Restricted vertex multicut on permutation graphs

Charis Papadopoulos

Department of Mathematics, University of Ioannina
GR-45110, Ioannina, Greece

charis@cs.uoi.gr

Abstract

Given an undirected graph and pairs of terminals the Restricted Vertex Multicut problem asks for a minimum set of nonterminal vertices whose removal disconnects each pair of terminals. The problem is known to be NP-complete for trees and polynomial-time solvable for interval graphs. In this paper we give a polynomial-time algorithm for the problem on permutation graphs. Furthermore we show that the problem remains NP-complete on split graphs whereas it becomes polynomial-time solvable for the class of co-bipartite graphs.

1 Introduction

One of the well-studied problems that fall in the area of cut and separation problems is the Multicut problem introduced by Hu in [14]. Given a graph G and a list L of pairs of vertices that are called terminals, the objective for the Multicut problem is to disconnect each terminal pair of the predefined list by removing a minimum set of edges or vertices of G. Problems of this kind arise from areas concerning with the reliability and robustness of network communications [6]. The Multicut problem is NP-complete [7] and several algorithms that approximate a solution on general or restricted graph classes are known [5, 9, 16], while the parameterized version was proved to be fixed-parameter tractable only very recently [3, 17]. This problem includes as a special case the Multiway Cut problem where instead of the list L we are given a set of terminals that need to be pairwise separated from each other; see [7, 11, 15]. Depending on the multicut, that is, the set of vertices or edges whose deletion disconnects each terminal pair, there are three variations of the problem.

The history of multicut problems begins with the edge variation, known as the Edge Multicut, that allows only the removal of edges. If L contains at most two terminal pairs the Edge Multicut problem admits a polynomial-time algorithm [21] whereas for at least three terminal pairs it becomes NP-hard [7]. Furthermore Edge Multicut remains NP-hard even when the input graph is a star (tree of height 1) [10] and therefore excluding any possible polynomial solution on many interesting graph classes. Similar to the Edge Multicut is the Vertex Multicut problem in which one is only allowed to remove vertices of the input graph.

As it was introduced by Călinescu et al. in [5] the Vertex Multicut problem splits into two subproblems depending on whether one is allowed to remove terminal vertices.

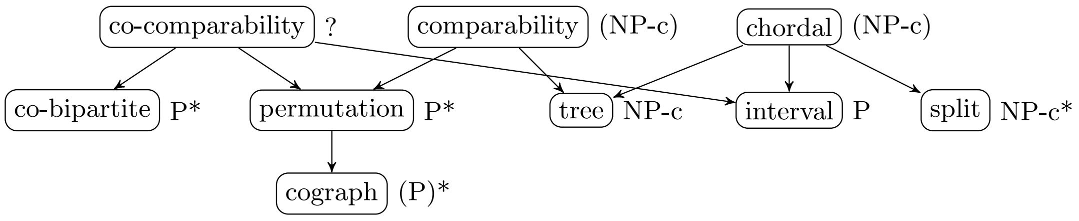

Figure 1: An inclusion relationship of the considered graph classes and the complexity of Restricted Vertex Multicut problem in each graph class. The arrow $\rightarrow$ represents the $\supset$ relation. NP-c means NP-complete, P means polynomial-time algorithm, the asterisk $*$ indicates that the result is obtained here, the pair of parenthesis ( ) means that the complexity is obtained from graph inclusion relationships, and ? stands for unknown complexity.

The Unrestricted Vertex Multicut problem refers to finding *any* minimum set of vertices of G whose removal separates each terminal pair of L whereas the Restricted Vertex Multicut refers to minimizing a set of *nonterminal* vertices for the same objective. Unrestricted Vertex Multicut is NP-hard on interval graphs [13], graphs with bounded treewidth [5], and planar graphs of bounded degree [5]. From the positive side there is a polynomial-time algorithm for the latter problem on trees [5]. Looking at the line graph of G (that is, the graph representing the adjacencies between the edges of G) an interesting reduction shows that the vertex variant is more general than the edge variant [5]. More precisely, consider an instance of Edge Multicut whose input is a graph G and a set of terminal pairs L, denoted by (G, L). Construct the line graph of G, denoted by $L(G)$ and construct the set of terminal pairs L' which contains for each pair (s, t) of L all pairs (e_i, f_i) such that e_i has s as endpoint and f_i has t as endpoint. Then it is known that a solution for the Edge Multicut on (G, L) is a solution for the Unrestricted Vertex Multicut on $(L(G), L')$ and vice versa [5]. Observe that the line graph of a star graph is a complete graph and, therefore, Unrestricted Vertex Multicut on complete graphs is NP-complete. Thus for many interesting and even constraint graph classes (e.g., cographs, split graphs, and co-bipartite graphs) the Unrestricted Vertex Multicut problem is already NP-hard. Here we focus on the Restricted Vertex Multicut problem.

