1 vertices and then to compute recursively in line 14 an optimal solution in the remaining graph $G_{t_1+1,n}$. Algorithm 3 reaches line 15 if G is connected, $|T| \geq 2$ and t_1 is a regular point of G. In this case, it calls Algorithm 5 as subroutine.

Algorithm 4 Compute $C(G,T)$, if G is connected and $|T| \leq 1$

```
1: if k = 0 then
2:    return {1 ∘ 2 ∘ ... ∘ n}
3: if k = 1 then
4:    if t_1 ∈ {1, n} then
5:       return {1 ∘ 2 ∘ ... ∘ n}
6:    else
7:       P_1 ← 2HP(G, 1, t_1)
8:       P_2 ← 2HP(G, t_1, n)
9:       if P_1="NO" then
10:         if P_2="NO" then
11:            return {1 ∘ ... ∘ t_1} ∪ {(t_1 + 1) ∘ ... ∘ n}
12:         else
13:            return {P_2}
14:      else
15:         return {P_1}
```

Algorithm 4 computes an optimal path cover $C(G,T)$ of G in the case, where G is connected and $|T| = k \leq 1$. If $k = 0$, then the optimal solution includes clearly only one path, which visits sequentially the vertices $1, 2, \ldots, n$, since G is connected. Let now $k = 1$. If $t_1 \in \{1, n\}$, then the optimal solution is again the single path $\{1, 2, \ldots, n\}$. Otherwise, suppose that $t_1 \in \{2, \ldots, n-1\}$. In this case, a trivial path cover is that with the paths $\{1 \circ \ldots \circ t_1\}$ and $\{(t_1+1) \circ \ldots \circ n\}$. This path cover is not optimal if and only if G has a Hamiltonian path P with $u = t_1$ as one endpoint. The other endpoint v of P lies either in $\{1, \ldots, t_1 - 1\}$ or in $\{t_1 + 1, \ldots, n\}$. If $v \in \{t_1 + 1, \ldots, n\}$, then H_{1,t_1+1} and $H_{v-1,n}$ have to be two-way matrices, due to Theorems 1 and 2. However, due to Definition 4, if $H_{v-1,n}$ is a two-way matrix, then $H_{n-1,n}$ is also a two-way matrix, since $H_{n-1,n}$ is a trivial submatrix of $H_{v-1,n}$.

Thus, if such a Hamiltonian path with endpoints t_1 and v exists, then there exists also one with endpoints t_1 and n. Similarly, if there exists a Hamiltonian path with endpoints $v \in \{1, \ldots, t_1 - 1\}$ and t_1, then there exists also one with endpoints 1 and t_1. Thus, we call $P_1 = \mathrm{2HP}(G, 1, t_1)$ and $P_2 = \mathrm{2HP}(G, t_1, n)$ in lines 7 and 8, respectively. If both outputs are "NO", then $\{1 \circ \ldots \circ t_1\}$ and $\{(t_1+1) \circ \ldots \circ n\}$ constitute an optimal solution. Otherwise, we return one of the obtained paths P_1 or P_2 in lines 15 or 13, respectively. Since the runtime of Algorithm 2 for the 2HP problem is $O(n)$, the runtime of Algorithm 4 is $O(n)$ as well.

In lines 5-9 and 12-14, Algorithm 3 separates G in two subgraphs and computes their optimal solutions recursively. Thus, since the computation of all values $s(w)$ and $e(w)$ can be done in $O(n)$ and since the runtime of Algorithms 4 and 5 is $O(n)$, Algorithm 3 runs in $O(n)$ time as well.

