Measuring Indifference: Unit Interval Vertex Deletion

René van Bevern*, Christian Komusiewicz**, Hannes Moser*, and Rolf Niedermeier

Institut für Informatik, Friedrich-Schiller-Universität Jena,
Ernst-Abbe-Platz 2, D-07743 Jena, Germany
{rene.bevern,c.komus,hannes.moser,rolf.niedermeier}@uni-jena.de

Abstract. Making a graph unit interval by a minimum number of vertex deletions is NP-hard. The problem is motivated by applications in seriation and measuring indifference between data items. We present a fixed-parameter algorithm based on the iterative compression technique that finds in $O((14k+14)^{k+1}kn^6)$ time a set of k vertices whose deletion from an n-vertex graph makes it unit interval. Additionally, we show that making a graph chordal by at most k vertex deletions is NP-complete even on {claw,net,tent}-free graphs.

1 Introduction

Being indifferent between two objects means to prefer neither of them. The indifference relation defines an undirected graph with an edge between two vertices if and only if they are judged indifferent. In this paper, we study measuring indifference in the context of seriation. The specific task is to put objects in a serial order, respecting the given indifference relation as much as possible.

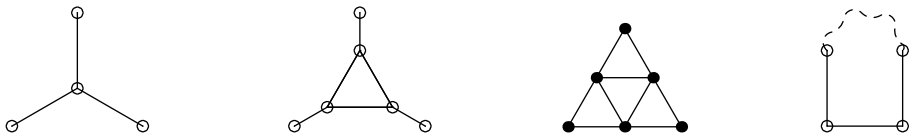
Indifference corresponds to "closeness" between data items [14, 1]. Accordingly, an undirected graph $G = (V, E)$ whose vertices represent data items is called an *indifference graph* if there exists a function $r\colon V \to \mathbb{R}$ such that for all $u, v \in V$

$$\{u, v\} \in E \Leftrightarrow |r(u) - r(v)| \leq \delta,$$

where δ is a positive number (the "threshold") measuring closeness. The function r induces a serial order. Informally, the above equivalence expresses that we distinguish between u and v only if, according to r, there is sufficiently high difference between them. Empirical indifference judgments (with correspondingly defined undirected graphs) usually do not permit such an assignment r satisfying the above equivalence. One possibility, however, is that "almost all" data items induce an indifference graph. In other words, given a graph based on empirical indifference judgments (which contain some "errors"), the task then is to spot as

* Supported by the DFG, project AREG, NI 369/9.
** Supported by a PhD fellowship of the Carl-Zeiss-Stiftung and the DFG, project PABI, NI 369/7.

Published in the *Proceedings of the 36th International Workshop on Graph Theoretic Concepts in Computer Science (WG'10)*, Zarós, Crete, Greece, June 2010. Volume 6410 in Lecture Notes in Computer Science, pp. 232–243 © Springer



(a) Claw (b) Net (c) Tent (d) Hole

Fig. 1: The (infinitely many) forbidden induced subgraphs for unit interval graphs. Holes are induced cycles of length at least four.

few "outlier vertices" as possible such that after the removal of these vertices the graph becomes an indifference graph. Thus, the minimum number of vertex deletions measures how close "empirical indifference data" is to mathematically defined indifference. Since indifference graphs are precisely the unit interval graphs [18, 19][1], we arrive at the central problem in this work:

Unit Interval Vertex Deletion
Input: An undirected graph $G = (V, E)$.
Question: Is there a set $S \subseteq V$ with $|S| \leq k$ and $G[V \setminus S]$ being unit interval?

A graph is unit interval if and only if it contains no induced claw, net, tent, or hole (an induced cycle of length at least four) [4]. These infinitely many forbidden induced subgraphs are illustrated in Figure 1. A general result on vertex deletion problems implies that Unit Interval Vertex Deletion is NP-complete [13].

Related Work. Roberts [18, 19] discusses indifference and seriation and explains applications in fields such as archaeology and developmental psychology. Seriation by transforming graphs into indifference graphs belongs to the field of "seriation in the presence of errors" and can be understood as a variant of fitting Robinson structures to distances [6, 5, 2]. More specifically, our setting is related to the special case where distances are specified by symmetric 0/1-matrices; here, the Robinson property becomes the consecutive-ones property [19, 5]. Thus, the Unit Interval Vertex Deletion problem is equivalent to making a symmetric 0/1-matrix fulfill the consecutive-ones property by simultaneous[2] column and row deletions. We remark that making a 0/1-matrix fulfill the consecutive-ones property by means of non-simultaneous column or row deletions has recently been studied in terms of approximability and fixed-parameter tractability [7].

