A Refined Complexity Analysis of Finding the Most Vital Edges for Undirected Shortest Paths*

Cristina Bazgan[1,2], André Nichterlein[3], and Rolf Niedermeier[3]

[1] PSL, Université Paris-Dauphine, LAMSADE UMR CNRS 7243, France
bazgan@lamsade.dauphine.fr
[2] Institut Universitaire de France
[3] Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany
{andre.nichterlein,rolf.niedermeier}@tu-berlin.de

Abstract. We study the NP-hard Shortest Path Most Vital Edges problem arising in the context of analyzing network robustness. For an undirected graph with positive integer edge lengths and two designated vertices s and t, the goal is to delete as few edges as possible in order to increase the length of the (new) shortest st-path as much as possible. This scenario has been mostly studied from the viewpoint of approximation algorithms and heuristics, while we particularly introduce a parameterized and multivariate point of view. We derive refined tractability as well as hardness results, and identify numerous directions for future research. Among other things, we show that increasing the shortest path length by at least one is much easier than to increase it by at least two.

1 Introduction

Shortest Paths, that is, given two distinguished vertices s and t in a graph with edge lengths with the task to find a shortest st-path, is arguably one of the most basic graph problems. We study the undirected case with positive integer edge lengths in the context of "most vital edges" or (equivalently) "interdiction" or "edge blocker" problems. That is, we are interested in the scenario where the goal is to delete (few) edges such that in the resulting graph the shortest st-path gets (much) longer. This is motivated by obvious applications in investigating robustness and critical infrastructure in the context of network design. Our results provide new insights with respect to classical, parameterized, and approximation complexity of this fundamental edge deletion problem which is known to be NP-hard in general [2, 17].

The central decision problem we study is defined as follows.

Shortest Path Most Vital Edges (SP-MVE)
Input: An undirected graph $G = (V, E)$ with positive edge lengths $\tau\colon E \to \mathbb{N}$, two vertices $s, t \in V$, and integers $k, \ell \in \mathbb{N}$.
Question: Is there an edge subset $S \subseteq E$, $|S| \leq k$, such that the length of a shortest st-path in $G - S$ is at least ℓ?

* Work started during a visit (March 2014) of the second and third author at Université Paris-Dauphine.

We set $b := \ell - \text{dist}_G(s,t)$ to be the number by how much the length of every shortest st-path needs to be increased. If all edges have length one, then we say that the graph has unit-length edges. Naturally, SP-MVE comes along with two optimization versions: either delete as few edges as possible in order to achieve a length increase of at least b (called Min-Cost SP-MVE) or getting maximum length increase under the constraint that k edges can be deleted (called Max-Length SP-MVE). For an instance of SP-MVE or Max-Length SP-MVE we assume that k is smaller than the size of any st-cut in the input graph. Otherwise, removing all edges of a minimum-size st-cut (which is polynomial-time computable) would lead to a solution disconnecting s and t.

Related work. Due to the immediate practical relevance, there are numerous studies on "most vital edges (and vertices)" and related problems. We focus on shortest paths here, while there are also studies for problems such as Minimum Spanning Tree [3, 4, 11, 15, 20] or Maximum Flow [15, 25, 27], to mention only two. With respect to shortest path computation, the following is known. First, we mention in passing that a (more general) result of Fulkerson and Harding [12] implies that allowing the subdivision of edges instead of edge deletions as modification operation makes the problem polynomial-time solvable. Notably, it also has been studied to find *one* most vital edge of a shortest path; this can be solved in almost linear time [22].

Bar-Noy et al. [2] showed that SP-MVE is NP-complete and also corrected some errors concerning algorithmic results from earlier work [21]. Khachiyan et al. [17] derived polynomial-time constant-factor inapproximability results for both optimization versions. For the case of directed graphs, Israeli and Wood [16] provided heuristic solutions based on mixed-integer programming together with experimental results. Pan and Schild [25] studied the restriction of the directed case to planar graphs and again obtained NP-hardness results.