Although many optimization problems that are NP-hard on arbitrary graphs are polynomially solvable on restricted graph classes [4, 12], not much seems to be known for the restricted variation of multicut on particular graph classes. The problem admits a polynomial-time algorithm for interval graphs [13], whereas it becomes NP-hard for trees [5, 13] and, thus, for chordal graphs which form a proper superclass of interval graphs. Therefore it is interesting to study the complexity of the Restricted Vertex Multicut problem on graph classes that are characterized without induced trees and are not included in the class of interval graphs.

In this paper we consider the restricted version of the multicut problem on split graphs, permutation graphs, and other related graph classes. Split graphs form a proper subclass of chordal graphs and such graphs are unrelated to interval graphs [12]. We prove that the problem remains NP-hard on split graphs. A natural superclass of interval graphs is the class of co-comparability graphs (complements of comparability graphs) where the complexity of the Restricted Vertex Multicut problem is still unknown. Co-bipartite (complements of bipartite graphs) and permutation graphs are two unrelated subclasses of co-comparability graphs [4, 12]. Interestingly most problems that are hard on co-comparability graphs are already hard on co-bipartite graphs. Here we show that the problem admits a simple and efficient (polynomial-time) solution on co-bipartite graphs and therefore excluding such an approach through a hardness result on co-bipartite graphs. Our main result is a polynomial-time algorithm for the class of permutation graphs. To do so, we take advantage of the notion of scanlines already efficiently applied for other problems such as treewidth and minimum fill-in on permutation graphs [1, 19]. We also give an independent result for cographs that can been seen as a special case of the polynomial-time algorithm on permutation graphs. An overall picture of our results is depicted in Figure 1.

2 Preliminaries

We consider undirected finite graphs with no loops or multiple edges. For a graph G, we denote its vertex and edge set by $V(G)$ and $E(G)$, respectively, with $n = |V(G)|$ and $m = |E(G)|$. For a vertex subset $S \subseteq V(G)$, the subgraph of G induced by S is denoted by $G[S]$. Moreover, we denote by $G - S$ the graph $G[V(G) \setminus S]$.

The *neighborhood* $N(x)$ of a vertex x of the graph G is the set of all the vertices of G which are adjacent to x. The *closed neighborhood* of x is defined as $N[x] = N(x) \cup \{x\}$. If $S \subseteq V(G)$, then the neighbors of S, denoted by $N(S)$, are given by $\bigcup_{x \in S} N(x) - S$. The *complement* $\overline{G}$ of a graph G has vertex set $V(G)$ and all edges not in G. A *clique* is a set of pairwise adjacent vertices while an *independent set* is a set of pairwise non-adjacent vertices. A graph is *connected* if there is a path between any pair of vertices. A *connected component* of G is a maximal connected subgraph of G. A chordless path on k vertices is denoted by P_k. A tree of height one is called a *star graph* and a star graph on four vertices is called *claw*.

For a set $\mathcal{F}$ of graphs, a graph is called *$\mathcal{F}$-free* if it does not contain a graph from $\mathcal{F}$ as induced subgraph. For all graph classes mentioned here proper definitions and characterizations can be found in [4, 12], though we give the corresponding characterizations at the appropriate places. We only mention at the moment that for every graph class Π that we consider here Π is closed under vertex removals, that is, Π is *hereditary*. The relationships between the considered graph classes are shown in Figure 1.

Let $G = (V, E)$ be a graph and let $L = \{(s_1, t_1), \ldots, (s_l, t_l)\}$ be a specified set of pairs of vertices, where the vertices of each pair are distinct, but vertices in different pairs are not required to be distinct. The set of vertices of L are called *terminals* denoted by T whereas the rest of the vertices are called *nonterminals*.

The Restricted Vertex Multicut problem can be formulated as follows.

Restricted Vertex Multicut
Input: An undirected graph $G = (V, E)$, a collection of pairs of vertices $L \subseteq V \times V$, and an integer $k \geq 0$.

Task: Find a subset S of V that contains only nonterminal vertices such that $|S| \leq k$ and vertices of each pair of L belong to different connected components of $G - S$.