Our work is closely related to a result by Marx [15], who shows that Chordal Vertex Deletion (asking whether a graph can be made chordal by k vertex deletions) is fixed-parameter tractable parameterized by k. One can observe that the result implies fixed-parameter tractability for Unit Interval Vertex Deletion. However, the running time of the Chordal Vertex Deletion algorithm [15] is not specified and it relies on solving Chordal Vertex Deletion

[1] Unit interval graphs are also equivalent to proper interval graphs [18].
[2] For an $i \in \mathbb{N}$, column i is deleted from the matrix if and only if row i is deleted.

on tree decompositions of worst-case-width $\Omega(k^4)$. This renders the algorithm unimplementable. Subsequently to our work, Villanger [21] presented a search-tree based algorithm for Unit Interval Vertex Deletion, improving the running time to $O(6^k kn^6)$.

Our Results. We present a fixed-parameter algorithm for Unit Interval Vertex Deletion running in $O((14k+14)^{k+1} \cdot kn^6)$ time, where k denotes the number of allowed vertex deletions and n is the number of graph vertices. Like Marx [15], we employ the iterative compression technique by Reed et al. [17, 12]. However, we do not employ bounded-treewidth techniques and circumvent huge hidden constants. Before that, we show that Unit Interval Vertex Deletion remains NP-hard when restricted to {claw, net, tent}-free graphs, where it is equivalent to Chordal Vertex Deletion. Due to lack of space, some proofs are omitted [2].

Preliminaries. We only consider simple *undirected* graphs $G = (V, E)$, where $V(G) \coloneqq V$ is the set of vertices and $E(G) \coloneqq E$ is the set of edges. Throughout this work, let $n \coloneqq |V|$ and $m \coloneqq |E|$. The *neighborhood* $N(v)$ of a vertex $v \in V$ is the set of vertices adjacent to v. A *clique* is a graph in which every two distinct vertices are adjacent. For a set $V' \subseteq V$, the *induced subgraph* $G[V']$ is the graph with vertex set V' and edge set $\{\{v, w\} \in E \mid v, w \in V'\}$. We use $G - V'$ as an abbreviation for $G[V \setminus V']$. A *path* P from v_i to v_ℓ is a sequence $(v_1, v_2, \ldots, v_\ell) \in V^\ell$ with $\{v_i, v_{i+1}\} \in E$ for $i \in \{1, \ldots, \ell-1\}$; it *visits* the vertices $v_1, \ldots, v_\ell$. If $i \neq j$ implies $v_i \neq v_j$, then P is *simple*. If $\{v_i, v_j\} \notin E$ for $|i-j| > 1$, then P is *induced*. An *(induced) cycle* is an (induced) path with $\{v_1, v_\ell\} \in E$, called (induced) C_ℓ. Two vertices $v, w \in V$ are *connected* in G if there is a path from v to w in G. A *vertex-cut* between v and w in G is a set $C \subseteq V$ such that v and w are not connected in $G - C$. The graph G is *F-free* if G does not contain an induced subgraph isomorphic to the graph F. A *segment* of a total order $\preceq$ on V is a set $[u, w] \coloneqq \{v \in V \mid u \preceq v \preceq w\}$.

2 NP-Hardness on a Restricted Graph Class

We show that Unit Interval Vertex Deletion is NP-complete even on {claw, net, tent}-free graphs, where it is equivalent to Chordal Vertex Deletion.

Theorem 1. Chordal Vertex Deletion *on {claw, net, tent}-free graphs is NP-complete.*

Proof. We only show NP-hardness and employ a reduction from the NP-complete Vertex Cover on triangle-free graphs [10]. First, we describe the reduction, then we prove its correctness. Let (G, k) be a Vertex Cover instance, where G is triangle-free and we ask whether there is a set of k vertices whose deletion makes G edgeless. We construct an instance (G', k) for Chordal Vertex Deletion as follows: let $G' \coloneqq \overline{G}$, where the complement graph $\overline{G}$ of G has the same vertices as G and $\overline{G}$ has an edge $\{v, w\}$ if and only if G has not. Add to G' two disjoint cliques A and B, each containing $k+1$ vertices, and make every

vertex in A and every vertex in B adjacent to every vertex in $V(G)$. Because claw, net, and tent each contain a size-three independent set, the constructed graph G' is {claw, net, tent}-free: since G is triangle-free, $\overline{G}$ does not contain a size-three independent set. Moreover, there is no size-three independent set in G', since the vertices in the cliques A and B are adjacent to every vertex in $V(G)$. It remains to show that (G, k) is a yes-instance for Vertex Cover if and only if (G', k) is a yes-instance for Chordal Vertex Deletion.