Algorithm 5 Compute $C(G,T)$, where G is connected, $|T| \geq 2$, $t_1 \notin S(G)$.

if $\{1, ..., t_1 - 1\} \cap S(G) = \emptyset$ **then** $\{e_1 = t_2\}$
  if $2\text{HP}(G_{1,t_2+1}, t_1, t_2) =$"NO" **then**
    $a \leftarrow t_2$
  else
    if $\{t_2 + 1, ..., t_3 - 1\} \cap S(G) \neq \emptyset$ **then**
      $a \leftarrow \min\{\{t_2 + 1, ..., t_3 - 1\} \cap S(G)\}$
    else
      $a \leftarrow t_3 - 1$
  $P_1 \leftarrow 2\text{HP}(G_{1,a}, t_1, t_2)$
  $C_2 \leftarrow C(G_{a+1,n}, T \setminus \{t_1, t_2\})$
else $\{e_1 = 1\}$
  if $2\text{HP}(G_{1,t_1+1}, 1, t_1) =$"NO" **then**
    $a \leftarrow t_1$
  else
    if $\{t_1 + 1, ..., t_2 - 1\} \cap S(G) \neq \emptyset$ **then**
      $a \leftarrow \min\{\{t_1 + 1, ..., t_2 - 1\} \cap S(G)\}$
    else
      $a \leftarrow t_2 - 1$
  $P_1 \leftarrow 2\text{HP}(G_{1,a}, 1, t_1)$
  $C_2 \leftarrow C(G_{a+1,n}, T \setminus \{t_1\})$
return $\{P_1\} \cup C_2$

4.2 Correctness of Algorithm 5

The correctness of Algorithm 5 follows from the technical Lemmas 3 and 4. To this end, we prove first the auxiliary Lemma 2. For the purposes of these proofs, we assume an optimal solution C of G. Denote by P_i the path in C, which has t_i as endpoint and let e_i be its second endpoint. Observe that, if $e_i = t_j$, then $P_i = P_j$. Let further ℓ_i be the vertex of P_i with the greatest index in the ordering of H_G. It holds clearly $\ell_i \geq t_i$, for every $i \in \{1, \ldots, k\}$.

Lemma 2. *If $e_1 \leq t_1$, then w.l.o.g. $\ell_1 < t_2$ and $e_1 = 1$.*

Proof. At first, suppose that $e_1 = t_1$, i.e. P_1 is a trivial path of one vertex. If $t_1 = 1$, the lemma holds obviously. Otherwise, we can extend P_1 by visiting sequentially the vertices $t_1 - 1, \ldots, 1$. Since there is no vertex of T among the vertices $\{1, \ldots, t_1 - 1\}$, the resulting path cover has not greater cardinality than C and $e_1 = 1$.

Let now $e_1 < t_1$. Suppose that $\ell_1 \geq t_2$. Thus, since ℓ_1 is not an endpoint of P_1, it holds that $t_i < \ell_1$ for some $i \in \{2, \ldots, k\}$. Suppose first that $t_i < \ell_1 < \ell_i$, as it is illustrated in Figure 4(a). Then, we can clearly transfer to P_i all vertices of P_1 with index between $t_i + 1$ and ℓ_1. The obtained path cover has the same cardinality as C, while the greatest index of the vertices of P_1 is less than t_i.

Suppose now that $t_i < \ell_i < \ell_1$, as it is illustrated in Figure 4(b). Since $e_1 < t_1$, the path P_1 is a Hamiltonian path of some subgraph of G_{1,ℓ_1} with endpoints e_1 and t_1. Now, we obtain similarly to the proofs of Theorems 1 and 2 that H_{t_1-1,ℓ_1} is a two-way matrix, since otherwise the path P_1 would visit two times the same vertex, which is a contradiction. It follows that H_{ℓ_i-1,ℓ_1} is also a two-way matrix, as a submatrix of H_{t_1-1,ℓ_1}. Thus, we can extend P_i by the vertices of P_1 with index between $\ell_i + 1$ and ℓ_1. In the obtained path cover, the greatest index ℓ'_1 of the vertices of P_1 is less than ℓ_i. Finally, if $t_i < \ell'_1$, we can obtain, similarly to the above, a new path cover with the same cardinality as C, in which the greatest index of the vertices of P_1 is less than t_i.