Let us note that a feasible solution for the Restricted Vertex Multicut problem may not always exist. If there is a path P that connects a terminal pair and P consists only of terminal vertices then no vertex removal is possible to achieve a solution since terminal removals are not allowed. The same situation arises when s and t are adjacent for a terminal pair $(s,t) \in L$. Notice however that detecting whether a feasible solution exists is polynomial-time computable by checking for each $(s,t) \in L$ whether s and t belong to different connected components in the graph obtained by removing all nonterminal vertices.

In order to avoid repeating the following notions we let

- $G = (V, E)$ be a graph,
- $L = \{(s_1, t_1), \ldots, (s_l, t_l)\}$ be a set of non-adjacent terminal pairs,
- T be a set of terminals that appear in L,
- S be a solution for the Restricted Vertex Multicut problem on G; that is, $S \subseteq V(G)$, $S \cap T = \emptyset$, every pair of L is separated in $G - S$, and $|S|$ is minimum.

Notice that if k is part of the input then we ask for $|S| \leq k$. In the Unrestricted Vertex Multicut problem the vertices of the solution S do not require to be nonterminal vertices; that is, the restriction $S \cap T = \emptyset$ is omitted.

2.1 NP-completeness on split graphs

Next we prove that Restricted Vertex Multicut is NP-complete for the class of split graphs. As already mentioned in the Introduction the Unrestricted Vertex Multicut problem is NP-complete even on cliques by reducing from the Edge Multicut problem on star graphs. In order to present the NP-completeness reduction of split graphs, we give an alternative reduction for the Unrestricted Vertex Multicut problem. The reduction is made from the NP-complete Vertex Cover problem [8], in which for a given graph $G = (V, E)$ and an integer $k \geq 0$ we ask for a subset $V' \subseteq V$ with $|V'| \leq k$ such that for every edge $uv \in E$ at least one of u and v belongs to V'. In this way the NP-completeness of split graphs can be seen through an immediate reduction from the Vertex Cover problem.

Proposition 2.1. Unrestricted Vertex Multicut *on complete graphs is NP-complete.*

Proof. Given an arbitrary graph G' we construct an instance for the Unrestricted Vertex Multicut problem as follows. We add all missing edges of G' resulting in a complete graph H and for each edge of G' we add a pair of terminals to L. Thus an edge of G' corresponds to a pair $(s,t) \in L$ and vertices incident to an edge in G' are terminals in H. Clearly H can be constructed in polynomial time. Consider a vertex cover S' of G'. Removing S' from G' results in an edgeless graph which implies that no terminal pair of L belongs to the same connected component of $H - S'$. Thus a vertex cover S' of G' gives a solution S for the UNRESTRICTED VERTEX MULTICUT on H. For the converse observe that every terminal pair of L is adjacent in H and thus a solution S for H contains at least one of the two terminal vertices; also note that a nonterminal vertex of H is not contained in S. Hence at least one of the two terminal vertices belongs to S which implies that at least one of the endpoints of each edge in G' belongs to S. Therefore G' has a vertex cover S' of size at most k if and only if H has a vertex multicut S by removing at most k vertices. □

The above proposition shows that for graphs with arbitrary large cliques (all graph classes considered here), it is unlikely to have a polynomial-time algorithm for the UNRESTRICTED VERTEX MULTICUT problem. Studying the reduction it becomes clear that the hardness of the UNRESTRICTED VERTEX MULTICUT problem is mainly due to the structure of the terminal pairs and not due to the structure of the input graph itself. We show that a similar situation appears in the RESTRICTED VERTEX MULTICUT problem for the class of split graphs. A graph is a *split graph* if its vertex set can be partitioned into a clique C and an independent set I, where (C, I) is called a *split partition*.