Finally, we note that, while most algorithmic studies focussed on polynomial-time solvable special cases or polynomial-time approximability, there seem to be almost no studies concerning multivariate complexity aspects [9, 24] of "most vital edges" ("edge interdiction") problems. We are only aware of the work of Guo and Shrestha [15] who performed a parameterized complexity analysis for minimum spanning tree, maximum matching, and maximum flow problems. They focus on standard parameterization by solution size, that is, the budget for the number of edge deletions, and derive several fixed-parameter tractability as well as parameterized hardness results.

Our results. We perform an extensive study of multivariate complexity aspects of SP-MVE. More specifically, we perform a refined complexity analysis in terms of how certain problem-specific parameters influence the computational complexity of SP-MVE and its optimization variants. The parameters we study include aspects of graph structure as well as special restrictions on the problem parameters itself. Moreover, we also report a few findings on (parameterized) approximability. Let us feature two main conclusions from our work: First, harming the network significantly (that is, $b \geq 2$) is NP-hard while harming it only a little bit

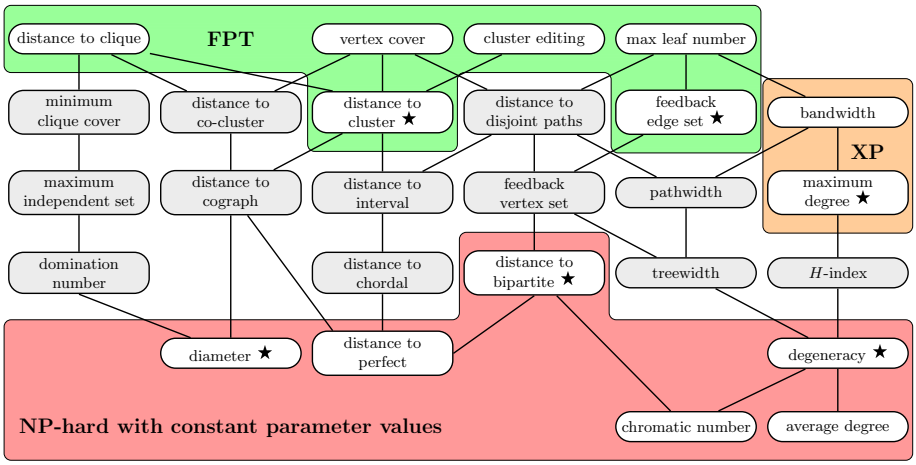

Fig. 1. The parameterized complexity of SP-MVE with unit-length edges with respect to different structural parameters. Herein, "distance to X" is the number of vertices that have to be deleted in order to transform the input graph into a graph from the graph class X. For two parameters that are connected by a line, the upper parameter is weaker (that is, larger) than the parameter below [18]. Refer to Sorge and Weller [26] for the formal definitions of the parameters. Results marked by a ★ are obtained in this work. XP means polynomial running time for constant parameter value.

(that is, $b = 1$) is doable in polynomial time. Second, while the parameterized complexity with respect to the standard parameter number k of edge deletions remains open, the cluster vertex deletion number, advocated by Doucha and Kratochvíl [7] as a parameterization between vertex cover number and clique-width, currently is our most interesting parameter that yields fixed-parameter tractability. Figure 1 surveys our current knowledge of the parameterized complexity of SP-MVE with respect to a number of well-known structural graph parameters, identifying numerous open questions. Moreover, towards the goal of spotting further fixed-parameter tractable special cases it also implicitly suggests to look for reasonable parameter combinations. To this end, a data-driven analysis of real-world parameter values would be valuable. In addition, Table 1 overviews our exact and approximate complexity results for SP-MVE.

