The most vital nodes with respect to Independent Set and Vertex Cover

Cristina Bazgan [1] Sonia Toubaline [1] Zsolt Tuza [2,*]

[1] Université Paris-Dauphine, LAMSADE
Place du Marechal de Lattre de Tassigny, 75775 Paris Cedex 16, France

[2] Computer and Automation Institute, Hungarian Academy of Sciences
H–1111 Budapest, Kende u. 13–17, Hungary
and
Department of Computer Science and Systems Technology, University of Pannonia
H–8200 Veszprém, Egyetem u. 10, Hungary

Abstract

Given an undirected graph with weights on its vertices, the k most vital nodes independent set (k most vital nodes vertex cover) problem consists of determining a set of k vertices whose removal results in the greatest decrease in the maximum weight of independent sets (minimum weight of vertex covers, respectively). We also consider the complementary problems, minimum node blocker independent set (minimum node blocker vertex cover) that consists of removing a subset of vertices of minimum size such that the maximum weight of independent sets (minimum weight of vertex covers, respectively) in the remaining graph is at most a specified value. We show that these problems are NP-hard on bipartite graphs but polynomial-time solvable on unweighted bipartite graphs. Furthermore, these problems are polynomial also on cographs and graphs of bounded treewidth. Results on the non-existence of ptas are presented, too.

Keywords: most vital vertices, independent set, vertex cover, time complexity, NP-hard, bipartite graph, bounded treewidth, cograph

Mathematics Subject Classification: 05C85, 05C69

1 Introduction

In many applications involving the use of communication or transportation networks, we often need to identify vulnerable or critical infrastructures. By critical infrastructure we mean a set of nodes/links whose damage causes the largest increase in the cost within the network. Modeling the network by a weighted graph, identifying a vulnerable infrastructure amounts to finding a subset of vertices/edges of a given size whose removal from the graph causes the largest inconvenience to a particular property of the graph in question. In the literature this problem is referred to as the *k most vital nodes/edges* problem. A complementary problem consists of determining a set of vertices/edges of minimum size whose removal involves that the cost within the network is at most a given value. In the literature this problem is referred to as the *min node/edge blocker* problem.

[*] Research supported in part by the Hungarian Scientific Research Fund, OTKA grant 81493

The problems of k most vital nodes/edges and min node/edge blocker have been studied for various problems, including shortest path, spanning tree, maximum flow, assignment, 1-median, 1-center and maximum matching. The k most vital edges problem with respect to shortest path was proved NP-hard [1]. Later, k most vital edges/nodes shortest path (and min node/edge blocker shortest path, respectively) were proved not 2-approximable (not 1.36-approximable, respectively) if P$\neq$ NP [15]. For spanning tree, k most vital edges is NP-hard [11] and $O(\log k)$-approximable [11]. In [22] it is proved that k most vital edges maximum flow is NP-hard. Also k most vital edges and min edge blocker assignment are proved NP-hard and not 2-approximable (not 1.36-approximable, respectively) if P$\neq$ NP [2]. In [4] it is proved that k most vital edges (nodes) 1-median (1-center) and min edge (node) blocker 1-median (1-center) are NP-hard to approximate within a factor c, for some $c > 1$. For maximum matching, min edge blocker is NP-hard even for unweighted bipartite graphs [24], but polynomial for grids and trees [20]; and the most vital nodes problem is NP-hard for weighted bipartite graphs but polynomial for unweighted ones, both results proved in [23].

In this paper, we are interested in determining a subset of k *vertices* of the graph whose deletion causes the largest decrease in the maximum weight of an independent set or the minimum weight of a vertex cover. These problems are referred to as k MOST VITAL NODES INDEPENDENT SET and k MOST VITAL NODES VERTEX COVER . We also consider the complementary versions of these problems, where given a threshold, we have to determine a subset of vertices of minimum size that has to be removed such that in the resulting graph the maximum-weight independent set or minimum-weight vertex cover is at most this threshold. These problems are referred to as MIN NODE BLOCKER INDEPENDENT SET and MIN NODE BLOCKER VERTEX COVER.

In Section 3 we consider bipartite graphs. It turns out that a substantial jump in complexity occurs between unweighted and weighted graphs for all these four problems. More precisely we show that the unweighted versions are polynomial while the weighted versions are NP-hard and the most vital nodes problems have no ptas. In Section 4 we deal with graphs with weights on their vertices, which have either a tree-like structure or a representation associated with trees. These include trees themselves, cycles, more generally graphs of bounded treewidth, and cographs (graphs containing no induced P_4). For these classes we design polynomial-time algorithms for all the four problems mentioned above.

In fact, trees and cycles have treewidth 1 and 2, respectively, therefore our general algorithm for bounded treewidth works for the former classes, too. Nevertheless, the algorithms on trees and cycles are simpler and this is why we include them here. It should be noted further that for k fixed, k MOST VITAL NODES INDEPENDENT SET and k MOST VITAL NODES VERTEX COVER are polynomial-time equivalent to the case $k = 0$ and since there are only polynomially many subsets of k removable vertices, therefore k MOST VITAL NODES INDEPENDENT SET and k MOST VITAL NODES VERTEX COVER are solvable efficiently on every graph class where the largest independent set and smallest vertex cover are tractable. On the other hand if $k \rightarrow \infty$ then a formula expressing the present problems in second-order monadic logic would have unbounded length. Consequently, the general approach to linear-time algorithms via second-order monadic logic (MSOL) is not applicable here.

In every graph, independent sets and vertex covers are complementary, and an independent set is of maximum weight if and only if its complement is a vertex cover of minimum weight. Contrary to this, however, it follows from our results that for $k \geq 1$ the optimal solutions of k MOST VITAL NODES INDEPENDENT SET and k MOST VITAL NODES VERTEX COVER can be substantially different.

The present paper is a substantially extended version of the limited-length conference contribution [3] where only independent sets are considered and only a part of proofs is included and only the independence number is studied. In particular, the non-approximability of most vital nodes for vertex cover has never been investigated before.

2 Preliminaries

Let $G = (V, E)$ be an undirected graph, $V = \{v_1, \ldots, v_n\}$, where each vertex v_i has a weight w_i. Given an edge $e = v_iv_j \in E$, by convenient abuse of notation, we shall write $v_i, v_j \in e$ and if $v_i, v_j \in V', V' \subseteq V$ then we shall write that $e \subset V'$. When removing a set V' of vertices from G, let us denote the remaining graph by $G - V'$. If H is a subgraph of G then $V(H)$ denotes the vertex set of H. Moreover, for a subset V' of vertices from G, the subgraph induced by V' is denoted by $G[V']$.

A maximum-weight independent set of G is a subset of vertices of maximum weight where any two vertices are nonadjacent. A minimum-weight vertex cover of G is a subset of vertices of minimum weight where every edge of G contains at least one vertex of the subset. We denote by $\alpha(G)$ the maximum weight of an independent set and by $\tau(G)$ the minimum weight of a vertex cover. Moreover, $\alpha(k)$ represents the minimum of $\alpha(G - V')$ after removing any set of vertices V' of size k; $\tau(k)$ is defined similarly. A matching is a set of mutually vertex-disjoint edges. The largest number of edges in a matching is denoted by $\nu(G)$.

In this paper we are interested in studying the complexity of the following problems.

k Most Vital Nodes Independent Set
Input: An undirected graph $G = (V, E)$ where each vertex v_i has a weight w_i, and an integer k.
Output: A subset $V' \subseteq V$ of size k such that the maximum weight $\alpha(G - V')$ of an independent set in $G - V'$ is minimum.

k Most Vital Nodes Vertex Cover
Input: An undirected graph $G = (V, E)$ where each vertex v_i has a weight w_i, and an integer k.
Output: A subset $V' \subseteq V$ of size k such that the minimum weight $\tau(G - V')$ of a vertex cover in $G - V'$ is minimum.

Min Node Blocker Independent Set
Input: An undirected graph $G = (V, E)$ where each vertex v_i has a weight w_i, and an integer U.
Output: A subset $V' \subseteq V$ of minimum size such that the maximum weight $\alpha(G - V')$ of an independent set in $G - V'$ is at most U.

Min Node Blocker Vertex Cover
Input: An undirected graph $G = (V, E)$ where each vertex v_i has a weight w_i, and an integer U.
Output: A subset $V' \subseteq V$ of minimum size such that the minimum weight $\tau(G - V')$ of a vertex cover in $G - V'$ is at most U.

Remark 1 *k Most Vital Nodes Independent Set and Min Node Blocker Independent Set are polynomial-time equivalent. Indeed, if an algorithm $\mathcal{A}_k$ solves k Most Vital Nodes Independent Set for all $1 \leq k \leq n$, then we can run $\mathcal{A}_k$ for $k = 1, \ldots, n$ and choose the smallest k yielding optimum at most U. Conversely, if an algorithm $\mathcal{B}_U$ solves Min Node Blocker Independent Set with any bound U, we can apply binary search to locate the smallest U that requires the removal of at most k vertices.*

Applying binary search and its accelerated logarithmic version, we obtain the following relation between the 'most vital nodes' and 'min node blocker' problems.