Theorem 2.2. RESTRICTED VERTEX MULTICUT *on split graphs is NP-complete.*

Proof. We reduce the problem to the NP-complete problem UNRESTRICTED VERTEX MULTICUT restricted to cliques given in Proposition 2.1. Consider an instance of the UNRESTRICTED VERTEX MULTICUT on a complete graph G' with a list of terminal pairs L' and let T' be the set of terminal vertices. From G' we construct a split graph H as follows. For each pair $\{s', t'\} \in L'$ we add two non-adjacent vertices s and t that are only adjacent to s' and t', respectively. The list of terminal pairs for H is $L = \{(s, t) \mid (s', t') \in L'\}$ and the set T of terminal vertices contains only vertices of L. This particularly means that there is a bijection f between T and $V(G')$ such that for each $v \in T$, $f(v) = N(v) = \{u\}$ where $u \in V(G')$. Thus H is a split graph with split partition $(V(G'), T)$. We prove that the complete graph G' has an unrestricted vertex multicut S if and only if H has a restricted vertex multicut S. Since both vertices of a pair (s, t) of L have exactly one neighbor in H, a pair (s, t) of L is separated only if the pair $(f(s), f(t))$ of L' is separated. Thus given a solution S' for G', the same set of vertices of S' can be removed from H so that each pair of L is separated. For the opposite direction, observe that for a pair (s, t) of L we can remove $f(s)$ or $f(t)$ (or both) from H, meaning that we are only allowed to remove vertices of the clique. Thus given a solution S for H, the set S remains a solution for the UNRESTRICTED VERTEX MULTICUT problem on G'. Therefore S is a solution for H if and only if S is a solution for G' and this completes the proof. □

In Figure 2 we give a simple example of the proposed two-step reduction, directly through VERTEX COVER. The above proof shows that RESTRICTED VERTEX MULTICUT is NP-complete even on split graphs where each vertex of I of the split partition (C, I) has degree 1 and each vertex of C is adjacent to at most one vertex of I. Such graphs do not contain a claw as an induced subgraph. Therefore RESTRICTED VERTEX MULTICUT is NP-complete on claw-free graphs.

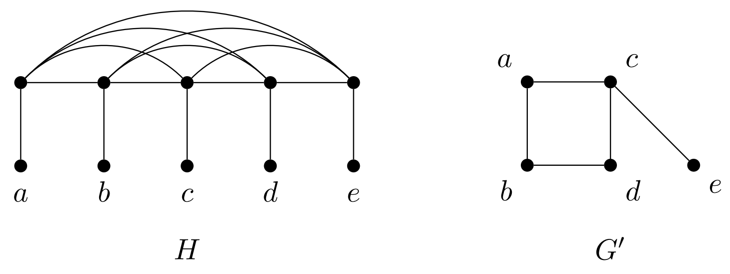

Figure 2: H is a split graph and G' is a graph constructed according to the terminal pairs $L = \{(a,b),(a,c),(b,d),(c,d),(c,e)\}$ for the Restricted Vertex Multicut problem in H. Notice that if we only remove $\{f(a), f(d)\}$ from H then there is path between the terminals c and e. A solution $S = \{f(b), f(c)\}$ for H corresponds to a solution $S' = \{b, c\}$ for the Vertex Cover problem in G' and vise versa.

2.2 Common neighborhood, cographs, and co-bipartite graphs

Let us now turn to positive results for the Restricted Vertex Multicut problem. As a first result it is an easy observation that the common neighbors of the terminals should be included in any solution. We show that such a reduction results in a simple algorithm for a certain graph class.

For simplicity we use $N_{st}(L)$ to denote the set of all common neighbors of terminal pairs, that is,

$$N_{st}(L) = \bigcup_{(s,t)\in L} N(s) \cap N(t).$$

We state the common neighborhood property in the following observation.

Observation 2.3. *S is a solution for G if and only if $N_{st}(L) \cap T = \emptyset$ and $S \setminus N_{st}(L)$ is a solution of $G - N_{st}(L)$.*

Proof. Consider a terminal pair $(s,t) \in L$. If a common neighbor of s and t is a terminal then there is no solution for the Restricted Vertex Multicut in G, since s and t can only be separated by the removal of a terminal vertex. Otherwise it is clear that every common neighbor of s and t belongs to a solution for G. □

Interestingly for the class of cographs such a simple observation readily applies. The class of cographs is strictly contained in the class of permutation graphs and we will show later a polynomial-time algorithm for permutation graphs. However for the sake of completeness we give a complete characterization on cographs for the Restricted Vertex Multicut problem. Cographs are exactly the graphs that do not contain a P_4 as an induced subgraph [4, 12].

Corollary 2.4. *Let G be a cograph. If $N_{st}(L) \cap T \neq \emptyset$ then there is no solution for G. Otherwise $S = N_{st}(L)$ is a solution for G.*

Proof. Due to the absence of a P_4 in a cograph G, two vertices of G are adjacent, or they have a common neighbor, or they are in different connected components. This particularly means that for a pair $(s,t) \in L$, there is no path between s and t in $G - N_{st}(L)$. Thus $N_{st}(L)$ is a solution for G. □

Therefore due to Corollary 2.4, Restricted Vertex Multicut in cographs can be solved in $\mathcal{O}(n|L|)$ time by computing the common neighbors for each terminal pair of L.