Organization of the paper. After introducing some preliminaries in Section 2, we prove in Section 3 some NP-hardness results and in Section 4 some polynomial-time solvable special cases. In Section 5 we provide fixed-parameter and approximation algorithms for SP-MVE. Conclusions and open questions are provided in Section 6. Due to the lack of space, several details are omitted.

2 Preliminaries

For an undirected graph $G = (V, E)$ we set $|V| := n$ and $|E| := m$. A path P of length $r - 1$ in G is a sequence of vertices $P = v_1$-v_2-...-v_r with $\{v_i, v_{i+1}\} \in E$

Table 1. Overview on the computational complexity classification of SP-MVE on n-vertex graphs. XP means polynomial running time for constant parameter value.

	k	ℓ
related to polynomial time	XP	NP-hard for $b = 2$ and $\ell = 9$ ℓ-approximation
related to fpt time	$n/2^{O(\sqrt{\log n})}$-approximation for unit-length edges	$r(n)$-approximation for every increasing r
	fpt with respect to combined parameter (k, ℓ)	

for all $i \in \{1, \ldots, r-1\}$; the vertices v_1 and v_n are the endpoints of the path. For $1 \leq i < j \leq r$, we set v_iPv_j to be the subpath of P starting in v_i and ending in v_j, formally $v_iPv_j := v_i\text{-}v_{i+1}\text{-}\ldots\text{-}v_j$. For $i = 1$ or $j = r$ we omit the corresponding endpoint, that is, we set $Pv_j := v_1Pv_j$ and $v_iP := v_1Pv_j$. For $u, v \in V$, an uv-path P is a path with endpoints u and v. The distance between u and v in G, denoted by $\mathrm{dist}_G(u, v)$, is the length of a shortest uv-path. The diameter of G is the length of the longest shortest path in G.

For each vertex $v \in V$ we denote by $N_G(v)$ the set of neighbors of v and $N_G[v] = N_G(v) \cup \{v\}$ denotes v's the closed neighborhood. Two vertices $u, v \in V$ are called *true twins* if $N_G[u] = N_G[v]$ and *false twins* if $N_G(u) = N_G(v)$; they are called *twins* if they are either true or false twins. We denote by $G - S$ the graph obtained from G by removing the edge subset $S \subseteq E$. For $s, t \in V$, an edge subset S is called st-cut if $G - S$ contains no st-path. For $V' \subseteq V$ we denote by $G[V']$ the subgraph induced by V'. For $E' \subseteq V$ we denote by $G[E']$ the subgraph consisting of all endpoints of edges in E' and the edges in E'.

An edge set $F \subseteq E$ is a *feedback edge set* of G if $G - F$ is a tree or a forest. The feedback edge set number of G is the size of a minimum feedback edge set. Graph G is a *cluster graph* if G consists of disjoint cliques. A vertex set $X \subseteq V$ is called cluster vertex deletion set if $G[V \setminus X]$ is a cluster graph. The cluster vertex deletion number is the size of a minimum cluster vertex deletion set.

Parameterized complexity. A parameterized problem is called *fixed-parameter tractable* (fpt) if there is an algorithm that decides any instance (I, k), consisting of the "classical" instance I and a parameter $k \in \mathbb{N}_0$, in $f(k) \cdot |I|^{O(1)}$ time, for some computable function f solely depending on k.

A core tool in the development of fixed-parameter algorithms is polynomial-time preprocessing by data reduction, called *kernelization* [14, 19]. Here, the goal is to transform a given problem instance (I, k) in polynomial time into an equivalent instance (I', k') whose size is upper-bounded by a function of k. That is, (I, k) is a yes-instance if and only if (I', k'), $k' \leq g(k)$, and $|I'| \leq g(k)$ for some function g. Thus, such a transformation is a polynomial-time self-reduction with the constraint that the reduced instance is "small" (measured by $g(k)$). If such a transformation exists, then I' is called *kernel* of size $g(k)$. We refer to the monographs [8, 10, 23] for more details on parameterized complexity.