Lemma 1 *If there exists an algorithm that solves the k most vital nodes version of an optimization problem $\mathcal{P}$ on graphs with n vertices in $O(t)$ time, then the min node blocker version of $\mathcal{P}$ can be solved in $O(t \log n)$ time. Moreover, for any $\epsilon > 0$, the optimum for min blocker can be approximated within $(1 + \epsilon)$ in $O(t(\log \log n + \log 1/\epsilon))$ time.*

Proof: If the value of an optimum solution is at most U, then the optimal blocker is the empty set, which can be tested in $O(t)$ time by assumption. Otherwise, to obtain a $(1 + \epsilon)$-approximation we first apply the approach of [13] to design a 16-approximation. We recursively compute triples (ℓ, u, i) such that ℓ is a lower bound, u is an upper bound, and $(u/\ell)^{1/4} \leq 2^{2^i} < (u/\ell)^{1/2}$. The values are initialized to $\ell_0 = 1$, $u_0 = n$, $i_0 = \lceil \log \log n \rceil - 2$; they clearly satisfy $(u_0/\ell_0)^{1/4} = \sqrt[4]{n} \leq 2^{2^{i_0}} < \sqrt{n} = (u_0/\ell_0)^{1/2}$ for all $n > 1$.

To determine the next triple (ℓ', u', i') if (ℓ, u, i) is already at hand, we test in $O(t)$ time whether the optimum is above or under $k := \ell \cdot 2^{2^i}$. Depending on the answer, $\ell \cdot 2^{2^i}$ becomes either ℓ' or u', and we keep $u' = u$ or $\ell' = \ell$ accordingly. The update from i to i' is very easy, for the following reason. We clearly have $i' \leq i$ because we never increase u or decrease ℓ. For $u' = u$ we apply the condition $2^{2^i} < (u/\ell)^{1/2}$ and obtain

$$\left(\frac{u'}{\ell'}\right)^{1/4} > \left(\frac{u}{(u\ell)^{1/2}}\right)^{1/4} = \left(\frac{u}{\ell}\right)^{1/8} > 2^{2^{i-2}}.$$

Similarly, for $\ell' = \ell$ we apply $2^{2^i} \geq (u/\ell)^{1/4}$ and obtain

$$\left(\frac{u'}{\ell'}\right)^{1/4} > \left(\frac{u^{1/4}\,\ell^{3/4}}{\ell}\right)^{1/4} = \left(\frac{u}{\ell}\right)^{1/16} > 2^{2^{i-3}}.$$

Since $2^{2^{i'}}$ should not be smaller than the left-hand side, $i' \geq i - 2$ must hold in either case. Thus, selecting the proper value of $i' \in \{i-2, i-1, i\}$ requires at most two comparisons, checking whether $i' = i$ or $i' = i - 1$ works for (u', ℓ').

On the other hand, for $u' = u$ the condition $2^{2^i} \geq (u/\ell)^{1/4}$ implies

$$\frac{u'}{\ell'} \leq \frac{u}{u^{1/4}\,\ell^{3/4}} = \left(\frac{u}{\ell}\right)^{3/4},$$

and for $\ell' = \ell$ we can use $2^{2^i} < (u/\ell)^{1/2}$ to obtain

$$\frac{u'}{\ell'} \leq \frac{(u\ell)^{1/2}}{\ell} = \left(\frac{u}{\ell}\right)^{1/2}.$$

The former upper bound is less restrictive, but in any case after three iterations we surely have

$$\left(\frac{u'''}{\ell'''}\right)^{1/2} < \left(\frac{u}{\ell}\right)^{27/128} < \left(\frac{u}{\ell}\right)^{1/4} < 2^{2^i},$$

and consequently $i''' < i$ holds. This implies that after at most $O(\log\log n)$ iterations we reach $i = 0$, which means $(u/\ell)^{1/4} \le 2$ and $u \le 16\ell$. Then we need at most $\lceil 3 + \log 1/\epsilon \rceil$ steps of binary search to obtain a pair (u^*, ℓ^*) with $u^* \le (1+\epsilon)\ell^*$. □

Remark 2 *On some restricted classes of problem instances, the algorithm above can be used to determine not only approximate but also exact solutions of min node blocker problems more efficiently than $O(t\log n)$. Namely, if a class satisfies opt $= n^{o(1)}$ for all feasible instances, then we can proceed as follows. First, applying logarithmic binary search, find a 16-approximation (u, ℓ) in $O(t\log\log n)$ time. Then $u - \ell = n^{o(1)}$ holds, and hence binary search to find exact optimum takes as short as $o(t\log n)$ time. This corresponds to the choice $\epsilon = 1/u$.*

For proofs concerning the non-existence of a ptas (polynomial-time approximation scheme), we shall use the notion of gap-preserving reduction introduced in [19].

Let A be a maximization problem and A' a minimization problem. Then A is said to be *gap-preserving reducible* to A' with parameters $(c, \rho), (c', \rho')$ (where $\rho, \rho' \ge 1$), if there is a polynomial-time algorithm that transforms any instance x of A to an instance x' of A' such that the following properties hold:

1. $opt_A(x) \ge c \Rightarrow opt_{A'}(x') \le c'$
2. $opt_A(x) < \frac{c}{\rho} \Rightarrow opt_{A'}(x') > \rho' \cdot c'$

Gap-preserving reductions have the following property. If it is NP-hard to decide whether the optimum of an instance of A is at least c or less than $\frac{c}{\rho}$, then it is NP-hard to decide whether the optimum of an instance of A' is at most c' or greater than $\rho' \cdot c'$. This NP-hardness implies that A' is hard to ρ'-approximate.

3 Complexity on bipartite graphs

In a graph, a maximum independent set is a complementary set of a minimum vertex cover, even for weighted graphs. Nevertheless, concerning the k most vital nodes (min node blocker) versions an optimum solution for k Most Vital Nodes Independent Set (Min Node Blocker Independent Set) is not necessarily an optimum solution for k Most Vital Nodes Vertex Cover (Min Node Blocker Vertex Cover), even for unweighted bipartite graphs. A class of counterexamples is that of complete bipartite graphs with vertex classes of unequal size, i.e. the graphs $K_{n,m}$ with $n > m \ge 1$. Assume $1 \le k \le \min(m, n-m)$. Then the optimum solution for k Most Vital Nodes Independent Set is to remove k vertices from the larger vertex class, this decreases the independence number from n to $n-k$; whereas for k Most Vital Nodes Vertex Cover we have to remove k vertices from the smaller vertex class, this decreases the minimum size of a vertex cover from m to $m-k$. Hence, there is a substantial difference already for $k = 1$, as illustrated by the instance G from Figure 1. The vertex labeled 1 is critical with respect to the vertex covering number (its

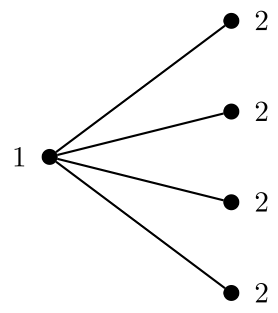

Figure 1: Instance G

removal yields a subgraph whose minimum vertex cover is the empty set), and each vertex labeled 2 is critical with respect to the independence number, but not conversely.

Maximum-weight independent set and minimum-weight vertex cover are polynomial-time solvable on bipartite graphs using the Kőnig-Egerváry theorem [10]. We show in this section that the k most vital nodes and min node blocker versions become NP-hard on bipartite graphs and k most vital nodes do not admit a ptas. Nevertheless, all these problems remain polynomial-time solvable in the unweighted case. We first prove this latter fact. Its 'min node blocker' part was proved independently by Costa *et al.* [8].

Theorem 1 *k Most Vital Nodes Independent Set and k Most Vital Nodes Vertex Cover and also Min Node Blocker Independent Set and Min Node Blocker Vertex Cover are polynomial for unweighted bipartite graphs. Moreover, if a largest matching and a smallest vertex cover are given with the input, all these problems are solvable in linear time.*

Proof: Let $G = (V, E)$ be a bipartite input graph on n vertices. From Kőnig's theorem [18] we know that $\tau(G) = \nu(G)$ holds; let us denote here their common value by t. The classical proof of the equality $\tau = \nu$ is algorithmic and also yields a maximum matching $M = \{e_1, \ldots, e_t\}$ and a minimum vertex cover $X = \{v_1, \ldots, v_t\}$ in polynomial time. Moreover, we have $\alpha(G) = n - t$ (known as a Gallai-type identity) and $V \setminus X$ is a largest independent set in G. Let us introduce the further notation $R = V \setminus V(M)$ and $r = |R| = n - 2t$; i.e., the number and the set of vertices not contained in any of the matching edges in M.

We can show now that all the four problems are solvable in linear time, as follows.

k Most Vital Nodes Independent Set

If $k \leq |R|$, we remove any k vertices from R. Since the remaining graph (of order $n-k$) still contains the matching M of size t, the independence number cannot be larger than $n - k - t$. It is also clear that α cannot be decreased by more than k if we remove just k vertices, hence the solution obtained is optimal.

If $k > |R|$, we remove the entire R and the vertices of $\lfloor (k - r)/2 \rfloor$ edges from M, and one further vertex if $k - r$ is odd. This decreases the size of M by $\lceil (k - r)/2 \rceil$ and the independence number by $\lfloor (k + r)/2 \rfloor$, and hence the new value is $\lceil (n - k)/2 \rceil$ (originally we had $\alpha(G) = (n + r)/2$). This decrease is optimal, because after the removal of k vertices at least half of the remaining $n - k$ belong to the same vertex class.

k Most Vital Nodes Vertex Cover

If $k \leq t$, we simply remove k vertices of X. The remaining part of X is a vertex cover in the smaller graph, hence τ is decreased by exactly k, which is optimal because at least $t-k$ edges of M would remain in the graph after the removal of any k vertices. If $k > t$, then removing X the graph becomes edgeless and the $k-t$ vertices outside X can be chosen arbitrarily for removal.

Min Node Blocker Independent Set

If $U \geq n-t$, no vertices need to be removed. If $t \leq U < n-t$, we remove $n-t-U$ vertices of R. If $U = t-\ell$ where $1 \leq \ell \leq t$, we remove the entire R and the 2ℓ vertices of ℓ arbitrarily chosen edges from M. All these choices are optimal, as follows from the proof concerning most vital nodes.