It follows now by induction that there is an optimal solution, in which the greatest index ℓ_1 of the vertices of P_1 is less than t_2, as it is illustrated in Figure 4(c). Then, similarly to above,

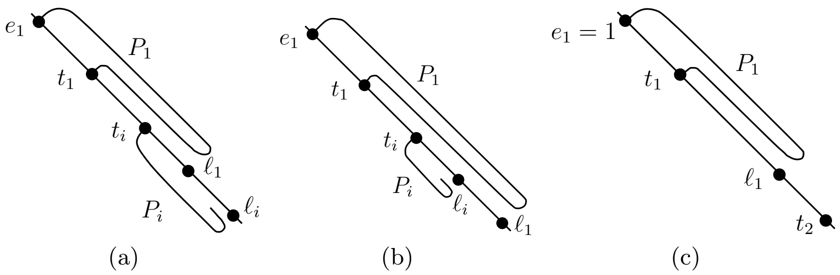


Fig. 4. The case $e_1 \leq t_1$.

H_{t_1-1,ℓ_1} is a two-way matrix. Now, Theorems 1 and 2 imply that G_{1,ℓ_1} has a Hamiltonian path with 1 and t_1 as endpoints. Thus, it is always optimal to choose $P_1 = 2\text{HP}(G_{1,\ell_1}, 1, t_1)$, for some $\ell_1 \in \{t_1, \ldots, t_2 - 1\}$.

Lemma 3. *If* $\{1, \ldots, t_1\} \cap S(G) = \emptyset$, *then w.l.o.g.* $e_1 = t_2$.

Proof. Suppose at first that $e_1 \leq t_1$. Then, Lemma 2 implies that $e_1 = 1$ and $\ell_1 < t_2$. In particular, the proof of Lemma 2 implies that $P_1 = 2\text{HP}(G_{1,\ell_1}, 1, t_1)$, as it is illustrated in Figure 5(a). Thus, since P_1 visits all vertices $\{1, 2, \ldots, \ell_1\}$, it holds that

$$|C| = 1 + |C(G_{\ell_1+1,n}, T \setminus \{t_1\})| \tag{3}$$

Suppose now that $e_1 > t_1$. Since there are no singular points of G among $\{1, \ldots, t_1\}$, the submatrix H_{1,t_1+1} is a two-way matrix. Then, Theorems 1 and 2 imply that G_{1,t_2} has a Hamiltonian path with endpoints t_1 and t_2. Thus, we may suppose w.l.o.g. that $P_1 = 2\text{HP}(G_{1,a}, t_1, t_2)$, for an appropriate $a \geq t_2$, as it is illustrated in Figure 5(b). Since $P_1 = P_2$ and thus $e_2 = t_1 < t_2$, we obtain similarly to Lemma 2 that $a = \ell_2 < t_3$. Since P_1 visits all vertices $\{1, 2, \ldots, a\}$, it follows in this case for the cardinality of C that

$$|C| = 1 + |C(G_{a+1,n}, T \setminus \{t_1, t_2\})| \tag{4}$$

Since in (3) it holds $\ell_1 < t_2$ and in (4) it holds $a \geq t_2$, it follows that $G_{a+1,n}$ is a strict subgraph of $G_{\ell_1+1,n}$. Since $T \setminus \{t_1, t_2\}$ is a subset of $T \setminus \{t_1\}$, it follows that the quantity in (4) is less than or equal to that in (3). Thus, we may suppose w.l.o.g. that $e_1 = t_2$.