Let S be a vertex cover of size at most k for G. Since S is a vertex cover, $G - S$ is an edgeless graph. As a consequence, the complement of $G - S$ is a clique C and thus $G' - S$ contains three cliques A, B, and C, where every vertex of A and B is adjacent to every vertex in C by construction of G'. The graph $G' - S$ is obviously chordal and S is a chordal vertex deletion set of size at most k for G'.

Let S be a chordal vertex deletion set for G' with $|S| \leq k$. Assume that $S \cap V(G)$ is not a vertex cover for G. Then, there is an edge $\{u, v\}$ in $G - S$, and, therefore, there is no edge $\{u, v\}$ in G' by construction of G'. Because A and B each contain $k+1$ vertices and $|S| \leq k$, there are vertices $a \in A \setminus S$ and $b \in B \setminus S$. The vertex set $\{a, u, v, b\}$ induces a hole in $G' - S$, a contradiction to S being a chordal vertex deletion set for G'. Hence, $S \cap V(G)$ is a vertex cover for G. □

3 An Outline of the Algorithm

Our algorithm employs the iterative compression technique by Reed et al. [17, 12]. The rough idea of this technique is to iteratively build up the input graph by adding vertices one by one and to compute in each iteration an optimal solution for the current subgraph, using the solution computed for the previous subgraph. More precisely, given an arbitrary order of the vertices from 1 to n, we start with the empty graph and an empty solution $S_0 := \emptyset$. The task of iteration i is to compute a solution for the graph $G[\{v_1, \ldots, v_i\}]$. Assume that the previously computed set S_{i-1} is a solution of size at most k for $G[\{v_1, \ldots, v_{i-1}\}]$. Then $S_{i-1} \cup \{v_i\}$ is a solution of size at most $k+1$ for the graph $G[\{v_1, \ldots, v_i\}]$. We apply a *compression routine* that either computes a size-k solution S_i using $S_{i-1} \cup \{v_i\}$ or proves that no such solution exists. The pseudo-code of this main loop is given in Algorithm 1.

Algorithm 1: Iterative Compression

Input: A graph G and its vertices $v_1, \ldots, v_n$ in an arbitrary order, $k \in \mathbb{N}$.
Output: A unit interval vertex deletion set S for G with $|S| \leq k$ or "no".

```
S_0 ← ∅;
for i := 1 to n do
    S_i ← compress(G[{v_1, ..., v_i}], S_{i-1} ∪ {v_i}, k);
    if S_i = "no" then return "no"
return S_n
```

The central part of the algorithm is the routine *compress* described below. Given an input graph G, a natural number k, and a unit interval vertex deletion set S' with $|S'| \leq k+1$, the routine can return S' unchanged if $|S'| \leq k$. Thus, we assume that $|S'| = k+1$. We now try all possible 2^{k+1} partitions of S' into two sets X and Y, where Y is a subset of the new solution S and $X \cap S = \emptyset$. For each partition, we delete the vertex set Y from G (since the vertices of Y are assumed to belong to the new solution). Then, the remaining task is to find a unit interval vertex deletion set disjoint from X and smaller than X. The crucial observation is that deleting X from $G - Y$ results in a unit interval graph. We say that X is a *unit interval vertex deletion set* for $G - Y$. Summarizing, we arrive at:

Disjoint Unit Interval Vertex Deletion
Input: A graph G and a unit interval vertex deletion set X for G.
Output: A unit interval vertex deletion set S with $|S| < |X|$ and $S \cap X = \emptyset$, or "no" if no such set exists.

Disjoint Unit Interval Vertex Deletion is NP-hard [8]. The advantage of working with Disjoint Unit Interval Vertex Deletion is that we can exploit $G - X$ being a unit interval graph. In the next section, we prove:

Theorem 2. Disjoint Unit Interval Vertex Deletion *can be solved in* $O((14|X|-1)^{|X|-1} \cdot |X|n^5)$ *time.*

Exploiting this in the routine *compress* of Algorithm 1 leads to the main theorem of this work. The running time follows from the fact that *compress* is invoked $O(n)$ times and that each invocation solves Disjoint Unit Interval Vertex Deletion for all partitions of the solution from the previous iteration.