Min Node Blocker Vertex Cover

All we need is to remove $t-U$ vertices of X. □

Remark 3 If we are interested in determining just the *number* of vertices to be removed for Min Node Blocker Independent Set and Min Node Blocker Vertex Cover, given n and τ of the bipartite input graph, the problem is solvable in *constant* time because the answer can be written as an explicit function of n and τ.

We show next that the four problems become NP-hard in the weighted case. The following notion will be of essence.

Definition 1 *Let $G = (V, E)$ be an undirected graph. The* bipartite incidence graph *of G is the bipartite graph H whose vertex set is $V \cup E$ and there is an edge in H between $v \in V$ and $e \in E$ if and only if e is incident to v in G.*

In order to prove NP-hardness, beside (unweighted) Independent Set we shall also consider the decision problem associated to its complementary version, Clique, defined as follows:

Clique
Input: A graph $G = (V, E)$ and an integer ℓ.
Question: Does G contain a clique of size at least ℓ, that is a subset $V' \subseteq V$ with $|V'| \geq \ell$ such that every two vertices in V' are joined by an edge in E ?

Clique is one of the well known NP-hard problems [14]. We can consider that $\ell > 3$ since otherwise Clique is solvable in polynomial time.

Theorem 2 *k Most Vital Nodes Independent Set and k Most Vital Nodes Vertex Cover and also Min Node Blocker Independent Set and Min Node Blocker Vertex Cover are strongly NP-hard even for bipartite graphs.*

Proof: We first prove hardness for k Most Vital Nodes Independent Set and k Most Vital Nodes Vertex Cover. For both problems let the instance be a graph $G = (V, E)$ with n vertices and m edges, and an integer ℓ; and let H denote the bipartite incidence graph of G. The construction of H from G requires linear time only.

For k Most Vital Nodes Independent Set we make reduction from the decision problem associated to Independent Set. Each vertex of E in H has weight 1 and each vertex of V in H has weight n^2. Due to this rather unbalanced weighting, the unique maximum-weight independent set in H is V; i.e., $\alpha(H) = n^3$.

We show in the following that if there is an independent set of size at least ℓ in G then H contains a set S of ℓ vertices such that $\alpha(H - S) = (n - \ell)n^2$, and otherwise removing any subset S of ℓ vertices from H, we have $\alpha(H - S) \geq (n - \ell)n^2 + 1$. Since vertices from V have weight n^2 and those from E have weight 1, in order to have a maximum-weight independent set as small as possible after removing a set S of size ℓ, S has to be included in V.

If G contains an independent set S of size ℓ, then removing S from the vertex set of H, we obtain a graph whose maximum-weight independent set is $V \setminus S$. This set has weight $(n-\ell)n^2$.

If G contains no independent set of size ℓ, then any $S \subset V$ of size ℓ contains at least an edge $e \in E$ in G, and this e in H is nonadjacent to the entire $V \setminus S$. Thus, when we remove any set S of ℓ vertices from H, $\alpha(H - S) \geq (n - \ell)n^2 + 1$.

In order to prove the NP-hardness of k Most Vital Nodes Vertex Cover, we apply the decision problem associated to Clique, with the same weights on H as above. Since the entire E has smaller weight than just one vertex from V, minimum vertex covers in any subgraph of H are subsets of E. We show that if there is a clique of size at least ℓ in G then $\tau(H - S) = m - \frac{\ell(\ell-1)}{2}$, and otherwise removing any subset S of ℓ vertices from H, we have $\tau(H - S) \geq m - \frac{\ell(\ell-1)}{2} + 1$.

If G contains a clique V' of size ℓ, then removing V' from the vertex set of H, we obtain a graph whose minimum-weight vertex cover is $E \setminus E'$, where E' is the edge set induced by V' in G. This vertex cover has weight $m - \frac{\ell(\ell-1)}{2}$.

Suppose that G contains no clique of size ℓ. Let $S \subset V \cup E$ be a set of ℓ vertices in H. From a vertex cover, this S saves us the selected edges $S \cap E$, plus those edges of G from the E-part of H which have both endpoints in $S \cap V$. Consequently,

$$\tau(H - S) \geq m - |E(G[S \cap V])| - |S \cap E| \geq m - \frac{\ell(\ell - 1)}{2} + 1$$

where the last inequality holds because if $S \cap E \neq \emptyset$ then $S \cap V$ induces at most $\binom{\ell - |S \cap E|}{2} < \binom{\ell}{2} - |S \cap E|$ edges (for $\ell > 3$), and if $S \cap E = \emptyset$ then S cannot be a clique in G and hence induces fewer than $\binom{\ell}{2}$ edges.

Due to Remark 1, Min Node Blocker Independent Set and Min Node Blocker Vertex Cover are also strongly NP-hard. □

We are going to prove some approximation hardness results, too. In the reductions, the following problems will be used.

Dense k Subgraph
Input: An undirected graph $G = (V, E)$.
Output: A subset $V' \subseteq V$ of size k so as to maximize the number of edges whose both endpoints are in V'.

Max k Vertex Cover
Input: An undirected graph $G = (V, E)$.
Output: A subset $V' \subseteq V$ of size k so as to maximize the number of edges with at least one endpoint in V'.

MAX k VERTEX COVER-B is the version of MAX k VERTEX COVER where the maximum degree of the graph is at most B.

We extract the key points of the reductions in the following lemma on independent sets and vertex covers.

Lemma 2 *Let $G = (V, E)$ be a graph with n vertices and m edges, and let H be the bipartite incidence graph of G. Then the following properties are valid.*

(1a) *Suppose that G has maximum degree at most B, and the weights in H are $w_v = B + 1$ for all $v \in V$ and $w_e = 1$ for all $e \in E$. Then, for any $V' \subset V$ and any independent set S disjoint from V' in H, there exists an independent set S' such that $w(S') \geq w(S)$ and $S' \cap V = V \setminus V'$. Thus, if S' is maximal, then*

$$S' = (V \setminus V') \cup \{e \in E \mid e \subset V'\}$$

and, in particular, $\alpha(H - V') \geq (B + 1) \cdot (n - |V'|) + |\{e \in E \mid e \subset V'\}|$.

(1b) *Under the conditions of (1a), for any $V' \subset V \cup E$ with $|V'| < |V|$ there exists a $V'' \subset V$ such that $|V''| = |V'|$ and the maximum weight of an independent set in $H - V''$ is not larger than that in $H - V'$. As a consequence,*

$$\alpha(H - V') \geq \alpha(H - V'') = (B + 1) \cdot (n - |V''|) + |\{e \in E \mid e \subset V''\}|.$$

(2a) *Suppose that the weights in H are $w_v = d(v)$ (vertex degree) for all $v \in V$ and $w_e = 1$ for all $e \in E$. Then, for any $V' \subset V \cup E$, there exists a minimum-weight vertex cover $T \subseteq E$ in $H - V'$, namely $T = \{e \in E \setminus V' \mid e \not\subset V'\}$.*

(2b) *Beside the conditions of (2a), assume further that G is connected and contains at least one cycle. Then, for any $V' \subset V \cup E$ with $|V'| \leq |V|$ and $|V'|$ being at least as large as the shortest cycle length in G, there exists a $V'' \subset V$ such that $|V''| = |V'|$ and the minimum weight of a vertex cover in $H - V''$ is not larger than that in $H - V'$. As a consequence,*

$$\tau(H - V') \geq \tau(H - V'') = |\{e \in E \mid e \not\subset V''\}|.$$

Moreover, the sets V'' in both (1b) and (2b) can be found efficiently.

Proof: (1a) If S contains all vertices of $V \setminus V'$, then we have nothing to prove. Otherwise we modify S step by step, keeping it independent and not decreasing its value, until it contains the entire $V \setminus V'$. Hence, assume that $v \in V$ is a vertex such that $v \notin V' \cup S$. If v has no neighbor in $S \cap E$, then $S \cup \{v\}$ is a proper extension. Suppose that this is not the case; i.e., there is an edge $e \in E \cap S$ such that $v \in e$. We now modify S to $(S \setminus N_H(v)) \cup \{v\}$, where $N_H(v)$ denotes the set of vertices adjacent to v in H, that is the set of edges incident to v in G. In this way we have removed at most B neighbors of v from S, each of weight 1, and inserted v of weight $B + 1$, hence the total weight of the modified set is at least $w(S)$. Moreover, the set remains independent because all neighbors of v have been removed. Thus, after $|(V \setminus V') \setminus S|$ steps, the required set S' is obtained.

(1b) If $V' \subset V$, then $V'' = V'$ is a proper choice. Hence suppose $V' \cap E \neq \emptyset$. Let us introduce the notation $n' = |V' \cap V|$, $m' = |E(G[V' \cap V]) \setminus (V' \cap E)|$. By (1$a$) we see that

$\alpha(H-V') = (B+1)\cdot(n-n')+m'$ holds. Choose $e \in V' \cap E$ and $v \in V \setminus V'$, and modify V' to the set $(V' \setminus \{e\}) \cup \{v\}$. This keeps cardinality unchanged, while the first term $(B+1)\cdot(n-n')$ decreases by precisely $B+1$. Moreover, since G has maximum degree at most B, the second term can increase by at most B when we insert v into the set, and can further increase by at most 1 when we omit e. Thus, the sum does not increase. Repeatedly eliminating all $e \in E$ from V', the required V'' is obtained. Then (1*a*) implies that the independent set of maximum weight in $H - V''$ consists of all $v \notin V''$ and all $e \subset V''$.

(2*a*) Consider any vertex cover T of $H - V'$. Suppose $v \in T \cap V$ for some vertex $v \notin V'$. Remove v from T and insert the entire neighborhood $N_H(v) \setminus V'$ of v in H into T. Since $w_v = d(v) \geq |N_H(v) \setminus (V' \cup T)|$, this modification does not increase the weight of T. After at most $|V| - |V'|$ steps all vertices of V are eliminated from T.