Approximation. Given an NP optimization problem and an instance I of this problem, we use $|I|$ to denote the size of I, $\mathrm{opt}(I)$ to denote the optimum value of I, and $\mathrm{val}(I,S)$ to denote the value of a feasible solution S of instance I. The *performance ratio* of S (or *approximation factor*) is $r(I,S) = \max\left\{\frac{\mathrm{val}(I,S)}{\mathrm{opt}(I)}, \frac{\mathrm{opt}(I)}{\mathrm{val}(I,S)}\right\}$. For a function ρ, an algorithm $\mathcal{A}$ is an $\rho(|I|)$-*approximation*, if for every instance I of the problem, it returns a solution S such that $r(I,S) \leq \rho(|I|)$. If the problem comes with a parameter k and the algorithm $\mathcal{A}$ runs in $f(k) \cdot |I|^{O(1)}$ time, then $\mathcal{A}$ is called *parameterized* $\rho(|I|)$-*approximation*.

3 NP-hard cases

Adapting a reduction idea due to Khachiyan et al. [17] for the vertex deletion variant of SP-MVE, we prove that SP-MVE is NP-hard even for constant values of b, ℓ, and graph diameter.

Theorem 1. *SP-MVE is NP-hard, even for unit-length edges, $b = 2$, $\ell = 9$, and diameter 8.*

Proof. As Khachiyan et al. [17], we reduce from Vertex Cover on three-partite graphs which remains NP-hard [13, GT1]. While the fundamental approach remains the same, the technical details when moving their vertex deletion scenario to our edge deletion scenario change to quite some extent. We refrain from a step-by-step comparison. Given a Vertex Cover instance (G,h) with $G = (V_1 \cup V_2 \cup V_3, E)$ being a tripartite graph on n vertices, we construct an SP-MVE instance I' as follows. First, we set $k := h$ and $\ell := 9$. The graph $G' = (V', E')$ contains vertices $V' = V_1 \cup V_2 \cup V_3 \cup V_2' \cup \{s,t\}$, where s and t are two new vertices, and for each $v \in V_2$ we add a copy $v' \in V_2'$.

Before describing the edge set E', we introduce edge-gadgets. Here, by adding a length α *edge-gadget*, $\alpha \geq 2$, from the vertex u to vertex v, we mean to add n vertex-disjoint paths of length $\alpha - 2$ and to make u adjacent to the first vertex of each path and v adjacent to the last vertex of each path. If $\alpha = 2$, then each path is just a single vertex which is at the same time the first and last vertex. The idea behind this is that we will never delete edges in an edge-gadget.

We add the following edges and edge-gadgets to G' (see Figure 2 for a schematic representation of the constructed graph). For each vertex $v \in V_2$ we add the edge $\{v, v'\}$ of length one. We add edges of length one between s and every vertex $v \in V_1$ and between t and every vertex $v \in V_3$. We also add the following edge-gadgets: For each edge $\{u,v\} \in (V_1 \times V_2) \cap E$ we add the edge-gadget $e_{u,v}$ of length two, for each edge $\{u,v\} \in (V_2 \times V_3) \cap E$ we add the edge-gadget $e_{u',v}$ of length two, and for each edge $\{u,v\} \in (V_1 \times V_3) \cap E$ we add the edge-gadget $e_{u,v}$ of length five. Furthermore, we add edge-gadgets of length four between s and every vertex $v \in V_2$ and between t and every vertex $v' \in V_2'$. Observe that we have $\mathrm{dist}_{G'}(s,t) = 7$ and thus $b = \ell - \mathrm{dist}_G(s,t) = 2$.

We now show that G has a vertex cover of size at most h if and only if deleting $k = h$ edges in G' results in s and t having distance at least $\ell = 9$.