Theorem 3. Unit Interval Vertex Deletion *can be solved in* $O((14k+14)^{k+1} \cdot kn^6)$ *time.*

4 Finding Disjoint Unit Interval Vertex Deletion Sets

For Disjoint Unit Interval Vertex Deletion, given a unit interval vertex deletion set X for G, we search for a unit interval vertex deletion set S for G with $|S| < |X|$ and $S \cap X = \emptyset$. Roughly, the algorithm works as follows: first, enumerate *all* minimal vertex sets of size at most $|X|-1$ whose deletion transforms G into a {claw,net,tent,C_4,C_5,C_6}-free graph, henceforth called *almost unit interval graph*. For each of these graphs, it remains to find a minimum-cardinality vertex set S' whose removal destroys all holes of length greater than six to make the graph unit interval. We call such a set *optimal*. If $|S'|$ and the number of vertex deletions needed to transform a graph into an almost unit interval graph add up to at most $|X|-1$, then we have found a solution. Since we try *all* minimal vertex sets of size at most $|X|-1$ whose deletion transforms a graph into an almost unit interval graph, we always find a size-$(|X|-1)$ unit interval vertex deletion set if it exists.

To destroy all holes in an almost unit interval graph G, we show that each hole can be destroyed by deleting any of at most $14|X|-1$ vertex sets, of which

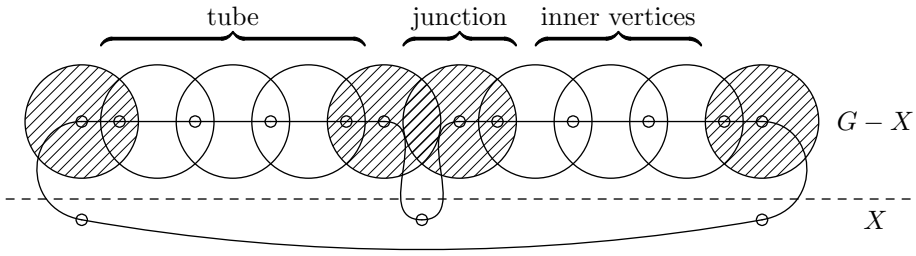


Fig. 2: A hole visiting the maximal cliques of $G-X$ (indicated by circles). Hatched circles show junctions. The vertices of $G - X$ are shown from left to right in a bicompatible elimination order. A maximal set of consecutive white cliques forms a tube. Tubes may contain vertices of junctions. For the right tube, we indicate the inner vertices.

at least one vertex set can be assumed to be in an optimal unit interval vertex deletion set. This allows us to use a bounded search tree algorithm that, for each hole H in G, branches into $14|X| - 1$ possibilities to destroy H. Because at most $|X| - 1$ vertices may be deleted from G to transform it into a unit interval graph, the height of the corresponding search tree is bounded by $|X| - 1$. To find these $14|X| - 1$ vertex sets for each hole in G, we exploit that $G - X$ is a unit interval graph and thus allows for a linear-time computable *bicompatible elimination order* of its vertices [16]:

Definition 1. *Let $G = (V, E)$ be a graph. A total order $\preceq$ on V is a* bicompatible elimination order *for G if for each vertex $v \in V$, the sets $\{w \in N(v) \mid w \preceq v\}$ and $\{w \in N(v) \mid v \preceq w\}$ induce cliques in G.*

Without loss of generality, we assume that the vertices of a connected component of G form a segment of $\preceq$. We will see in Proposition 2 that, with respect to a bicompatible elimination order $\preceq$, $G - X$ forms a sequence of maximal cliques such that the vertices of each maximal clique are a segment of $\preceq$. Figure 2 illustrates this together with the following classification of the maximal cliques of $G-X$: a *junction* is a maximal clique in $G-X$ containing neighbors of vertices in X; a *tube* is a maximal set of maximal cliques of $G - X$ that are not junctions and whose vertices form a segment of $\preceq$. We say that a vertex is *contained in a tube T* if it is contained in a maximal clique of T. A hole *visits* a junction (or tube) if it contains a vertex of a junction (or tube). Vertices of a tube that are not in junctions are *inner vertices*.

Now, assume that there is a hole H in an almost unit interval graph G as illustrated in Figure 2. We show $14|X| - 1$ possibilities to destroy H of which one is optimal. Each vertex of H in $G-X$ is contained in a junction or tube (or both). First, we show that H contains at most $12|X|$ vertices in junctions and that H contains inner vertices of at most $2|X| - 1$ tubes. Additionally, we show that there is an optimal unit interval vertex deletion set that contains a vertex of H in junctions or a polynomial-time computable vertex subset of one of the $2|X| - 1$ tubes whose inner vertices are visited by H. Then, we solve Unit Interval Vertex Deletion by repeatedly searching for a hole H in G in polynomial

time and branching into the following $14|X|-1$ possibilities to destroy H: delete one of the $12|X|$ vertices of H in junctions, or delete an optimal, polynomial-time determinable vertex subset of one of the $2|X|-1$ tubes whose inner vertices are visited by H. Using this branching, the overall search tree size is $O((14|X|-1)^{|X|-1})$, which results in the running time of Theorem 2. In the following, we show in detail the $14|X|-1$ possibilities to destroy H of which one is optimal.