(2*b*) Due to (2*a*), if $V' \subset E$, then $\tau(H - V') = |E| - |V'|$. In this case we can get an at least as good V'' by choosing a $|V'|$-element subset of V which induces a connected subgraph of G containing a shortest cycle. On the other hand, if $V' \subset V$, then we have nothing to prove. Hence, assume that $V' \cap V \neq \emptyset$ and $V' \cap E \neq \emptyset$.

If there is an edge $e = vv' \in V'$ such that $v \in V'$ and $v' \notin V'$, then we replace e with v' in V'. This modification keeps $|V'|$ unchanged, and it does not increase $\tau(H - V')$ because e is a subset of the modified V' and therefore it does not have to be put into a vertex cover of the new $H - V'$. If no such e exists but $V' \not\subset V$, we consider any $e' \in V' \cap E$. Since G is connected, there is a path from e' to $V' \cap V$, and its last edge say $e = vv'$ satisfies the conditions $e \in E \setminus V'$, $v \in V'$, $v' \notin V'$. Let us replace e' with e in V'. Then both $|V'|$ and $\tau(H-V')$ remain unchanged, and we are back to the previous situation where the replacement of e with v' maintains the conditions but decreases the size of $V' \cap E$. Hence, the repeated application of these operations eliminates all elements of V' from E. □

Theorem 3 *k Most Vital Nodes Independent Set has no ptas even for bipartite graphs if* P$\neq$ NP. *k Most Vital Nodes Vertex Cover has no ptas even for bipartite graphs if* NP $\not\subseteq \cap_{\delta>0}$ BPTIME (2^{n^δ}), *where* $\cap_{\delta>0}$ BPTIME (2^{n^δ}) *is the class of problems that admit randomized algorithms that run in time 2^{n^δ} for some constant $\delta > 0$.*

Proof: For both problems, we construct gap-preserving reductions. Throughout the proof, H denotes the bipartite incidence graph of the input graph $G = (V, E)$, the latter having n vertices and m edges.

k MOST VITAL NODES INDEPENDENT SET: We prove the non-existence of a ptas for $k = n/2$, constructing a gap-reduction from MAX $n/2$ VERTEX COVER-B to $n/2$ MOST VITAL NODES INDEPENDENT SET, where instances of the former are restricted to graphs G of maximum degree at most B. In Theorem 4 of [5], it is proved that there exists a constant $\rho > 1$ such that it is NP-hard to distinguish whether such a graph G has $opt(G) = m$ or $opt(G) < \frac{m}{\rho}$. In this case, let the vertices of H have weight $w_v = B + 1$ for all $v \in V$ and $w_e = 1$ for all $e \in E$.

Consider first the case $opt(G) = m$ and let V' be an optimum solution in G for MAX $n/2$ VERTEX COVER-B. Then removing $V \setminus V'$ from the vertex set of H, we obtain a graph in which the maximum weight of an independent set is $((B+1)/2)\cdot n$, as implied by part (1*a*) of Lemma 2. On the other hand, parts (1*a*) and (1*b*) together yield that after the removal of any $n/2$ vertices from H, there always remains an independent set of at least that large weight, thus $opt(H) = \frac{B+1}{2} \cdot n$.

Consider now the case $opt(G) < \frac{m}{\rho}$ and let V' be an optimum solution in G for Max $n/2$ Vertex Cover-B. Using part $(1a)$ of Lemma 2, when removing $V \setminus V'$ from the vertex set of H, we obtain a graph in which the maximum weight of an independent set is $((B+1)/2) \cdot n + m - opt(G)$. On the other hand, parts $(1a)$ and $(1b)$ together yield that after the removal of any $n/2$ vertices from H, there always remains an independent set of at least that large weight, thus $opt(H) = \frac{B+1}{2} \cdot n + m - opt(G) > \frac{B+1}{2} \cdot n + m - \frac{m}{\rho} \geq \frac{B+1}{2} \cdot n \cdot \rho'$, where $\rho' = 1 + \frac{2(1-1/\rho)}{B+1}$ since $m \geq n$.

<u>k Most Vital Nodes Vertex Cover</u>: We construct a gap-preserving reduction from the Dense k Subgraph problem. To achieve this goal, we first analyze which instances are hard for Dense k Subgraph. In Theorem 1.1 of [16] it is proved that the problem has no ptas in the range $k = \Theta(n)$ if $NP \nsubseteq \cap_{\delta>0} BPTIME\ (2^{n^\delta})$.

In general, the condition $k = \Theta(n)$ implies $opt(G) = \Theta(m)$ because for $c := k/n$ the selection of a k-element set at random induces an expected number of $\binom{k}{2}/\binom{n}{2} = (c^2 - o(1))\, m$ edges, which is a positive fraction of E. We observe further that non-approximability remains valid for instances restricted to *connected* input graphs containing at least one cycle of length 3. Indeed, let G' be obtained from G taking a new vertex w and inserting the new edges vw for all $v \in V$. We view G' as an instance of Dense $(k+1)$ Subgraph. Denote by opt the optimum value of G and by opt' the optimum value of G'. Clearly $opt' = opt + k$. Moreover, we may assume without loss of generality that a densest subgraph of G' contains w. Indeed, if the algorithm on G' finds a solution V' of value val' not containing w, then we remove a vertex which has minimum degree in the subgraph induced by V' and insert w into V'. This transformation (executable in linear time) does not decrease the number of edges inside V'. Then, restricting attention to the $(k+1)$-subgraphs containing w in G' they are in one-to-one correspondence with the k-subgraphs of G. This bijection yields $val = val' - k$.

Let $\varepsilon > 0$ be fixed, and suppose that an algorithm finds a solution on G' with value $val' \geq (1 - \frac{\varepsilon}{3})\, opt'$. Then the corresponding solution on G has value

$$val = val' - k \geq (1 - \frac{\varepsilon}{3})\, opt' - k = (opt' - k) - \frac{\varepsilon}{3}\, opt' = (1 - \frac{\frac{\varepsilon}{3}\, opt'}{opt})\, opt$$
$$= (1 - \frac{\frac{\varepsilon}{3}\,(opt + k)}{opt})\, opt = (1 - (\frac{\varepsilon}{3} + \frac{\varepsilon/3}{opt/k}))\, opt \geq (1 - \varepsilon)\, opt$$

because $opt \geq k/2$ (except for the rather trivial case where G is a matching and k is odd). Thus, a ptas on the connected instances of type G' would yield a ptas on general instances G. As a consequence, we may assume without loss of generality that all input graphs are connected and contain at least one cycle of length 3, hence making Lemma 2 $(2b)$ applicable.

Turning now to the gap-preserving reduction, let the vertices of the bipartite incidence graph H of G have weight $w_v = d(v)$ for $v \in V$ and $w_e = 1$ for $e \in E$. The case $k = n$ being trivial, we assume $k < n$ and hence $opt(G) < m$.

Consider first the case $opt(G) = v$ and let V' be an optimum solution in G for Dense k Subgraph, that is a set of k vertices that induces v edges. Then removing V' from the vertex set of H, we obtain a graph whose minimum-weight vertex cover is not larger than $|E \setminus E'|$, where E' is the edge set induced by V' in G, as implied by part $(2a)$ of Lemma 2. On the other hand, parts $(2a)$ and $(2b)$ together yield that after the removal of any subset of k vertices from H, there always remains a minimum-weight vertex cover of at least that large weight and thus $opt(H) = m - opt(G) = m - v$.

Consider now the case $opt(G) < \frac{v}{\rho}$ and let V' be an optimum solution in G for Dense k Subgraph. Using part $(2a)$ of Lemma 2, when removing V' from the vertex set of H, we obtain a graph whose minimum-weight vertex cover is not larger than $|E \setminus E'|$, where E' is the edge set induced by V' in G, and hence we have $opt(H) = \tau(H - V') = |\{e \in E \mid e \not\subset V'\}|$. On the other hand, parts $(2a)$ and $(2b)$ together yield that after the removal of any subset of k vertices from H, there always remains a minimum-weight vertex cover of at least that large weight and thus denoting $c' := v/m$ and $\rho' := \frac{1-c'/\rho}{1-c'}$ we obtain $opt(H) = m - opt(G) > m - \frac{v}{\rho} = \rho'(m - v)$. Here $c' < 1$ because $opt(G) < m$; moreover, as we noted at the beginning, $c' \geq c^2 - o(1) > 0$ and hence $\rho > 1$ implies $\rho' > 1$. □

4 Graph classes related to tree structures

In this section we consider graph classes representable over tree structures (trees, graphs of bounded treewidth, cographs), and prove that they admit algorithms solving the considered four problems in polynomial time. Efficient solvability for the graph classes in the first two subsections are implied by the results of the third subsection, too, but the methods for the former are simpler. The flavor of our algorithm for graphs of bounded treewidth is similar to that of the one in [23], which solves related problems on maximum matchings in pseudo-polynomial time and is, to our best knowledge, the first work applying dynamic programming for node/edge interdiction. The matching interdiction problem in the particular class of trees with its dynamic programming approach was also studied in [20].

4.1 Trees

Theorem 4 *k Most Vital Nodes Independent Set and k Most Vital Nodes Vertex Cover are polynomial on trees. On trees of order n the problems can be solved in $O(nk^2)$ time, for any $k \geq 1$.*

Proof: Our general approach is to find not only a set of k most vital nodes but simultaneously also the value of a corresponding largest independent set or smallest vertex cover. For this purpose we view the input as a *rooted* tree with an arbitrarily chosen root, and organize computation according to a postorder traversal.