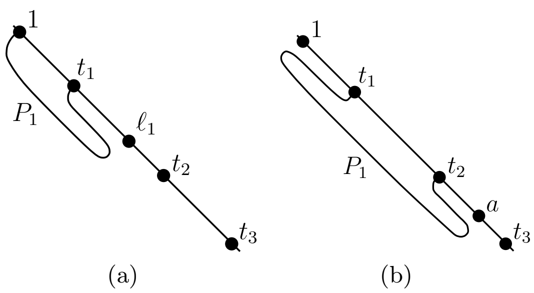


Fig. 5. The case, where there is no singular point of G among $\{1, \ldots, t_1\}$.

Lemma 4. *If* $\{1, \ldots, t_1 - 1\} \cap S(G) \neq \emptyset$ *and* $t_1 \notin S(G)$, *then w.l.o.g.* $e_1 = 1$.

Proof. Let $w \in \{1, \ldots, t_1 - 1\}$ be the singular point of G with the smallest index. Due to Proposition 1, it holds $w \geq 2$. Then, there is a path P_0 in the optimal solution C, which has an endpoint $t_0 \in \{1, \ldots, w-1\}$. Indeed, otherwise there would be a path visiting vertex w at least twice, which is a contradiction.

Thus, since $\{1, \ldots, t_0\} \cap S(G) = \emptyset$ and since t_0 is an endpoint, Lemma 3 implies for the other endpoint e_0 of P_0 that $e_0 = t_1$ and therefore $P_0 = P_1$. Thus, since the second endpoint of P_1 is $e_1 = t_0 < t_1$, Lemma 2 implies that w.l.o.g. it holds $e_1 = t_0 = 1$ and, in particular that $P_1 = 2\mathrm{HP}(G_{1,a}, 1, t_1)$ for some $a \in \{t_1, \ldots, t_2 - 1\}$, as it is illustrated in Figure 6.

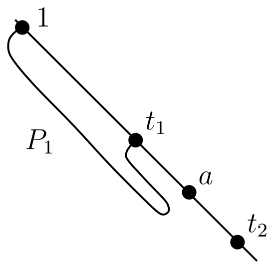


Fig. 6. The case, where there are singular points of G among $\{1, \ldots, t_1 - 1\}$ and t_1 is a regular point of G.

Algorithm 5 considers in lines 1-10 the case where there are no singular points of G among $\{1, \ldots, t_1 - 1\}$. The proof of Lemma 3 implies for this case that $e_1 = t_2$ and, in particular that $P_1 = 2\mathrm{HP}(G_{1,a}, t_1, t_2)$ for some $a \in \{t_2, \ldots t_3 - 1\}$. In order to maximize P_1 as much as possible, we choose the greatest possible value of a, for which $G_{1,a}$ has a Hamiltonian path with endpoints t_1, t_2. Namely, if G_{1,t_2+1} does not have such a Hamiltonian path, we set $a = t_2$ in line 3. Suppose now that G_{1,t_2+1} has such a path. In the case, where there is at least one singular point of G among $\{t_2 + 1, \ldots, t_3 - 1\}$, we set a to be this one with the smallest index among them in line 6. Otherwise, we set $a = t_3 - 1$ in line 8. Denote for simplicity $G_{1,n+1} = G$. Then, in the extreme cases $t_3 = t_2 + 1$ or $t_2 = n$, the algorithm sets $a = t_2 = t_3 - 1$.

Next, in lines 11-20, Algorithm 5 considers the case, where there are some singular points of G among $\{1, \ldots, t_1 - 1\}$. Then, the proof of Lemma 4 implies that $e_1 = 1$ and, in particular that $P_1 = 2\mathrm{HP}(G_{1,a}, 1, t_1)$, for some $a \in \{t_1, \ldots, t_2 - 1\}$. In order to maximize P_1 as much as possible, we choose the greatest possible value of a, for which $G_{1,a}$ has a Hamiltonian path with endpoints 1 and t_1. Namely, if G_{1,t_1+1} does not have such a Hamiltonian path, we set $a = t_1$ in line 13. Suppose now that G_{1,t_1+1} has such a path. In the case, where there is at least one singular point of G among $\{t_1 + 1, \ldots, t_2 - 1\}$, we set a to be this one with the smallest index among them in line 16. Otherwise, we set $a = t_2 - 1$ in line 18. Note that in the extreme case $t_2 = t_1 + 1$, the algorithm sets $a = t_1 = t_2 - 1$.