Bounding the Number of Vertices in Junctions. We now prove the following:

Lemma 1. *Let X be a unit interval vertex deletion set for an almost unit interval graph G. A hole in G contains at most $12|X|$ vertices from junctions in $G-X$.*

First, observe that a hole contains at most two vertices of a clique. We now exploit that G is an almost unit interval graph. In the following, we say that a vertex set can be covered by two cliques if it is the union of two vertex sets that induce cliques.

Proposition 1. *If a connected almost unit interval graph G contains a hole, then the neighborhood of each vertex in G can be covered by two cliques.*

Proof. If G contains a hole, then it must contain a hole with more than six vertices, since G is {claw, net, tent, C_4, C_5, C_6}-free. Thus, G contains an independent set of size three. We now exploit a result due to Fouquet [9]:

> In a connected claw-free graph containing an independent set of size three, every vertex v satisfies exactly one of the following properties:
> (i) $N(v)$ can be covered by two cliques or
> (ii) $N(v)$ contains an induced C_5.

Because G contains no induced C_5, the proposition follows immediately. □

From Proposition 1, one can conclude that if an almost unit interval graph G contains a hole, then the neighborhood in $V \setminus X$ of a unit interval vertex deletion set X can be covered by $2|X|$ cliques.

We now prove that the maximal cliques of a unit interval graph form segments of a bicompatible elimination order and that vertices on induced paths occur in the same (or reverse) order as in a bicompatible elimination order.

Proposition 2. *Let $v_1 \preceq v_2 \preceq \ldots \preceq v_n$ be a bicompatible elimination order for a connected unit interval graph G.*
(1) If there is an induced path $P = (v_i, \ldots, v_k)$ with $v_i \preceq v_k$, then each vertex v_j on P satisfies $v_i \preceq v_j \preceq v_k$.
(2) If $v_i \preceq v_k$ and there is an edge between v_i and v_k, then the segment $[v_i, v_k]$ induces a clique in G. In particular, maximal cliques of G form segments.

Proof. We prove the two statements independently. To show (1), for the purpose of contradiction, assume that there is an induced path $P = (v_i, \ldots, v_j, \ldots, v_k)$ with $v_j \preceq v_i \preceq v_k$ (the case $v_i \preceq v_k \preceq v_j$ can be proven analogously) such that v_j is the minimum vertex with respect to $\preceq$ that appears between v_i and v_k on P;

(a) The vertices $v_j \preceq v_{j'} \preceq v_{j''}$ form the induced path $(v_{j'}, v_j, v_{j''})$, in wrong order.

(b) A maximal clique induced by the non-consecutive vertices v_1, v_2, v_4, and v_5.

Fig. 3: The vertex orderings from left to right are not bicompatible elimination orders, as they violate Proposition 2.

this arrangement is illustrated in Figure 3a. Because there are induced subpaths of P from v_j to both v_i and v_k, the vertex v_j has two distinct neighbors $v_{j'}$ and $v_{j''}$ on P. Because v_j is the minimum vertex with respect to $\preceq$ that appears between v_i and v_k on P, it holds that $v_j \preceq v_{j'}$ and $v_j \preceq v_{j''}$. The vertices $v_{j'}$ and $v_{j''}$ are adjacent by Definition 1, because both are succeeding neighbors of v_j. This contradicts P being an *induced* path.

Before showing (2), we show that there is an edge between a vertex v_i and its direct successor v_{i+1} for $i \in \{1, \ldots, n-1\}$. Recall that G is connected by assumption. This implies that there is a shortest (and hence, induced) path from v_i to v_{i+1}. It follows from (1) that this path can neither contain a predecessor of v_i nor a successor of v_{i+1}. Because v_{i+1} directly succeeds v_i in $\preceq$, v_i and v_{i+1} are the only vertices on the shortest path from v_i to v_{i+1}, implying that they are adjacent.