Consider any tree T with vertices $v_1, \ldots, v_n$. Each vertex v_i can have three positions in a solution, that we shall denote by marks $+, -, 0$ as follows:

- '$+$' means that v_i is selected into an independent set or a vertex cover;
- '$-$' means that v_i is selected for deletion;
- '0' means that v_i is none of the above two types.

In a solution exactly k marks '$-$' have to occur.

The subtree rooted in v_i is denoted by T_i. For each $i = 1, \ldots, n$, each $* \in \{+, -, 0\}$, and each $j = 0, 1, \ldots, k$, a value $z_i(j, *)$ will be computed. This $z_i(j, *)$ represents the minimum achievable weight of a solution (largest independent set or smallest vertex cover) on T_i under the conditions that *exactly* j vertices are removed from T_i and v_i has mark $*$. For the recursive computation the children of v_i with degree d will be denoted by $v_{i_1}, \ldots, v_{i_d}$. We traverse T in postorder and apply dynamic programming.

Recursion for Independent Set. If v_i is marked '$+$', then all its children must have '$-$' or '0', since otherwise two vertices selected for the independent set would be adjacent. Moreover, $z_i(j,*)$ requires that the total number of vertices marked '$-$' should be exactly j in T_i. On the other hand, we have one and only one way to make the final result as small as possible: decide which of the vertices should be marked with '$-$'. Once this has been decided, the distribution of '$+$' and '0' positions aims at maximizing the total weight of '$+$'. This leads to the following general recursions:

$$z_i(j,+) = w_i + \min_{\substack{j_1,\dots,j_d \geq 0 \\ j_1+\dots+j_d=j}} \sum_{\ell=1}^{d} \min\left(z_{i_\ell}(j_\ell,-), z_{i_\ell}(j_\ell,0)\right),$$

$$z_i(j,-) = \min_{\substack{j_1,\dots,j_d \geq 0 \\ j_1+\dots+j_d=j-1}} \sum_{\ell=1}^{d} \min\left(z_{i_\ell}(j_\ell,-),\, \max\left(z_{i_\ell}(j_\ell,+), z_{i_\ell}(j_\ell,0)\right)\right),$$

$$z_i(j,0) = \min_{\substack{j_1,\dots,j_d \geq 0 \\ j_1+\dots+j_d=j}} \sum_{\ell=1}^{d} \min\left(z_{i_\ell}(j_\ell,-),\, \max\left(z_{i_\ell}(j_\ell,+), z_{i_\ell}(j_\ell,0)\right)\right),$$

For a leaf v_i we clearly have $z_i(0,+) = w_i$ and $z_i(1,-) = z_i(0,0) = 0$. Further, to indicate that all other combinations of $j \in \{0,1,\dots,k\}$ and $* \in \{+,-,0\}$ are infeasible, we set a dummy symbol $z_i(j,*) = \mathsf{NIL}$ for them. In the recursive step, terms with value NIL on the right-hand side are neglected, except when all terms are the same, and in this case we define $z_i(j,*) = \mathsf{NIL}$, too.

Recursion for Vertex Cover. If v_i is marked '0', then all its children must have '$+$' or '$-$', because no edge must have both endpoints marked with '0'. Further, we again need for $z_i(j,*)$ that the total number of vertices marked '$-$' should be exactly j in T_i. The recursive step is simpler than above, however, because τ is defined to be minimum, what matches the goal of the 'most vital nodes' problem. Hence, we now have:

$$z_i(j,+) = w_i + \min_{\substack{j_1,\dots,j_d \geq 0 \\ j_1+\dots+j_d=j}} \sum_{\ell=1}^{d} \min\left(z_{i_\ell}(j_\ell,+), z_{i_\ell}(j_\ell,-), z_{i_\ell}(j_\ell,0)\right),$$

$$z_i(j,-) = \min_{\substack{j_1,\dots,j_d \geq 0 \\ j_1+\dots+j_d=j-1}} \sum_{\ell=1}^{d} \min\left(z_{i_\ell}(j_\ell,+), z_{i_\ell}(j_\ell,-), z_{i_\ell}(j_\ell,0)\right),$$

$$z_i(j,0) = \min_{\substack{j_1,\dots,j_d \geq 0 \\ j_1+\dots+j_d=j}} \sum_{\ell=1}^{d} \min\left(z_{i_\ell}(j_\ell,+), z_{i_\ell}(j_\ell,-)\right).$$

Also here, for a leaf v_i we have $z_i(0,+) = w_i$ and $z_i(1,-) = z_i(0,0) = 0$. Now, as an alternative of NIL, it is equally fine to set $z_i(j,*) = +\infty$ for the other combinations of $j \in \{0,1,\dots,k\}$ and $* \in \{+,-,0\}$.

Finding an optimal solution. Assuming that T has root v_{i_0}, after the removal of k properly chosen vertices, the smallest possible value of τ is just $\min_{*\in\{+,-,0\}} z_{i_0}(k,*)$; whereas for α it is $\min\left(z_{i_0}(k,-),\, \max\left(z_{i_0}(k,+), z_{i_0}(k,0)\right)\right)$. (In fact, inserting a new vertex v_0 with weight $w_0 = 0$ as new root and parent for v_{i_0} does not change the optimum, and then we would have

$z_0(k,+) \leq opt = z_0(k,0) \leq z_0(k,-)$ for Independent Set.) A set of k most vital nodes can also be determined in $O(n)$ additional steps in the following way. At the recursive step for each $z_i(j,*)$ we register for each edge $v_i v_{i_\ell}$ the corresponding value of j_ℓ in the optimal distribution $(j_1,\dots,j_d)$ for j and also the mark $* \in \{+,-,0\}$ of i_ℓ which gave the optimum for v_i. Once these data are available for all v_i and all pairs $(j,*)$, we can traverse T in preorder and select the vertices having '$-$' mark for the most vital set.

Efficient implementation. The key point is to find in polynomial time a best distribution $(j_1,\dots,j_d)$ for the 'max' and 'min' functions acting on the sums. This can be done, despite that the number of possibilities can even be exponential if d is proportional to n.

If $d = 2$ then we have at most $j+1$ combinations of feasible pairs j_1, j_2. Hence, optimal choice can be made in $O(k)$ steps for any one particular j, and in $O(k^2)$ steps for all $0 \leq j \leq k$. If d is larger, we can split the children of v_i into two sets of (nearly) equal size, $\{v_\ell \mid 1 \leq \ell \leq \lfloor d/2 \rfloor\}$ and $\{v_\ell \mid \lfloor d/2 \rfloor + 1 \leq \ell \leq d\}$, make all computation separately for each of them, and then combine the results for v_i. (Splitting corresponds to inserting a 'supernode' above each of the two sets, which has weight zero and becomes a virtual child of v_i.) This requires $d-1$ rounds for v_i. Since T is a tree, those $d-1$ sum up to $n-2$, thus the overall running time is $O((k^2+1)n)$, and never exceeds $O(n^3)$. (Here '+1' is needed for $k=0$.) Note that there are no 'hidden large constants' in the 'O' notation. $\square$

Theorem 5 Min Node Blocker Independent Set *and* Min Node Blocker Vertex Cover *are polynomial on trees. On trees of order n the problems can be solved in $O(n^3 \log n)$ time.*

Proof: The above algorithm in one iteration for any $1 \leq v \leq n$ runs in $O(v^2 n) = O(n^3)$ time. Hence, using Lemma 1, finding the smallest k for which the solution has value at most U takes total running time $O(n^3 \log n)$. $\square$

Remark 4 The algorithms proposed in Theorem 4 solve the k Most Vital Nodes Independent Set and k Most Vital Nodes Vertex Cover problems on *paths* in $O(kn)$ time. In fact, in the general time bound $O(nk^2)$ for trees, the factor k^2 occurs due to the presence of vertices with more than one child. This observation implies further that the algorithms proposed in Theorem 5 solve Min Node Blocker Independent Set and Min Node Blocker Vertex Cover on paths in $O(n^2 \log n)$ time.

4.2 Cycles

Theorem 6 *k* Most Vital Nodes Independent Set *and k* Most Vital Nodes Vertex Cover *are polynomial on cycles. On cycles of order n the problems can be solved in $O(kn^2)$ time, for any $k \geq 1$.*

Proof: Let $S^* = \{v_1,\dots,v_r\} \subset V$ be a maximum-weight independent set of a given cycle $C = (V,E)$. An optimal solution $V' \subset V$ of k Most Vital Nodes Independent Set must contain at least one node of S^*, since otherwise $\alpha(C - V')$ is not smaller than $\alpha(C)$. Thus, for each $v_j \in S^*$, $j = 1,\dots,r$, we determine the $k-1$ further nodes to remove in the resulting path as follows. We delete v_j from C and determine a maximum-weight independent set in the resulting path $C - v_j$ by applying the algorithm given in Theorem 4 in order to find an optimal solution $R^*_j \subset V \setminus \{v_j\}$ of $k-1$ Most Vital Nodes Independent Set on the path

$C - v_j$. Then, an optimal solution for k Most Vital Nodes Independent Set on C is $R^*_\ell \cup \{v_\ell\}$ such that $\alpha(C - v_\ell - R^*_\ell) = \min_{1 \le j \le r} \alpha(C - v_j - R^*_j)$. If the root is chosen to be an endpoint of the path, the complexity of the algorithm given in Theorem 4 for path $C - v_j$ is $O(kn)$. Since $|S^*| \le n$, in this way k Most Vital Nodes Independent Set is solved in $O(kn^2)$.