The algorithm computes P_1 in lines 9 and 19, respectively. Then, it computes recursively the optimum path cover C_2 of the remaining graph in lines 10 and 20, respectively, and it outputs $\{P_1\} \cup C_2$. Since the computation of a 2HP by Algorithm 2 can be done in $O(n)$ time, the runtime of Algorithm 5 is $O(n)$ as well.

5 Concluding remarks

In this article we presented a simple algorithm for the k-fixed-endpoint path cover problem on proper interval graphs with runtime $O(n)$. Since any algorithm for this problem has to visit at least all n vertices of G, this runtime is optimal. The presented algorithm is based on the characterization of proper interval graphs by the SNIR matrix. The complexity status of the k-fixed-endpoint path cover problem, as well as of 1HP and 2HP, on the general class of interval graphs remain interesting open questions for further research.

References

1. P. Hell, R. Shamir, and R. Sharan. A fully dynamic algorithm for recognizing and representing proper interval graphs. *SIAM J. Comput.*, 31(1):289–305, 2001.
2. W.L. Hsu. A simple test for interval graphs. In *WG '92: Proceedings of the 18th International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 11–16, London, 1993. Springer-Verlag.
3. D. Corneil, H. Kim, S. Natarajan, S. Olariu, and A.P. Sprague. Simple linear time recognition of unit interval graphs. *Inform. Process. Lett.*, 55:99–104, 1995.
4. D.G. Corneil, S. Olariu, and L. Stewart. The ultimate interval graph recognition algorithm? In *SODA '98: Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pages 175–180, 1998.
5. B.S. Panda and S.K. Das. A linear time recognition algorithm for proper interval graphs. *Information Processing Letters*, 87(3):153–161, 2003.
6. M.C. Golumbic and A.N. Trenk. *Tolerance graphs.* Cambridge University Press, Cambridge, 2004.
7. A.V. Carrano. Establishing the order to human chromosome-specific DNA fragments. In A. D. Woodhead and B. J. Barnhart, editors, *Biotechnology and the Human Genome*, pages 37–50. Plenum Press, New York, 1988.
8. M.R. Garey and D.S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* W.H. Freeman, San Francisco, 1979.
9. U.I. Gupta, D.T. Lee, and J.Y.T. Leung. Efficient algorithms for interval graphs and circular-arc graphs. *Networks*, pages 459–467, 1982.
10. Ju Yuan Hsiao and Chuan Yi Tang. An efficient algorithm for finding a maximum weight 2-independent set on interval graphs. *Inf. Process. Lett.*, 43(5):229–235, 1992.
11. M.S. Chang, S.L. Peng, and J.L. Liaw. Deferred-query - an efficient approach for problems on interval and circular-arc graphs (extended abstract). In *WADS*, pages 222–233, 1993.
12. S.R. Arikati and C.P. Rangan. Linear algorithm for optimal path cover problem on interval graphs. *Information Processing Letters*, 35(3):149–153, 1990.
13. G.S. Adhar and S. Peng. Parallel algorithms for path covering, hamiltonian path and hamiltonian cycle in cographs. In *International Conference on Parallel Processing*, volume 3, pages 364–365, 1990.
14. R. Lin, S. Olariu, and G. Pruesse. An optimal path cover algorithm for cographs. *Comput. Math. Appl.*, 30:75–83, 1995.
15. R. Srikant, R. Sundaram, K.S. Singh, and C.P. Rangan. Optimal path cover problem on block graphs and bipartite permutation graphs. *Theoretical Computer Science*, 115:351–357, 1993.
16. M.R. Garey, D.S. Johnson, and R.E. Tarjan. The planar hamiltonian circuit problem is np-comlete. *SIAM J. Comput.*, 5:704–714, 1976.
17. M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, volume 57. Annals of Discrete Mathematics, Amsterdam, The Netherlands, 2004.
18. H. Müller. Hamiltonian circuits in chordal bipartite graphs. *Discrete Mathematics*, 156:291–298, 1996.
19. A.A. Bertossi and M.A. Bonucelli. Finding hamiltonian circuits in interval graph generalizations. *Information Processing Letters*, 23:195–200, 1986.
20. R.W. Hung and M.S. Chang. Solving the path cover problem on circular-arc graphs by using an approximation algorithm. *Discrete Applied Mathematics*, 154(1):76–105, 2006.
21. P. Damaschke. Paths in interval graphs and circular-arc graphs. *Discrete Mathematics*, 112:49–64, 1993.
22. K. Asdre and S.D. Nikolopoulos. A linear-time algorithm for the k-fixed-endpoint path cover problem on cographs. *Networks*, 50:231–240, 2007.
23. K. Asdre and S.D. Nikolopoulos. A polynomial solution to the k-fixed-endpoint path cover problem on proper interval graphs. In *18th International Conference on Combinatorial Algorithms (IWOCA'07)*, Newcastle, Australia, 2007.
24. G.B. Mertzios. A matrix characterization of interval and proper interval graphs. *Applied Mathematics Letters*, 21(4):332–337, 2008.