We now show (2). Let $v_i \preceq v_k$ and assume that there is an edge between v_i and v_k. We have shown that v_i is also adjacent to its direct successor v_{i+1}. Because v_{i+1} and v_k are succeeding neighbors of v_i, the vertices v_i, v_{i+1}, and v_k form a clique by Definition 1. Inductively, it follows that all vertices v_j with $v_i \preceq v_j \preceq v_k$ are adjacent to v_k. Because all vertices v_j with $v_i \preceq v_j \preceq v_k$ are preceding neighbors of v_k in the bicompatible elimination order $\preceq$, these vertices must form a clique together with v_k. A maximal clique in G forms a segment because, with respect to $\preceq$, it contains an edge from its minimum vertex to its maximum vertex. □

To prove Lemma 1, we finally need the following definition (illustrated in Figure 4).

Definition 2. *Let G be a unit interval graph with a bicompatible elimination order $\preceq$ and let C be a clique of G. We define $\mathcal{S}(C)$ to be the set of vertices of all maximal cliques in G that contain vertices of C.*

Let c_{min} (and c_{max}) denote the minimum (or maximum, respectively) elements of $\mathcal{S}(C)$ with respect to $\preceq$. We define $\mathcal{S}_{min}(C)$ (and $\mathcal{S}_{max}(C)$) to be the vertex set of the (uniquely determined) maximal clique in G that contains c_{min} (or c_{max}, respectively) and some vertex from C.

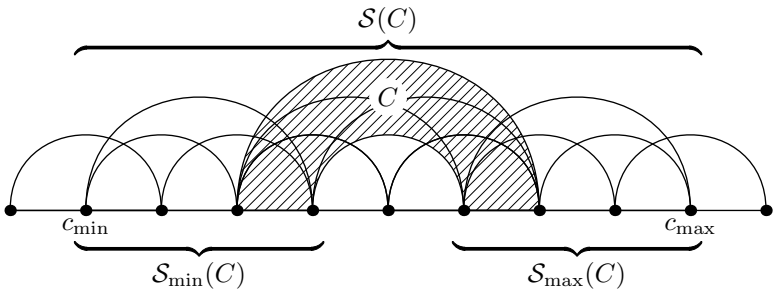


Fig. 4: Illustration for Definition 2. The vertices are shown from left to right in a bicompatible elimination order. The hatched clique is C.

Using Proposition 2, one can show that $\mathcal{S}(C)$ is the union of $\mathcal{S}_{\min}(C)$, $\mathcal{S}_{\max}(C)$, and the vertex set of any maximal clique containing C. Therefore, $\mathcal{S}(C) = [c_{\min}, c_{\max}]$. We have now collected the necessary observations to prove Lemma 1.

Proof (Proof of Lemma 1). Let X be a unit interval vertex deletion set for an almost unit interval graph G. Assume that G contains a hole. By Proposition 1, the neighborhood of X in $V \setminus X$ can be covered by a set $\mathcal{C}$ of at most $2|X|$ cliques. By Definition 2, the vertices of all junctions containing vertices of a clique $C \in \mathcal{C}$ are in $\mathcal{S}(C)$. We show that a hole contains at most six vertices from $\mathcal{S}(C)$ and, thus, in all junctions containing vertices of C. A hole contains at most two vertices in the vertex set C' of any maximal clique containing C, at most two vertices of the clique induced by $\mathcal{S}_{\min}(C)$, and at most two vertices in the clique induced by $\mathcal{S}_{\max}(C)$. Because $\mathcal{S}(C) = \mathcal{S}_{\min}(C) \cup \mathcal{S}_{\max}(C) \cup C'$, a hole contains at most six vertices from $\mathcal{S}(C)$. Finally, since $|\mathcal{C}| \leq 2|X|$ and for each $C \in \mathcal{C}$, a hole contains at most six vertices in junctions containing vertices of C, it follows that a hole contains at most $12|X|$ vertices in junctions of $G - X$. □

Finding Optimal Solutions in Tubes. We now show how to find optimal solutions in tubes. To this end, one can prove that a hole H visits inner vertices of at most $2|X| - 1$ tubes in $G - X$. Moreover, one can show that there is an optimal unit interval vertex deletion set containing at least one vertex of H in junctions or a polynomial-time computable vertex subset of one of the $2|X| - 1$ tubes.

Lemma 2. *Let X be a unit interval vertex deletion set for an almost unit interval graph G. A hole in G contains inner vertices of at most $2|X| - 1$ tubes with respect to a bicompatible elimination order $\preceq$ for $G - X$.*

To state the second result (Lemma 3), we need the following concepts.