The proof for k Most Vital Nodes Vertex Cover is similar. □

Theorem 7 Min Node Blocker Independent Set *and* Min Node Blocker Vertex Cover *are polynomial on cycles. On cycles of order n the problems can be solved in $O(n^3 \log n)$ time.*

Proof: The theorem follows from Theorem 6 and Lemma 1. □

4.3 Graphs of bounded treewidth

A *tree decomposition* of a graph $G = (V, E)$ without isolated vertices is a pair $(T, \mathcal{X})$ where

- $T = (X, F)$ is a tree graph with a set $X = \{x_1, \dots, x_m\}$ of *nodes* and a set F of *lines*;
- $\mathcal{X} = \{X_1, \dots, X_m\}$ is a set system over V (i.e., over the vertex set of G), where each X_q is associated with node x_q of T;
- each edge $v_i v_j \in E$ of G is contained in at least one X_q for some $1 \le q \le m$;
- for any $v_i \in V$, if $v_i \in X_{q'}$ and $v_i \in X_{q''}$, then $v_i \in X_q$ for all q such that x_q lies on the $x_{q'}$–$x_{q''}$ path in T.

The width of $(T, \mathcal{X})$ is $\max\limits_{1 \le q \le m} |X_q| - 1$, and the *treewidth* of G, denoted by $tw(G)$, is the smallest integer t for which G admits a tree decomposition of width t. For undefined details on tree decomposition we refer to [17].

Theorem 8 k Most Vital Nodes Independent Set *and* k Most Vital Nodes Vertex Cover *are polynomial on bounded treewidth graphs. On graphs of order n the problems can be solved in $O(nk^2)$ time for any $k \ge 1$.*

Proof: Suppose that we wish to solve the problems on graphs of treewidth at most $t - 1$. Hence, assume that G has treewidth *less than* t, and let $(T, \mathcal{X})$ be a tree decomposition of G, such that $|X_q| \le t$ holds for all $1 \le q \le m$. We view T as a *rooted tree*, by choosing an arbitrary node as root. The choice of the root generates parent-child relation between nodes in the usual way. Using standard terminology in a slightly stricter (but still wide-spread) way, we say that the tree decomposition $(T, \mathcal{X})$ is a *nice tree decomposition* if it has only four types of nodes, as follows:

- a *start* node x_q that has no children (a leaf in T), with $|X_q| = 1$;
- a *join* node x_q that has two children $x_{q'}, x_{q''}$, with $X_q = X_{q'} = X_{q''}$;
- an *introduce* node x_q that has one child $x_{q'}$, with $X_q = X_{q'} \cup \{v\}$ for some $v \in V$;
- a *forget* node x_q that has one child $x_{q'}$, with $X_q = X_{q'} \setminus \{v\}$ for some $v \in V(G)$.

As is well known, a nice tree decomposition of size $O(n)$ and of minimum width can be found in linear time for graphs of bounded treewidth [6, 17]. Hence, we may assume without loss of generality that T is a *nice tree decomposition of width less than t* for G. We are going to show how $\alpha(k)$ and $\tau(k)$ can be determined using dynamic programming. The general frame is the same for both problems, only the details of computation will be different.

Let T_q denote the subtree of T rooted in x_q, for $1 \le q \le m$. Over the nodes of T_q we set $V_q = \bigcup_{x_{q'} \in V(T_q)} X_{q'}$, and denote by G_q the subgraph induced by V_q in G. Hence, if x_q is a *join* node with children $x_{q'}$ and $x_{q''}$, then $V_{q'} \cap V_{q''} = X_q$ holds, and there are no edges between $V_{q'} \setminus X_q$ and $V_{q''} \setminus X_q$ in G.

At each $x_q \in X$ we construct a matrix M_q that represents the traces inside X_q for all possible decisions with respect to the problem solution. This M_q has $k+1$ columns corresponding to the number $j = 0, 1, \ldots, k$ of vertices removed from G_q in a solution, and $3^{|X_q|}$ rows representing the partitions $Z_+ \cup Z_- \cup Z_0 = X_q$ into three disjoint labeled sets.

Each row of M_q can be associated with a sequence $\boldsymbol{r} \in \{+,-,0\}^{|X_q|}$, where the ith term indicates whether the ith vertex of X_q belongs to the independent set to be selected $(+)$, or is to be removed from G $(-)$, or neither of these (0). Hence, for $* \in \{+,-,0\}^{|X_q|}$, the occurrences of $*$ in $\boldsymbol{r}$ represent the characteristic vector of Z_*. We shall denote by $|\boldsymbol{r}_-|$ the number of '$-$' components in row $\boldsymbol{r}$. For $0 \le j \le k$ the jth entry of $\boldsymbol{r}$ in M_q, which we shall denote by $z_q(\boldsymbol{r}, j)$, is the optimum value of a solution in G_q that meets the conditions expressed in $\boldsymbol{r}$. If a combination of conditions is infeasible (e.g., there are fewer than j vertices in G_q, or two vertices associated with '$+$' in $\boldsymbol{r}$ are adjacent in G) then we assign the dummy symbol $z_q(\boldsymbol{r}, j) = \mathsf{NIL}$.

The computation of $z_q(\boldsymbol{r}, j)$ is problem specific, we give the details next. The way of finding the final solutions will be described afterwards.

Recursion for Independent Set. For a *start* node, M_q is a $3 \times (k+1)$ matrix. Assuming $X_q = \{v_i\}$, vertex v_i counts with weight w_i if it is selected into the independent set and counts 0 otherwise. Hence we have $z_q(+,0) = w_i$, $z_q(-,1) = z_q(0,0) = 0$, and $z_q(*,j) = \mathsf{NIL}$ for any other combination of $* \in \{+,-,0\}$ and $j \in \{0, 1, \ldots, k\}$.

If x_q is a *join* node with children $x_{q'}, x_{q''}$, then row $\boldsymbol{r}$ of M_q has to be composed from the rows belonging to the same $\boldsymbol{r}$ in $M_{q'}$ and $M_{q''}$. Since the sets $Z_+, Z_- \subseteq X_q$ appear in both $G_{q'}$ and $G_{q''}$, we see that $j_{q'}$ resp. $j_{q''}$ vertices deleted from $G_{q'}$ resp. $G_{q''}$ mean $j_{q'} + j_{q''} - |\boldsymbol{r}_-|$ deleted ones for G_q. An optimal solution for G_q is obtained from the best possible combination of $G_{q'}$ and $G_{q''}$; that is,

$$z_q(\boldsymbol{r}, j) = \min_{\substack{j_{q'}, j_{q''} \ge |\boldsymbol{r}_-| \\ j_{q'} + j_{q''} = j + |\boldsymbol{r}_-|}} \left(z_{q'}(\boldsymbol{r}, j_{q'}) + z_{q''}(\boldsymbol{r}, j_{q''}) \right) - \sum_{v_i \in Z_+} w_i .$$

If x_q is an *introduce* node with $X_q = X_{q'} \cup \{v\}$, there are three possible decisions concerning v; and if v is selected for the independent set, then none of its neighbors can be selected. Hence, if $\boldsymbol{r'}$ denotes the sequence obtained by deleting the v-component from $\boldsymbol{r}$, then the three cases yield the following recursions:

- The v-component is $+$ $\Rightarrow$ $z_q(\boldsymbol{r}, j) = z_{q'}(\boldsymbol{r'}, j) + w_i$ if v is not adjacent to any vertex of Z_+, and $z_q(\boldsymbol{r}, j) = \mathsf{NIL}$ otherwise.
- The v-component is $-$ $\Rightarrow$ $z_q(\boldsymbol{r}, j) = z_{q'}(\boldsymbol{r'}, j-1)$ for $j \ge 1$; $z_q(\boldsymbol{r}, 0) = \mathsf{NIL}$.
- The v-component is 0 $\Rightarrow$ $z_q(\boldsymbol{r}, j) = z_{q'}(\boldsymbol{r'}, j)$.

Finally, if x_q is a *forget* node and its child is associated with the set $X_q = X_{q'} \setminus \{v\}$, then $\boldsymbol{r}$ is obtained from some $\boldsymbol{r}'$ of $M_{q'}$ by deleting its v-component, where the deleted component can be any $* \in \{+,-,0\}$. Let us denote the corresponding row by $\boldsymbol{r}'_*$. While searching for most vital nodes, we may decide whether or not the v-component should be '$-$' but we cannot make any decision between '+' and '0'. Thus, the smallest possible weight of a maximum independent set is obtained by

$$z_q(\boldsymbol{r}, j) = \min\left(z_q(\boldsymbol{r}'_-, j), \max\left(z_q(\boldsymbol{r}'_+, j), z_q(\boldsymbol{r}'_0, j)\right)\right).$$

Recursion for Vertex Cover. Since the approach is similar to the one given above, we describe the method here in less detail. In the present case '+' means that the corresponding vertex is selected into a vertex cover. The union of the sets Z_+ has to meet all edges after the removal of all Z_- from G. Since each edge is a subset of at least one X_q, a necessary and sufficient condition for this property is that the sets Z_0 must be independent for each X_q. It will be enough to check this property at the *introduce* nodes.

For a *start* node with $X_q = \{v_i\}$, we have $z_q(+,0) = w_i$, $z_q(-,1) = z_q(0,0) = 0$, and $z_q(*,j) = \mathsf{NIL}$ for any other combination of $* \in \{+,-,0\}$ and $j \in \{0,1,\dots,k\}$.