Next we consider a subclass of graphs that is not contained in the class of permutation graphs and settle its complexity status. More specifically we prove that for co-bipartite graphs the Restricted Vertex Multicut problem can be solved in polynomial time.

A graph is *bipartite* if its vertex set can be partitioned into two independent sets. The partition of a bipartite graph G into two independent vertex sets is called *bipartition* and this partition is unique if and only if G is connected. Bipartite graphs are exactly the class of graphs that do not contain cycles of odd length (see [4, 12]). Note that the Restricted Vertex Multicut remains NP-complete for trees [5, 13] and, thus, for bipartite graphs. The complement of a bipartite graph is called *co-bipartite graph* and the bipartition into two independent sets of a bipartite graph is a bipartition into two cliques in its complement. In other words, for a co-bipartite graph G there exists a partition (V_1, V_2) of V such that $G[V_1]$ and $G[V_2]$ are cliques.

Theorem 2.5. Restricted Vertex Multicut *in co-bipartite graphs can be solved in* $\mathcal{O}(m\sqrt{n} + |L|n)$ *time.*

Proof. Let $G = (V, E)$ be a co-bipartite graph with (V_1, V_2) its co-bipartition. Observe first that for a pair (s, t) of L we know that $s \in V_1$ and $t \in V_2$; otherwise we have no solution. At the beginning, for each pair (s, t) of L we remove the common neighbors of s and t by Observation 2.3. Let G' be the graph obtained after removing the common neighbors. That is, $G' = G - N_{st}(L)$ and G' is co-bipartite since we only removed vertices from a co-bipartite graph.

Let (V_1', V_2') be the co-bipartition of G'. Let A_1 be the set of vertices of V_1' having at least one neighbor in V_2' and let A_2 be the analogously subset of V_2'. We show that no terminal vertex of L belongs to $A_1 \cup A_2$. Assume for contradiction that there is a pair (s, t) such that $s \in A_1$. We know that $t \in V_2'$ which implies that every vertex of $N(s) \cap V_2' \neq \emptyset$ is adjacent to t. Due to the common neighborhood removal of (s, t) we reach to a contradiction. Hence for every pair (s, t) we know that $s \in V_1' \setminus A_1$ and $t \in V_2' \setminus A_2$. Observe that s is adjacent to every vertex of A_1, t is adjacent to every vertex of A_2, and we are forced to remove vertices only from $A_1 \cup A_2$. Thus in a solution there cannot be any edge between A_1 and A_2.

Consider the bipartite graph B taken from $G'[A_1 \cup A_2]$ where every edge between vertices of A_1 is removed and every edge between vertices of A_2 is removed. Then a minimum vertex cover for B is exactly the minimum set of vertices that can be removed from G', since removing the vertex cover set from G' disconnects the sets A_1 and A_2. Finding a minimum vertex cover in a bipartite graph takes time $\mathcal{O}(m\sqrt{n})$ [20]. Together with the $\mathcal{O}(|L|n)$ time needed for the first step of removing the common neighbors we obtain the stated result. □

3 A polynomial-time algorithm for the restricted vertex multicut problem on permutation graphs

In this section we show that Restricted Vertex Multicut can be solved in polynomial time for the class of permutation graphs. Recall that we already showed in Corollary 2.4 a polynomial-time algorithm for a proper subclass which can be seen as a special case for permutation graphs. Let π be a permutation over the set $N_n = \{1, \ldots, n\}$ and let $\pi^{-1}(i)$ be the index of i in π. We define the graph $G[\pi]$ with vertex set N_n and edge set $ij \in E(G[\pi])$ whenever $(i-j)(\pi^{-1}(i) - \pi^{-1}(j)) < 0$. A graph G is called *permutation graph* if there exists a permutation π such that G is isomorphic to $G[\pi]$. Permutation graphs can be represented by a model on a plane that is called *permutation diagram* [12] and is defined as follows: we take two horizontal lines and label points on the upper line with numbers 1 to n from left to right; on the lower horizontal line we label points with numbers $\pi(1)$ to $\pi(n)$ and connect two points with the same label between the horizontal lines by a line segment.

The connection between permutation graphs and permutation diagrams is given according to the intersection of the line segments and the edges of the graph. More precisely each vertex of the graph corresponds to a line segment and two vertices are adjacent if and only if the corresponding line segments cross in the diagram [12]. Figure 3 shows a permutation graph and its corresponding permutation diagram.