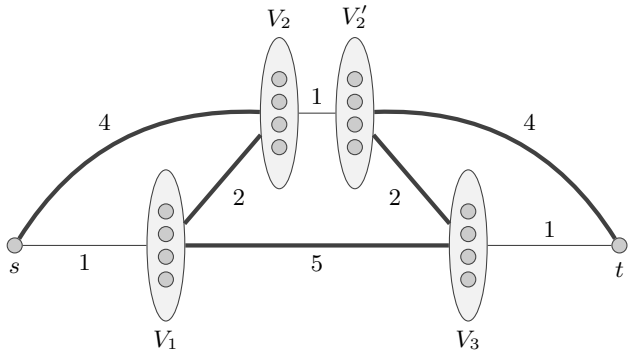

Fig. 2. A schematic representation of the graph G' constructed from the tripartite graph $G = (V_1 \cup V_2 \cup V_3, E)$. The vertices are grouped to the used sets. Each edge in G' is represented by a length-one edge in the picture. A bold edge indicates an edge-gadget and the corresponding number denotes its length.

"⇒:" Let $V'' \subseteq V$ be a vertex cover of size at most h in G. It is not hard to verify that for the set $E'' = \{\{s, v\} : v \in V_1 \cap V''\} \cup \{\{v, v'\} : v \in V_2 \cap V''\} \cup \{\{v, t\} : v \in V_3 \cap V''\}$ it holds that $\mathrm{dist}_{G'-E''}(s, t) = 9$ and $|E''| = |V''| \leq h$.

"⇐:" Let $E'' \subseteq E'$ be a set of edges such that $\mathrm{dist}_{G'-E''}(s, t) \geq 9$ and $|E''| \leq h$. If E'' contains edges from an edge-gadget $e_{u,v}$, then it must contain at least n edges from this gadget in order to have a chance to increase the solution value. Therefore, since $h < n$, we can assume that E'' does not contain any edge contained in an edge-gadget. Thus $E'' \subseteq (\{s\} \times V_1) \cup (V_2 \times V_2') \cup (V_3 \times \{t\})$. We construct a vertex cover V'' for G as follows: For each edge $\{s, v\} \in E''$ it follows that $v \in V_1$ and we add v to V''. Similarly, for each edge $\{v, t\} \in E''$ it follows that $v \in V_3$ and we add v to V''. Finally, for each edge $\{v, v'\} \in E'' \cap (V_2 \times V_2')$, we add v to V''.

Suppose towards a contradiction, that V'' is not a vertex cover in G, that is, there exists an edge $\{u, v\} \in E$ with $u, v \notin V''$. If $v \in V_1$ and $u \in V_2$, then the st-path s-v-u-u'-t of length $8 < \ell$ is contained in $G - E''$. If $v \in V_1$ and $u \in V_3$, then the st-path s-v-u-t of length $7 < \ell$ is contained in $G - E''$. Finally, if $v \in V_2$ and $u \in V_3$, then the st-path s-v-v'-u-t of length $8 < \ell$ is contained in $G - E''$. Each of the three cases contradicts the assumption that $\mathrm{dist}_{G'-E''}(s, t) \geq 9$. □

When allowing length zero on edges, Khachiyan et al. [17] stated that it is NP-hard to approximate Max-Length SP-MVE within a factor smaller than two. We consider in this paper only positive edge lengths and, by adapting the construction given in the above proof by considering edge-gadgets of lengths polynomial in n (with high degree), we obtain the following.

Theorem 2. Max-Length SP-MVE *is not* $4/3 - 1/poly(n)$*-approximable in polynomial time, even for unit-length edges, unless* $P = NP$.

see Proof 1 (appendix) Concerning special graph classes, we can show that the

problem remains NP-hard on restricted bipartite graphs. To this end, a graph G has *degeneracy* d if every subgraph of G contains a vertex of degree at most d. By subdividing every edge, we obtain the following.