If x_q is a *join* node with children $x_{q'}, x_{q''}$, then no vertex of $V_{q'} \setminus X_q$ is contained in any edge meeting $V_{q''} \setminus X_q$, and vice versa. Thus, vertex covers in G_q are the unions of those in $G_{q'}$ and $G_{q''}$, therefore we have

$$z_q(\boldsymbol{r}, j) = \min_{\substack{j_{q'}, j_{q''} \geq |\boldsymbol{r}_-| \\ j_{q'} + j_{q''} = j + |\boldsymbol{r}_-|}} \left(z_{q'}(\boldsymbol{r}, j_{q'}) + z_{q''}(\boldsymbol{r}, j_{q''})\right) - \sum_{v_i \in Z_+} w_i.$$

If x_q is an *introduce* node with $X_q = X_{q'} \cup \{v\}$, vertex v may belong to Z_+, Z_-, or Z_0; and in the third case if v has a neighbor in Z_0, then the selection is not feasible for vertex cover. Let $\boldsymbol{r}'$ denote the sequence obtained by deleting the v-component from $\boldsymbol{r}$. Depending on the position of v, we have the following rules for the recursion:

- The v-component is $+ \;\Rightarrow\; z_q(\boldsymbol{r}, j) = z_{q'}(\boldsymbol{r}', j) + w_i$.
- The v-component is $- \;\Rightarrow\; z_q(\boldsymbol{r}, j) = z_{q'}(\boldsymbol{r}', j-1)$ for $j \geq 1$; $z_q(\boldsymbol{r}, 0) = \mathsf{NIL}$.
- The v-component is $0 \;\Rightarrow\; z_q(\boldsymbol{r}, j) = z_{q'}(\boldsymbol{r}', j)$ if v is not adjacent to any vertex of Z_0, and $z_q(\boldsymbol{r}, j) = \mathsf{NIL}$ otherwise.

Finally, if x_q is a *forget* node and its child is associated with the set $X_q = X_{q'} \setminus \{v\}$, then $\boldsymbol{r}$ is obtained from some $\boldsymbol{r}'$ of $M_{q'}$ by deleting its v-component, where the deleted component can be any $* \in \{+,-,0\}$. Denoting the corresponding row of $M_{q'}$ by $\boldsymbol{r}'_*$, the best local choice is:

$$z_q(\boldsymbol{r}, j) = \min_{* \in \{+,-,0\}} z_q(\boldsymbol{r}'_*, j).$$

Finding an optimal solution. For any of the two problems, assume that the matrices M_q have been determined for all nodes x_q, and let the root of T be x_{q_0}. Then the optimal value for k Most Vital Nodes Vertex Cover is simply $z_{q_0}(\boldsymbol{r}_0, k)$, where row $\boldsymbol{r}_0$ attains minimum in the last column of M_{q_0}. But the situation for k Most Vital Nodes Independent Set is more complicated. With respect to the most vital set, the rows of M_{q_0} can be classified according to the positions of their '$-$' components. In this way we have $\sum_{i=0}^{k} \binom{|X_{q_0}|}{i}$ classes (where i represents the number of '$-$'). We have no influence on the 0/+ distribution; the only detail we can decide is the position of the '$-$' marks; that is, from which class we choose the solution. Once the class is fixed, under this constraint the solution would be the maximum taken over all 0/+ distributions, let us call this the value of the class. Then the overall optimum of the problem is the minimum value taken over all classes.

An optimal set of k vertices can also be constructed if we do a little more bookkeeping during the recursive steps. For each triple $(q, \boldsymbol{r}, j)$ we store the relevant pointer(s) showing which entry (entries) of the child(ren) have given the value of $z_q(\boldsymbol{r}, j)$ in the recursion. Then, starting from $(q_0, \boldsymbol{r}_0, k)$ we can trace all relevant triples $(q_\ell, \boldsymbol{r}_\ell, j_\ell)$ which have contributed to the composition of $z_{q_0}(\boldsymbol{r}_0, k)$. A most vital k-set is obtained by the union of the sets Z_- belonging to those sequences $\boldsymbol{r}_\ell$. This top-down (partial preorder) traversal needs only $O(n)$ additional steps. Indeed, the union of the Z_- can be gathered while moving from the *forget* nodes to their children, adding the corresponding vertex v to the most vital set if $v \in Z_-$ in the actual $X_{q'}$.

Time analysis. To compute one entry of M_q we need constant time for *start*, *introduce* and *forget* nodes. This also includes the side conditions on *introduce* nodes, because nonadjacency of the new vertex has to be checked[1] with respect to fewer than t other vertices of Z_+ or Z_-. Hence, the most time-consuming case of the recursion occurs at the *join* nodes. For a particular $\boldsymbol{r}$, we have $0 \le |\boldsymbol{r}_-| \le j' \le j$; i.e., minimum or maximum has to be selected from at most $j+1$ possibilities, which takes at most j comparisons. Here j ranges from 0 to k, therefore the computation of an entire row requires at most $(k+1)^2$ steps. There are at most 3^t rows in any M_q, which is constant whenever treewidth is bounded; and the number of matrices to be computed is $O(n)$. Consequently, the total number of steps needed is $O(nk^2) = O(n^3)$ because $k \le n$ holds in both problems. Traversing T needs as few as $O(n)$ additional steps. □

Theorem 9 Min Node Blocker Independent Set *and* Min Node Blocker Vertex Cover *are polynomial on bounded treewidth graphs. On graphs of order n the problems can be solved in $O(n^3 \log n)$ time.*

Proof: The theorem follows from Theorem 8 and Lemma 1. □

4.4 Cographs

To each cograph G with n vertices, we can associate a rooted tree T, called the *cotree* of G. Leaves of T correspond to vertices of the graph G and internal nodes of T are labeled with either '$\cup$' (union-node) or '$\times$' (join-node). A subtree rooted at node '$\cup$' corresponds to the union of the subgraphs defined by the children of that node, and a subtree rooted at node '$\times$' corresponds to the join of the subgraphs defined by the children of that node; that is, we add an edge between every two vertices corresponding to leaves in different subtrees. Cographs can be recognized in linear time and the cotree representation can be obtained efficiently [7, 12]. Moreover, this cotree can easily be transformed in linear time to a binary cotree with $O(n)$ nodes.

Theorem 10 k Most Vital Nodes Independent Set *and* k Most Vital Nodes Vertex Cover *are polynomial on cographs. On cographs of order n,* k Most Vital Nodes

[1]Each check can be done in constant time if adjacency matrix is used with direct addressing. This requires $O(n^2)$ space.

Independent Set *can be solved in* $O(nk^2)$ *time and* k Most Vital Nodes Vertex Cover *can be solved in* $O(n^2 + nk^2)$ *time, for any* $k \geq 1$.

Proof: Consider a cograph G with n vertices $v_1, \ldots, v_n$. Given a binary cotree representation T of G, we show in the following how to solve the k Most Vital Nodes Independent Set and k Most Vital Nodes Vertex Cover using dynamic programming.

Let $x_1, \ldots, x_t$ be the nodes of T where x_r is its root and t is in $O(n)$. For $i = 1, \ldots, t$, denote by T_i the subtree rooted at x_i, G_i the subgraph induced by the vertices corresponding to the leaves of T_i, and V_i these vertices.

Recursion for Independent Set. We associate a $(k+1)$-vector to each node x_i of T, $i = 1, \ldots, t$. In the following, a $(k+1)$*-vector* is simply called a vector. For each i and each $j = 0, 1, \ldots, k$, we compute $z_i(j)$ that is the minimum weight of a maximum independent set on G_i where exactly j vertices are removed from G_i. These vectors are computed 'bottom-up' in the cotree. So, we start by computing vectors of leaves and after that the vector of an internal node if the vectors of its two children are already computed.

Given a node x_i of the cotree, the corresponding vector is obtained as follows:

- If x_i is a union-node with two children x_ℓ and x_r, we have no edges between G_ℓ and G_r. Then the maximum independent set in G_i is the union of those in G_ℓ and G_r. Thus, since we want to find a maximum-weight independent set as small as possible, the best choice is given by $z_i(j) = \min_{j_1+j_2=j} (z_\ell(j_1) + z_r(j_2))$.

- If x_i is a join-node with two children x_ℓ and x_r, every vertex in V_ℓ is adjacent to every vertex in V_r. Then each independent set in G_i is entirely contained either in G_ℓ or in G_r. So, $z_i(j) = \min_{j_1+j_2=j} (\max(z_\ell(j_1), z_r(j_2)))$.

- If x_i is a leaf then $z_i(0) = w_i$, $z_i(1) = 0$, and $z_i(j) = \mathsf{NIL}$ for $j = 2, \ldots, k$ which means that the latter configurations are infeasible. In the recursive step, terms with value NIL on the right-hand side are neglected, except when all terms are the same, and in this case we define $z_i(j) = \mathsf{NIL}$, too.

Recursion for Vertex Cover. The approach is similar to the previous one. We associate a vector to each node x_i of T, $i = 1, \ldots, t$. For each i and each $j = 0, 1, \ldots, k$, a value $z_i(j)$ and a subset $S_i(j)$ are computed. Here $z_i(j)$ means the minimum weight of a vertex cover of G_i where exactly j vertices are removed from G_i, and $S_i(j)$ is the subset of vertices that are neither included in the vertex cover of G_i nor are removed from G_i.

Given a node x_i of the cotree, the corresponding vector is obtained as follows:

- If x_i is a union-node with two children x_ℓ and x_r , we have no edges between G_ℓ and G_r. Then the minimum vertex cover in G_i is the union of those in G_ℓ and G_r. Thus, since we want to find a minimum-weight vertex cover as small as possible, the best choice is given by $z_i(j) = \min_{j_1+j_2=j} (z_\ell(j_1) + z_r(j_2))$ and $S_i(j) = S_\ell(j_1^*) \cup S_r(j_2^*)$ where j_1^* and j_2^* are the indices that realize the minimum for $z_i(j)$. If we have many j_1^* and j_2^*, we choose the one with the smallest $\sum_{v_s \in S_\ell(j_1^*) \cup S_r(j_2^*)} w_s$.