The algorithm for Restricted Vertex Multicut in permutation graphs is achieved through the notion of *scanlines* that were introduced in [1], and have been applied in several problems on permutation graphs [18, 19]. A scanline ℓ is a pair of numbers (x, y) where $x, y \in \{\frac{1}{2}, 1\frac{1}{2}, \ldots, n\frac{1}{2}\}$. We say that a scanline $\ell = (x, y)$ *crosses* a vertex i if either $i < x$ or $\pi(i)^{-1} < y$. The set of vertices that are crossed by a scanline ℓ is denoted by $Q(\ell)$.

Let $G[\pi]$ be a permutation graph and let u, v be two vertices with $u < v$. If u and v are non-adjacent then according to the definition of permutation graph we have $\pi^{-1}(u) < \pi^{-1}(v)$. For two non-adjacent vertices u, v with $u < v$ we define the following set of scanlines that are between u and v:

$$A(u,v) = \left\{(x,y) \mid x \in \left\{u\frac{1}{2}, \ldots, (v-1)\frac{1}{2}\right\} \text{ and } y \in \left\{\pi^{-1}(u)\frac{1}{2}, \ldots, \pi^{-1}(v-1)\frac{1}{2}\right\}\right\}.$$

The usage of scanlines in accordance to the Restricted Vertex Multicut problem can be seen through the following result. We note here that the results of [1, 19] imply the existence of a similar scanline. However we explicitly prove the following without involving or introducing further notions relevant to other problems.

Lemma 3.1. *Let $G[\pi]$ be a permutation graph and let $(u, v) \in L$ be a terminal pair so that $u < v$. If there exists a solution S for* Restricted Vertex Multicut *on $(G[\pi], L)$ then there exists a scanline $\ell \in A(u, v)$ such that $Q(\ell) \subset S$ and $Q(\ell) \cap T = \emptyset$.*

Proof. Suppose first that there is no path between u and v. Let C_u and C_v be the connected components of u and v, respectively. Let u' be the rightmost vertex of C_u and let v' be the vertex of C_v with the leftmost $\pi^{-1}(v')$. Observe that $u \leq u' < v$ and $\pi^{-1}(u) < \pi^{-1}(v') \leq \pi^{-1}(v)$ since there is no crossing of line segments between C_u and C_v. Then for every vertex $w \in C_u$ it holds that $\pi^{-1}(w) < \pi^{-1}(v')$ and, similarly, for every vertex $z \in C_v$ it holds that $u' < z$. Hence there exists a scanline $\ell = (u'\frac{1}{2}, (\pi^{-1}(v') - 1)\frac{1}{2})$ such that $Q(\ell) = \emptyset$.

Now suppose that there is a path between u and v. Then from the permutation diagram we know that every path between u and v has a vertex w for which $u < w < v$.

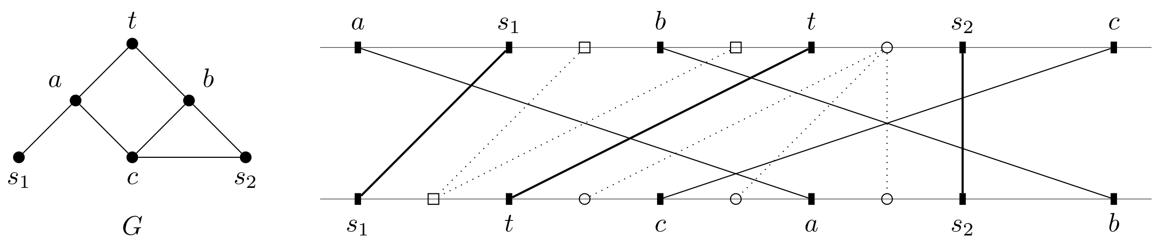

Figure 3: A permutation graph G and its corresponding permutation diagram. For the presented example we let the terminal pairs $L = \{(s_1, t), (s_2, t)\}$. With dotted lines we represent the scanlines of $A(s_1, t)$ with squared end-points and $A(s_2, t)$ with circled end-points that do not cross a terminal. According to the algorithm observe that $S_{t\frac{1}{2}} = \{a\}$ which implies $S_{s_2\frac{1}{2}} = \{a, b\}$ as the overall solution.