Theorem 3. *SP-MVE is NP-hard, even for bipartite graphs with degeneracy two, unit-length edges,* $b = 4$*,* $\ell = 18$*, and diameter* 8*.*

Proof. We provide a self-reduction from SP-MVE with unit-length edges with $b = 2$, $\ell = 9$, and diameter 8. Let $I = (G = (V, E), k, \ell, s, t)$ be the given SP-MVE instance. We construct the instance $I' = (G', k, 2\ell, s, t)$ where G' is obtained from G by subdividing all edges, that is, each edge is replaced by a path of length two. The correctness of the reduction is easy to see as any minimal solution contains at most one edge of each induced path. Clearly, I' can be constructed in polynomial time. Furthermore, G' is bipartite and has degeneracy two. □

4 Polynomial-time algorithms

In this section, we discuss two polynomial-time algorithms for special cases of SP-MVE. First, we consider bounded-degree graphs. Here, the basic observation is that the maximum degree Δ of the graph upper-bounds the solution size: a budget of Δ would allow to disconnect s from t by deleting all edges incident to s. Hence, the simple brute-force algorithm trying all possibilities to delete at most $\Delta - 1$ edges yields a polynomial-time algorithm in graphs with constant maximum degree.

Proposition 1. *SP-MVE can be solved in* $O(m^{\Delta-1}(m + n \log n))$*.*

see Proof 2 (appendix) The question whether one can replace $m^{\Delta-1}$ by $f(\Delta) \cdot m^{O(1)}$ for some function f, that is, whether SP-MVE is fixed-parameter tractable with respect to Δ, remains open.

The second, more interesting special case we consider is when we only want to increase the distance between s and t by one. In this case, we can exploit the connection between SP-MVE and minimum st-cuts. Observe that a minimum st-cut in the undirected graph can be larger than the edge set we are looking for, see left-hand side of Figure 3 for an example. Instead, the idea is to direct the edges from s to t (see right-hand side of Figure 3) and then search an st-cut; this only works if $b = 1$, as also witnessed by the NP-hardness for the case $b = 2$ (Theorem 1).

Theorem 4. *SP-MVE can be solved in* $O(nm)$ *time when* $b = 1$*.*

Proof. Let $I = (G = (V, E), k, \ell, s, t, \tau)$ be an instance of SP-MVE with $\ell = \text{dist}_G(s, t) + 1$, that is, $b = 1$. The task is to delete at least one edge in each shortest st-path. To achieve this goal, our algorithm works in three main steps:

1. Using an adaption of Dijkstra's algorithm, compute the subgraph $G' = (V', E')$ of the original graph containing *all* shortest path edges.

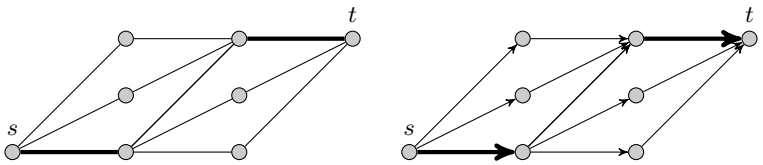

Fig. 3. The left-hand side shows a graph consisting of five st-paths of length three each. Removing the two bold edges increases the length of a shortest st-path to five. A minimum st-cut has size three, showing the difference between our vital edges scenario and minimum edge cuts. The right-hand side shows a digraph obtained from the graph on the left-hand side by directing all edges towards t. In this graph, the minimum st-cut has also size two. The proof of Theorem 4 shows that for $b = 1$ we can reduce SP-MVE to finding a minimum st-cut in a directed graph with unit arc-weights.

2. Direct all edges in these subgraphs from s to t (so that the tail of the each edge is closer to s than to t), obtaining a directed graph $D = (V', A)$.
3. Compute a minimum st-cut in D and decide accordingly (if it contains more than k arcs, then there is no solution, otherwise the cut arcs form a solution).