Definition 3. *Let X be a unit interval vertex deletion set for a graph G and let T be a tube in $G - X$ with respect to a bicompatible elimination order $\preceq$ such that T contains inner vertices visited by a hole H.*

(1) *For two vertices v_i and v_k of H, we call (v_i, v_k) the* T-boundary *of H if, with respect to $\preceq$, v_i is in H the preceding neighbor of H's minimum inner vertex in T and if v_k is in H the succeeding neighbor of H's maximum inner vertex of T.*
(2) *For the T-boundary (v_i, v_k) of a hole H, we call a (minimum-cardinality) vertex-cut between v_i and v_k in $G - X$ a* (minimum) H-T-cut.

Lemma 3. *Let X be a unit interval vertex deletion set for a graph G. Let $\mathcal{T}$ be the set of tubes in $G - X$ with respect to a bicompatible elimination order $\preceq$ that contain inner vertices visited by a hole H.*

If a unit interval vertex deletion set S for G with $S \cap X = \emptyset$ does not contain vertices of H in junctions of $G - X$, then there is a unit interval vertex deletion set S' with $|S'| \leq |S|$ and a tube $T \in \mathcal{T}$ for which S' contains a minimum H-T-cut.

Lemma 3 can be shown exploiting Proposition 2(1), which implies that if a hole enters a tube at one side, it must leave the tube at the opposite side. Using this fact, one can show the following two claims, which together imply Lemma 3.

Claim. A unit interval vertex deletion set S for G with $S \cap X = \emptyset$ contains vertices of H in junctions or contains an H-T-cut for some tube $T \in \mathcal{T}$.

Claim. For a vertex set S containing a H-T-cut for a tube $T \in \mathcal{T}$, no hole in $G - S$ contains vertices of the segment $[v_i, v_k]$, where (v_i, v_k) is the T-boundary of H.

The Algorithm

Combining Lemmas 1, 2 and 3, we finally present the algorithm for Disjoint Unit Interval Vertex Deletion, thus proving Theorem 2. The algorithm employs the following branching rule:

Branching Rule 1. If $G - S$ contains a forbidden induced subgraph induced by a vertex set F with $|F| \leq 6$, then branch into all possibilities of adding a vertex $v \in F \setminus X$ to S.

Proof (Proof of Theorem 2). Given a graph G and a unit interval vertex deletion set X for G, we search for a unit interval vertex deletion set S for G with $|S| < |X|$ and $S \cap X = \emptyset$. We start with $S := \emptyset$ and apply Branching Rule 1 as long as $|S| < |X|$ to destroy forbidden induced subgraphs with at most six vertices. Because each such forbidden induced subgraph contains one vertex in the unit interval vertex deletion set X, we find such a forbidden induced subgraph in $O(|X|n^5)$ time and branch into at most five cases to add one of its vertices to S.

If $|S| \geq |X|$ and Branching Rule 1 is still applicable, return "no" because S is not extensible to a unit interval vertex deletion set for G that is smaller than X and disjoint from X. Otherwise, proceed as follows: compute a bicompatible elimination order $\preceq$ for $G - (S \cup X)$. This works in linear time [16] because $G - (S \cup X)$ is a unit interval graph. From this bicompatible elimination order $\preceq$,

a set $\mathcal{C}$ of all maximal cliques of $G-(S\cup X)$ can easily be computed in $O(n^2)$ time by finding for each vertex v in $G-(S\cup X)$ its last neighbor with respect to $\preceq$.

Now, we find junctions and tubes. For each clique $C \in \mathcal{C}$, check whether C has neighbors in X. If this is the case, which can be checked in $O(kn^2)$ time for all $C \in \mathcal{C}$, then C is a junction. To find tubes, sort the set $\mathcal{C}$ such that C_1 occurs before C_2 if, in $\preceq$, the minimum vertex of C_1 occurs before the minimum vertex of C_2. Because a unit interval graph has at most n maximal cliques, this is possible in $O(n \log n)$ time. From the sorted set $\mathcal{C}$, compute a set $\mathcal{T}$ of all tubes in $G-(S\cup X)$ in $O(n)$ time: repeatedly find the first clique C in $\mathcal{C}$ that is not a junction and not yet part of a tube and add C and all succeeding cliques in $\mathcal{C}$ to a new tube T until a junction is encountered.

Next, as long as $|S| < |X|$, repeatedly find a hole H in $G-S$ and add at least one vertex of H to S as follows: because $G-S$ is an almost unit interval graph and $G-(S\cup X)$ is a unit interval graph, recursively branch into at most $12|X|$ possibilities to choose a vertex of H from a junction for inclusion in S (Lemma 1) and into at most $2|X|-1$ possibilities to choose tube $T \in \mathcal{T}$ for which a H-T-cut shall be included in S (Lemma 2, Lemma 3).