Aachener Informatik-Berichte

This list contains all technical reports published during the past five years. A complete list of reports dating back to 1987 is available from http://aib.informatik.rwth-aachen.de/. To obtain copies consult the above URL or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

2003-01 *	Jahresbericht 2002
2003-02	Jürgen Giesl, René Thiemann: Size-Change Termination for Term Rewriting
2003-03	Jürgen Giesl, Deepak Kapur: Deciding Inductive Validity of Equations
2003-04	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Improving Dependency Pairs
2003-05	Christof Löding, Philipp Rohde: Solving the Sabotage Game is PSPACE-hard
2003-06	Franz Josef Och: Statistical Machine Translation: From Single-Word Models to Alignment Templates
2003-07	Horst Lichter, Thomas von der Maßen, Alexander Nyßen, Thomas Weiler: Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung
2003-08	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Mechanizing Dependency Pairs
2004-01 *	Fachgruppe Informatik: Jahresbericht 2003
2004-02	Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic
2004-03	Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting
2004-04	Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming
2004-05	Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming
2004-06	Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming
2004-07	Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination
2004-08	Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information
2004-09	Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity
2004-10	Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules
2005-01 *	Fachgruppe Informatik: Jahresbericht 2004
2005-02	Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: "Aachen Summer School Applied IT Security"
2005-03	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions
2005-04	Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem

2005-05 Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots

2005-06 Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information

2005-07 Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks

2005-08 Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut

2005-09 Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures

2005-10 Benedikt Bollig: Automata and Logics for Message Sequence Charts

2005-11 Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture

2005-12 Neeraj Mittal, Felix Freiling, S. Venkatesan, Lucia Draque Penso: Efficient Reductions for Wait-Free Termination Detection in Faulty Distributed Systems

2005-13 Carole Delporte-Gallet, Hugues Fauconnier, Felix C. Freiling: Revisiting Failure Detection and Consensus in Omission Failure Environments

2005-14 Felix C. Freiling, Sukumar Ghosh: Code Stabilization

2005-15 Uwe Naumann: The Complexity of Derivative Computation

2005-16 Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)

2005-17 Uwe Naumann: Syntax-directed Derivative Code (Part II: Intraprocedural Adjoint Code)

2005-18 Thomas von der Maßen, Klaus Müller, John MacGregor, Eva Geisberger, Jörg Dörr, Frank Houdek, Harbhajan Singh, Holger Wußmann, Hans-Veit Bacher, Barbara Paech: Einsatz von Features im Software-Entwicklungsprozess - Abschlußbericht des GI-Arbeitskreises "Features"