In order to separate u and v there must be a set of vertices $S_{uv} \subseteq S$ such that u and v belong to different connected components of $G[\pi] - S_{uv}$ and for every vertex w of S_{uv}, $u < w < v$. As shown previously in $G[\pi] - S_{uv}$ we can choose two vertices u' and v' that belong to the connected component of u and v, respectively, with the properties mentioned above. Let a scanline $\ell = (u'\frac{1}{2}, (\pi^{-1}(v')-1)\frac{1}{2})$. Every vertex that is crossed by ℓ in $G[\pi]$ must belong to S_{uv} meaning that $Q(\ell) \subseteq S_{uv}$. Furthermore no terminal belongs to S_{uv} since S_{uv} is part of the solution. Therefore $Q(\ell) \subseteq S_{uv}$ and $S_{uv} \cap T = \emptyset$. $\square$

We are now ready to state our algorithm. Let $G = G[\pi]$ be a permutation graph and let L be a collection of terminal pairs. For $1 \leq i \leq n$ we denote the graph $G_i = G[\{1, \ldots, i\}]$ and L_i to denote the terminal pairs of G_i. In a vertex-incremental fashion starting from vertex 1 we compute the solution $S_{i\frac{1}{2}}$ for the graph G_i and terminal pairs L_i. Each solution $S_{i\frac{1}{2}}$ is computed based on the solutions $S_{j\frac{1}{2}}$ for $0 \leq j < i$. At the end $S_{n\frac{1}{2}}$ is a solution for the graph $G = G_n$ and terminal pairs $L = L_n$.

Algorithm Multicut_Permutation
Input: a permutation graph $G = G[\pi]$ and a collection of terminal pairs L
Output: a vertex set S such that every pair of L is separated in $G - S$ and $|S|$ is minimum

```
1. Let $S_{1/2} = \emptyset$
2. for $i = 1, \ldots, n$ do
3.     Let $s, t \in \{1, \ldots, i\}$ such that $s < t$, $(s, t) \in L$, and $s$ is maximum
4.     Let $A^T(s, t) = \{\ell \mid \ell \in A(s, t)$ and $Q(\ell) \cap T = \emptyset\}$
5.     for every scanline $\ell = (x, y) \in A^T(s, t)$ do
6.         $S_i^\ell = S_x \cup Q(\ell)$
7.     Let $S_{i\frac{1}{2}} = S_i^\ell$ with minimum $|S_i^\ell|$
8. Return $S = S_{n\frac{1}{2}}$
```

We illustrate an example of the algorithm in the graph given in Figure 3. The main result of this section is given in the following theorem.

Theorem 3.2. Restricted Vertex Multicut *in permutation graphs can be solved in $\mathcal{O}(n^3)$ time.*

Proof. Such an algorithm is described in Multicut_Permutation. Let us first show the correctness of the algorithm. For every graph G_i, $1 \leq i \leq n$, we collect the current solution in $S_{i\frac{1}{2}}$. At vertex i we consider the pair (s,t) with $s < t \leq i$ that needs to be separated. By Lemma 3.1 there is a scanline ℓ between s and t for which $Q(\ell)$ does not contain a terminal vertex and $Q(\ell)$ is included in the solution. Thus among all scanlines $\ell = (x,y) \in A(s,t)$ for which $Q(\ell) \cap T = \emptyset$ we compute the set $S_i^\ell = S_x \cup Q(\ell)$ with the minimum $|S_i^\ell|$ at lines 5–7. If $Q(\ell) \cap T \neq \emptyset$ for every ℓ (that is, $A^T(s,t) = \emptyset$) then clearly there is no solution for G. Observe that the solution S_x corresponds to the solution for the graph G_j where $j = (x-1)\frac{1}{2} < i$. If we remove the vertices of S_i^ℓ then G_i breaks into two induced subgraphs $G_j - Q(\ell)$ and $G[\{j+1,\ldots,i\}] - Q(\ell)$. Since for the terminal pair (s,t) we choose s to be maximum at line 3, there is no terminal pair in the graph $G[\{j+1,\ldots,i\}]$. Thus $S_{i\frac{1}{2}}$ is obtained for a suitable choice of pair (x,y) for which $|S_x| + |Q(\ell)|$ is minimized.