- If x_i is a join-node with two children x_ℓ and x_r then a vertex cover in G_i has to contain all non-removed vertices in one of V_ℓ or V_r, and also a vertex cover of the non-removed subgraph in the other part. Once we decide which part is completely included as removal and cover, the best way to select its given number j' of removed vertices is to delete the j' vertices of largest weights of that part. Assuming that j_1 vertices are removed from V_ℓ and j_2 are removed from V_r, we denote by $s_\ell(j_1)$ and $s_r(j_2)$ the minimum sum of weights of the remaining $|V_\ell| - j_1$ and $|V_r| - j_2$ vertices, respectively. That is, $s_\ell(j_1) = w(V_\ell) - \max\limits_{Y \subset V_\ell,\, |Y| = j_1} w(Y)$, and $s_r(j_2)$ is defined analogously. If $j_1 > |V_\ell|$ or $j_2 > |V_r|$, the value of s is defined to be $+\infty$. Then we have

$$z_i(j) = \min_{j_1 + j_2 = j} \min\left(s_\ell(j_1) + z_r(j_2), s_r(j_2) + z_\ell(j_1)\right)$$

and $S_i(j) = S_\ell(j_1)$ or $S_i(j) = S_r(j_2)$, depending on whether the minimum for $z_i(j)$ has been obtained from $z_\ell(j_1)$ or $z_r(j_2)$.

- If x_i is a leaf then $z_i(0) = z_i(1) = 0$, $z_i(j) = +\infty$ for $j = 2, \ldots, k$, $S_i(0) = \{v_i\}$ and $S_i(j) = \emptyset$ for $j = 1, \ldots, k$.

Finding an optimal solution. For each of the two problems, an optimal solution is obtained at the root x_r of T and its weight is equal to $z_r(k)$. Moreover, an optimal set of k removed vertices can be computed step by step in the recursion. Indeed, let $S_i^-(j)$ be the subset of j removed vertices in G_i. For a leaf x_i we have $S_i^-(0) = \emptyset$, $S_i^-(1) = \{v_i\}$ and $S_i^-(j) = \emptyset$ for $j = 2, \ldots, k$. For a union-node or a join-node x_i with two children x_ℓ and x_r, recursion yields $S_i^-(j) = S_\ell^-(j_1^*) \cup S_r^-(j_2^*)$ where j_1^* and j_2^* are the indices that realize the minimum for $z_i(j)$.

Time analysis. For k Most Vital Nodes Independent Set, vectors are computed in $O(k)$ for each leaf and in $O(k^2)$ for each union-node and each join-node. Since $t = O(n)$, the algorithm runs in $O(nk^2)$.

For k Most Vital Nodes Vertex Cover, the computation of vector for a leaf takes $O(k)$ time. For a union-node and a join-node, we have to compare and select a minimum value from at most $j + 1$ possibilities and determine a subset of vertices which attains this minimum. Note that at most k vertices of largest weight are relevant for s. For leaves of the cotree this is just one element and can be viewed to be in decreasing order of weight; and then for any union- or join-node the (at most) k largest elements can be selected in $O(k)$ time from the lists of the children using merge sort and keeping the decreasing order. Since $\sum_{v_s \in S_i(j)} w_s$ and $S_i(j)$ are obtained in $O(n)$ for any given i and j, the computation of vector corresponding to an internal node takes $O(k^2 + n)$. Therefore, the algorithm runs in $O(n^2 + nk^2)$. Speed-up for the sets $S_i(j)$ can also be made if we do not explicitly list them at each node but only store their values and the pointers to the children from which they have been obtained. □

Theorem 11 Min Node Blocker Independent Set *and* Min Node Blocker Vertex Cover *are polynomial on cographs. On cographs of order n the problems can be solved in $O(n^3 \log n)$ time.*

Proof: The theorem follows from Theorem 10 and Lemma 1. □

5 Conclusion

In this paper we studied the complexity of the k most vital nodes and min node blocker versions of the maximum-weight independent set and minimum-weight vertex cover problems. While maximum-weight independent set and minimum-weight vertex cover are polynomial on bipartite graphs, the k most vital nodes and min node blocker versions become NP-hard, and we also proved that most vital nodes have no ptas. We obtained further that for $k > 0$ the complementarity of maximum independent sets and minimum vertex covers does not remain valid.

An interesting perspective for future research is to study the complexity of the k most vital nodes and min node blocker versions of the maximum-weight independent set problem for graphs of bounded cliquewidth [9] and graphs of bounded NLC-width [21], that generalize cographs. Moreover, it would be worth studying the complexity and approximation of these versions on further classes of graphs for which maximum-weight independent set and minimum-weight vertex cover are polynomial.

We close the paper with some explicitly stated problems.

Conjecture 1 *The problems* MIN NODE BLOCKER INDEPENDENT SET *and* MIN NODE BLOCKER VERTEX COVER *have no ptas on bipartite graphs.*

Problem 1 *Are the problems k* MOST VITAL NODES INDEPENDENT SET*, k* MOST VITAL NODES VERTEX COVER*,* MIN NODE BLOCKER INDEPENDENT SET *and* MIN NODE BLOCKER VERTEX COVER *solvable in polynomial time on graphs of bounded cliquewidth?*

Acknowledgement. We are deeply indebted to Rico Zenklusen for several important insightful comments on a previous version of this paper, and also for pointing out gaps in the proofs of some assertions announced in the conference contribution [3].

References

[1] A. Bar-Noy, S. Khuller, and B. Schieber, The complexity of finding most vital arcs and nodes, *Technical Report CS-TR-3539, Department of Computer Science, University of Maryland*, 1995.

[2] C. Bazgan, S. Toubaline, and D. Vanderpooten, Détermination des éléments les plus vitaux pour le problème d'affectation, *Actes de la 10ème Conférence de la Société Française de Recherche Opérationelle et d'Aide à la Décision (ROADEF 2010)*, 2010.

[3] C. Bazgan, S. Toubaline, and Zs. Tuza, Complexity of most vital nodes for Independent Set in graphs related to tree structures, *Proceedings of the 21st International Workshop on Combinatorial Algorithms (IWOCA 2010)*, LNCS 6460, 154–166, 2011.

[4] C. Bazgan, S. Toubaline, and D. Vanderpooten, Complexity of determining the most vital elements for the 1-median and 1-center location problems, *Proceedings of the 4th Annual International Conference on Combinatorial Optimization and Applications (COCOA 2010)*, LNCS 6508, part I, 237–251, 2010.

[5] C. Bazgan, Zs. Tuza and D. Vanderpooten, Approximation of satisfactory bisection problems, *Journal of Computer and System Sciences*, 74(5), 875–883, 2008

[6] H. L. Bodlaender, A linear time algorithm for finding tree-decompositions of small treewidth, *SIAM Journal on Computing*, 25(6), 1305–1317, 1996.

[7] D. G. Corneil, Y. Perl, and L.K. Stewart, A linear recognition algorithm for cographs, *SIAM Journal on Computing*, 14(4), 926–934, 1985.

[8] M.-C. Costa, D. de Werra, and C. Picouleau, Minimum d-blockers and d-transversals in graphs, to appear in *Journal of Combinatorial Optimization*, 2011.

[9] B. Courcelle and S. Olariu, Upper bounds to the clique width of graphs, *Discrete Apllied Mathematics*, 101(1-3), 77–114, 2000.

[10] J. Egerváry, On combinatorial properties on matrices,*Math. Fiz. Lapok* 38, 16–28, 1931 (in Hungarian).

[11] G. N. Frederickson and R. Solis-Oba, Increasing the weight of minimum spanning trees, *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1996)*, 539–546, 1996.

[12] M. Habib and C. Paul, A simple linear time algorithm for cograph recognition, *Discrete Applied Mathematics*, 145(2), 183–197, 2005.

[13] R. Hassin. Approximation schemes for the restricted shortest path. *Mathematics of Operations Research*, 17(1), 36–42, 1992.

[14] R. M. Karp, Reducibility among combinatorial problems, in R. E. Miller and J. W. Thatcher (eds.), *Complexity of Computer Computations*, Plenum Press, 85–103, 1972.

[15] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, V. Gurvich, G. Rudolf, and J. Zhao, On short paths interdiction problems: total and node-wise limited interdiction, *Theory of Computing Systems*, 43(2), 204–233, 2008.

[16] S. Khot, Ruling Out PTAS for Graph Min-Bisection, Dense k-Subgraph, and Bipartite Clique, *SIAM Journal on Computing*, 36(4), 1025–1071, 2006.

[17] T. Kloks, Treewidth, computations and approximations, *Lecture Notes in Computer Science*, 842, 1994.

[18] D. Kőnig, Graphs and matrices, *Math. Fiz. Lapok* 38, 116–119 (in Hungarian), 1931.

[19] E. Petrank, The hardness of approximation: gap location, *Computational Complexity*, 4, 133–157, 1994.

[20] B. Ries, C. Bentz, C. Picouleau, D. de Werra, M. Costa, and R. Zenklusen, Blockers and transversals in some subclasses of bipartite graphs: When caterpillars are dancing on a grid, *Discrete Mathematics*, 310(11), 132–146, 2010.

[21] E. Wanke, k-NLC graphs and polynomial algorithms, *Discrete Applied Mathematics*, 54(2-3), 251–266, 1994.

[22] R. K. Wood, Deterministic network interdiction, *Mathematical and Computer Modeling*, 17(2), 1–18, 1993.

[23] R. Zenklusen, Matching interdiction, *Discrete Applied Mathematics*, 158, 1676–1690, 2010.

[24] R. Zenklusen, B. Ries, C. Picouleau, D. de Werra, M. Costa, and C. Bentz, Blockers and transversals, *Discrete Mathematics*, 309(13), 4306–4314, 2009.