2005-19 Uwe Naumann, Andre Vehreschild: Tangent-Linear Code by Augmented LL-Parsers

2005-20 Felix C. Freiling, Martin Mink: Bericht über den Workshop zur Ausbildung im Bereich IT-Sicherheit Hochschulausbildung, berufliche Weiterbildung, Zertifizierung von Ausbildungsangeboten am 11. und 12. August 2005 in Köln organisiert von RWTH Aachen in Kooperation mit BITKOM, BSI, DLR und Gesellschaft fuer Informatik (GI) e.V.

2005-21 Thomas Noll, Stefan Rieger: Optimization of Straight-Line Code Revisited

2005-22 Felix Freiling, Maurice Herlihy, Lucia Draque Penso: Optimal Randomized Fair Exchange with Secret Shared Coins

2005-23 Heiner Ackermann, Alantha Newman, Heiko Röglin, Berthold Vöcking: Decision Making Based on Approximate and Smoothed Pareto Curves

2005-24 Alexander Becher, Zinaida Benenson, Maximillian Dornseif: Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks

2006-01 * Fachgruppe Informatik: Jahresbericht 2005

2006-02 Michael Weber: Parallel Algorithms for Verification of Large Systems

2006-03 Michael Maier, Uwe Naumann: Intraprocedural Adjoint Code Generated by the Differentiation-Enabled NAGWare Fortran Compiler

2006-04 Ebadollah Varnik, Uwe Naumann, Andrew Lyons: Toward Low Static Memory Jacobian Accumulation

2006-05 Uwe Naumann, Jean Utke, Patrick Heimbach, Chris Hill, Derya Ozyurt, Carl Wunsch, Mike Fagan, Nathan Tallent, Michelle Strout: Adjoint Code by Source Transformation with OpenAD/F

2006-06 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Divide-and-Color

2006-07 Thomas Colcombet, Christof Löding: Transforming structures by set interpretations

2006-08 Uwe Naumann, Yuxiao Hu: Optimal Vertex Elimination in Single-Expression-Use Graphs

2006-09 Tingting Han, Joost-Pieter Katoen: Counterexamples in Probabilistic Model Checking

2006-10 Mesut Günes, Alexander Zimmermann, Martin Wenig, Jan Ritzerfeld, Ulrich Meis: From Simulations to Testbeds - Architecture of the Hybrid MCG-Mesh Testbed

2006-11 Bastian Schlich, Michael Rohrbach, Michael Weber, Stefan Kowalewski: Model Checking Software for Microcontrollers

2006-12 Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, Martin Leucker: Replaying Play in and Play out: Synthesis of Design Models from Scenarios by Learning

2006-13 Wong Karianto, Christof Löding: Unranked Tree Automata with Sibling Equalities and Disequalities

2006-14 Danilo Beuche, Andreas Birk, Heinrich Dreier, Andreas Fleischmann, Heidi Galle, Gerald Heller, Dirk Janzen, Isabel John, Ramin Tavakoli Kolagari, Thomas von der Maßen, Andreas Wolfram: Report of the GI Work Group “Requirements Management Tools for Product Line Engineering”

2006-15 Sebastian Ullrich, Jakob T. Valvoda, Torsten Kuhlen: Utilizing optical sensors from mice for new input devices

2006-16 Rafael Ballagas, Jan Borchers: Selexels: a Conceptual Framework for Pointing Devices with Low Expressiveness

2006-17 Eric Lee, Henning Kiel, Jan Borchers: Scrolling Through Time: Improving Interfaces for Searching and Navigating Continuous Audio Timelines

2007-01 * Fachgruppe Informatik: Jahresbericht 2006

2007-02 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl: SAT Solving for Termination Analysis with Polynomial Interpretations