Regarding the running time of the algorithm notice that the permutation diagram (and therefore the permutation π and the ordering of the vertices) can be computed in $\mathcal{O}(n+m)$ time [4, 12]. Computing each $S_{i\frac{1}{2}}$ requires $\mathcal{O}(n^2)$ time since there are at most i^2 scanlines in $A(s,t)$. Moreover for a fixed $\ell = (x,y)$ the set $Q(\ell)$ can be computed in $\mathcal{O}(n)$ time. Therefore the overall running time of the algorithm is $\mathcal{O}(n^3)$. □

4 Concluding remarks

A future research direction is resolving the computational complexity of Restricted Vertex Multicut for larger classes of permutation graphs. It seems that the proposed algorithm for permutation graphs has the potential of being generalizable to larger graph classes, such as trapezoid, or more general, d-trapezoid graphs, or co-comparability graphs of bounded dimension. Note that these graph classes already have an established scanline notion [2]. Moreover permutation graphs are exactly the graphs that are both comparability and co-comparability [4, 12]. For comparability graphs it follows that the problem is NP-complete since every tree is a comparability graph. In the case of co-comparability graphs the complexity status remains open.

Furthermore the characterization given in Corollary 2.4 for P_4-free graphs implies that the solution consists of a separator set S for each pair of terminals without taking into account its (minimum) cardinality. As an interesting dichotomy result notice that split graphs do not contain P_5 or larger chordless paths and, therefore, Restricted Vertex Multicut remains NP-complete for the class of P_k-free graphs for $k \geq 5$ whereas due to Corollary 2.4 the problem is polynomial-time solvable for $k \leq 4$. For split graphs it is interesting to determine the maximum number $k > 2$ of terminal pairs for which the problem admits a polynomial solution. We note that both problems Edge Multicut and Restricted Vertex Multicut for general graphs on at least 3 terminal pairs are NP-complete [7, 13].

Acknowledgements

The author would like to thank the anonymous referees whose valuable suggestions helped improve the presentation of the paper.

References

[1] H. Bodlaender, T. Kloks, and D. Kratsch. Treewidth and pathwidth of permutation graphs. *SIAM Journal on Discrete Mathematics*, 8:606–616, 1995.

[2] H. Bodlaender, T. Kloks, D. Kratsch, and H. Müller. Treewidth and minimum fill-in on d-trapezoid graphs, *Journal of Graph Algorithms and Applications*, 2:1–23, 1998.

[3] N. Bousquet, J. Daligault, and S. Thomassé, Multicut is FPT. In *Proceedings of STOC 2011*, pp. 459–468, 2011.

[4] A. Brandstädt, V. B. Le, and J. Spinrad. *Graph Classes: A Survey.* SIAM Monographs on Discrete Mathematics and Applications, vol. 3, Philadelphia, 1999.

[5] G. Călinescu, C. G. Fernandes, and B. A. Reed. Multicuts in unweighted graphs and digraphs with bounded degree and bounded tree-width. *Journal of Algorithms*, 48:333–359, 2003.

[6] M.-C. Costa, L. Létocart, and F. Roupin. Minimal multicut and maximal integer multiflow: A survey. *European Journal of Operational Research*, 162:55–69, 2005.

[7] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P.D.Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23:864–894, 1994.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-completeness.* Freeman, 1979.

[9] N. Garg, V. V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM Journal on Computing*, 25:235–251, 1996.

[10] N. Garg, V. V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18:3–20, 1997.

[11] N. Garg, V. V. Vazirani, and M. Yannakakis. Multiway cuts in node weighted graphs. *Journal of Algorithms*, 50:49–61, 2004.

[12] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Annals of Discrete Mathematics (second ed.), vol. 57, Elsevier, 2004.

[13] J. Guo, F. Huffner, E. Kenar, R. Niedermeier, and J. Uhlmann. Complexity and exact algorithms for vertex multicut in interval and bounded treewidth graphs. *European Journal of Operational Research*, 186:542–553, 2008.

[14] T. C. Hu. *Integer Programming and Network Flows*. Addison-Wesley, Reading, MA, 1969.

[15] D. Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351:394–406, 2006.

[16] D. Marx and I. Razgon. Constant ratio fixed-parameter approximation of the edge multicut problem. *Information Processing Letters*, 109:1161–1166, 2009.

[17] D. Marx and I. Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. In *Proceedings of STOC 2011*, pp. 469–478, 2011.

[18] D. Meister. A characterisation of the minimal triangulations of permutation graphs. In *Proceedings of WG 2007*, LNCS 4769:99–108, 2007.

[19] D. Meister. Treewidth and minimum fill-in on permutation graphs in linear time. *Theoretical Computer Science*, 411:3685–3700, 2010.

[20] S. Micali and V.V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *Proceedings of FOCS 1980*, pp. 17–27, 1980.

[21] M. Yannakakis, P. C. Kanellakis, S. S. Cosmadakis, and C. H. Papadimitriou. Cutting and partitioning a graph after a fixed pattern. In *Proceedings of ICALP 1983*, pp. 712–722, 1983.