In each search tree node, we branch into at most $14|X|-1$ cases (at most five cases for Branching Rule 1 and at most $14|X|-1$ for a hole in $G-S$). In each case, at least one vertex is added to S. As a result, the corresponding search tree has depth at most $|X|-1$ and thus at most $(14|X|-1)^{|X|-1}$ nodes.

To analyze the running time for processing each node, it remains to analyze the running time for finding holes and minimum H-T-cuts. A hole in $G-S$ can be found in $O(|X|(n+m))$ time by breadth-first search starting at each vertex in X. A minimum H-T-cut is computable in $O(\sqrt{n}m)$ time [20, Theorem 9.8]. □

5 Conclusion

It remains open to study the existence of a polynomial-size problem kernel [3, 11] for Unit Interval Vertex Deletion. Another task for future study is to search for polynomial-time algorithms with low-degree polynomials in case of constant k. Villanger's algorithm [21] runs in $O(6^k kn^6)$ time and thus also is far from this goal. We remark that already Marx [15] asked for the study of the parameterized complexity of the related Interval Vertex Deletion problem, remaining a challenge for future research. In general, interval graphs do not allow for bicompatible elimination orders; as our algorithm heavily relies on them, it is not straightforward to extend it to Interval Vertex Deletion.

Acknowledgments. We thank anonymous referees for their constructive feedback.

References

[1] Aleskerov F, Bouyssou D, Monjardet B (2007) Utility Maximization, Choice and Preference, Studies in Economic Theory, vol 16. Springer-Verlag

[2] van Bevern R (2010) The Computational Hardness and Tractability of Restricted Seriation Problems on Inaccurate Data. Diplomarbeit, Institut für Informatik, Friedrich-Schiller-Universität, Jena, Germany
[3] Bodlaender HL (2009) Kernelization: New upper and lower bound techniques. In: Proc. 4th IWPEC, Springer, LNCS, vol 5917, pp 17–37
[4] Brandstädt A, Le VB, Spinrad JP (1999) Graph classes: a survey. SIAM, Philadelphia, PA, USA
[5] Chepoi V, Seston M (2009) Seriation in the presence of errors: A factor 16 approximation algorithm for l_∞-fitting Robinson structures to distances. Algorithmica Available electronically.
[6] Chepoi V, Fichet B, Seston M (2010) Seriation in the presence of errors: NP-hardness of l_∞-fitting Robinson structures to dissimilarity matrices. J Classification 26(3):279–296
[7] Dom M, Guo J, Niedermeier R (2010) Approximation and fixed-parameter algorithms for consecutive ones submatrix problems. J Comput System Sci 76(3–4):204–221
[8] Fellows MR, Guo J, Moser H, Niedermeier R (2009) A complexity dichotomy for finding disjoint solutions of vertex deletion problems. In: Proc. 34th MFCS, Springer, LNCS, vol 5734, pp 319–330
[9] Fouquet JL (1993) A strengthening of Ben Rebea's lemma. J Combin Theory Ser B 59(1):35–40
[10] Garey MR, Johnson DS, Stockmeyer LJ (1976) Some simplified NP-complete graph problems. Theor Comp Sci 1(3):237–267
[11] Guo J, Niedermeier R (2007) Invitation to data reduction and problem kernelization. ACM SIGACT News 38(1):31–45
[12] Guo J, Moser H, Niedermeier R (2009) Iterative compression for exactly solving NP-hard minimization problems. In: Algorithmics of Large and Complex Networks, LNCS, vol 5515, Springer, pp 65–80
[13] Lewis JM, Yannakakis M (1980) The node-deletion problem for hereditary properties is NP-complete. J Comput System Sci 20(2):219–230
[14] Luce RD (1956) Semiorders and a theory of utility discrimination. Econometrica 24:178–191
[15] Marx D (2010) Chordal deletion is fixed-parameter tractable. Algorithmica 57(4):747–768
[16] Panda BS, Das SK (2003) A linear time recognition algorithm for proper interval graphs. Inf Process Lett 87(3):153–161
[17] Reed B, Smith K, Vetta A (2004) Finding odd cycle transversals. Oper Res Lett 32(4):299–301
[18] Roberts FS (1969) Indifference graphs. In: Proof Techniques in Graph Theory, Academic Press, New York, pp 139–146
[19] Roberts FS (1979) Indifference and seriation. Annals of the New York Academy of Sciences 328:173–182
[20] Schrijver A (2003) Combinatorial Optimization: Polyhedra and Efficiency, vol A. Springer
[21] Villanger Y (2010) Proper interval vertex deletion. In: Proc. 5th IPEC, Springer, LNCS