2007-03 Jürgen Giesl, René Thiemann, Stephan Swiderski, and Peter Schneider-Kamp: Proving Termination by Bounded Increase

2007-04 Jan Buchholz, Eric Lee, Jonathan Klein, and Jan Borchers: coJIVE: A System to Support Collaborative Jazz Improvisation

2007-05 Uwe Naumann: On Optimal DAG Reversal

2007-06 Joost-Pieter Katoen, Thomas Noll, and Stefan Rieger: Verifying Concurrent List-Manipulating Programs by LTL Model Checking

2007-07 Alexander Nyßen, Horst Lichter: MeDUSA - MethoD for UML2-based Design of Embedded Software Applications

2007-08 Falk Salewski and Stefan Kowalewski: Achieving Highly Reliable Embedded Software: An empirical evaluation of different approaches

2007-09 Tina Kraußer, Heiko Mantel, and Henning Sudbrock: A Probabilistic Justification of the Combining Calculus under the Uniform Scheduler Assumption

2007-10 Martin Neuhäußer, Joost-Pieter Katoen: Bisimulation and Logical Preservation for Continuous-Time Markov Decision Processes

2007-11 Klaus Wehrle (editor): 6. Fachgespräch Sensornetzwerke

2007-12 Uwe Naumann: An L-Attributed Grammar for Adjoint Code

2007-13 Uwe Naumann, Michael Maier, Jan Riehme, and Bruce Christianson: Second-Order Adjoints by Source Code Manipulation of Numerical Programs

2007-14 Jean Utke, Uwe Naumann, Mike Fagan, Nathan Tallent, Michelle Strout, Patrick Heimbach, Chris Hill, and Carl Wunsch: OpenAD/F: A Modular, Open-Source Tool for Automatic Differentiation of Fortran Codes

2007-15 Volker Stolz: Temporal assertions for sequential and concurrent programs

2007-16 Sadeq Ali Makram, Mesut Güneç, Martin Wenig, Alexander Zimmermann: Adaptive Channel Assignment to Support QoS and Load Balancing for Wireless Mesh Networks

2007-17 René Thiemann: The DP Framework for Proving Termination of Term Rewriting

2007-18 Uwe Naumann: Call Tree Reversal is NP-Complete

2007-19 Jan Riehme, Andrea Walther, Jörg Stiller, Uwe Naumann: Adjoints for Time-Dependent Optimal Control

2007-20 Joost-Pieter Katoen, Daniel Klink, Martin Leucker, and Verena Wolf: Three-Valued Abstraction for Probabilistic Systems

2007-21 Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre: Compositional Modeling and Minimization of Time-Inhomogeneous Markov Chains

2007-22 Heiner Ackermann, Paul W. Goldberg, Vahab S. Mirrokni, Heiko Röglin, and Berthold Vöcking: Uncoordinated Two-Sided Markets

2008-01 * Fachgruppe Informatik: Jahresbericht 2007

2008-02 Henrik Bohnenkamp, Marielle Stoelinga: Quantitative Testing

2008-03 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René :Thiemann, Harald Zankl: Maximal Termination

2008-04 Uwe Naumann, Jan Riehme: Sensitivity Analysis in Sisyphe with the AD-Enabled NAGWare Fortran Compiler

2008-05 Frank G. Radmacher: An Automata Theoretic Approach to the Theory of Rational Tree Relations

2008-06 Uwe Naumann, Laurent Hascoet, Chris Hill, Paul Hovland, Jan Riehme, Jean Utke: A Framework for Proving Correctness of Adjoint Message Passing Programs

2008-08 George B. Mertzios, Stavros D. Nikolopoulos: The λ-cluster Problem on Parameterized Interval Graphs

2008-09 George B. Mertzios, Walter Unger: An optimal algorithm for the k-fixed-endpoint path cover on proper interval graphs

* These reports are only available as a printed version.
Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.