We first discuss the correctness and then show that the algorithm runs indeed in $O(nm)$ time. To this end, we introduce the following notation. For an arc subset $A' \subseteq A$ we denote by $E(A')$ the underlying undirected edges of A'; thus $E(A') \subseteq E' \subseteq E$. For the correctness, we first show that for every st-cut $A' \subseteq A$ in D it holds that every shortest st-path in G contains an edge in $E(A')$: Suppose towards a contradiction that there is a shortest st-path P not containing an edge in $E(A')$. Recall that we require the edge lengths to be positive. Hence, for every edge $\{u, v\}$ on the path P it holds that $\mathrm{dist}_G(s, u) \neq \mathrm{dist}_G(s, v)$ since P is a shortest path. Assume without loss of generality that $\mathrm{dist}_G(s, u) < \mathrm{dist}_G(s, v)$. By construction of D, this implies that $(u, v) \in A$. Thus, P is also an st-path in D containing no arc from A'; a contradiction to the assumption that A' is an st-cut.

Conversely, let $S \subseteq E'$ be an edge set containing at least one edge of every shortest st-path in G. Then, the arc set $A' \subseteq A$ with $E(A') = S$ is an st-cut in D: Suppose towards a contradiction that A' is not an st-cut in D. Thus, there exists an st-path P in $D - A'$. Hence, P is also an st-path in $G' - S$. We next show that P is indeed a shortest st-path in G'. Suppose towards a contradiction that there exists a vertex $v \in V'$ that is the first vertex on P such that Pv (the path P until vertex v) is not a shortest sv-path. Clearly, $s \neq v$. Let $u \in V'$ be the predecessor of v on P, that is, $Pv = Pu$-v (allowing $s = u$). By construction of G', each edge in E' is contained in at least one shortest st-path. Hence, also the edge $\{u, v\}$ is contained in some shortest st-path P'. This implies that $P'v$ is a shortest sv-path in G'. Since Pu is a shortest su-path, it follows that Pu and $P'u$ have same length. This implies that also $Pv = Pu$-v and $P'v = P'u$-v have same length. Hence, Pv is a shortest sv-path in G'; a contradiction. Thus, P is a shortest st-path in G'. Since P is also contained in $G' - S$, this contradicts the assumption that S contains an edge of each shortest st-path in G.

Summarizing, we showed that a set $A' \subseteq A$ is an st-cut in D if and only if $E(A')$ contains an edge of each shortest st-path in G. Hence, any minimum-size st-cut computed in Step 3 induces a minimum-size edge set S in G such that $\mathrm{dist}_{G-S}(s,t) > \mathrm{dist}_G(s,t)$. This completes the correctness proof.

For the running time we show that Steps 1 and 2 can be performed in $O(n \log n + m)$ time: First, run in $O(n \log n + m)$ time two times Dijkstra's algorithm to compute for each vertex its distance to s and to t. Then, an edge $\{u, v\}$ is contained in a shortest st-path if either $\mathrm{dist}_G(s,u) + \tau(\{u,v\}) + \mathrm{dist}_G(v,t) = \mathrm{dist}_G(s,t)$ (Case 1) or $\mathrm{dist}_G(s,v) + \tau(\{u,v\}) + \mathrm{dist}_G(u,t) = \mathrm{dist}_G(s,t)$ (Case 2). Furthermore, in Case 1, the edge will be directed from u to v and in Case 2 from v to u. This can be done in $O(m)$ time by iterating over all edges. Using the Ford-Fulkerson-algorithm to compute the minimum cut, Step 3 can be performed in D in $O(nm)$ time. Overall, this gives a running time of $O(nm)$. □

5 Algorithms for NP-hard cases

In this section, we present fixed-parameter and approximation algorithms. As a warm-up, we show that SP-MVE is fixed-parameter tractable when combining the parameters number k of removed edges and minimum st-path length ℓ to be achieved.

Proposition 2. *SP-MVE can be solved in $O((\ell-1)^k \cdot (n \log n + m))$ time.*

Proof. We employ a simple depth-bounded search tree. The basic idea is to search for a shortest st-path and to "destroy" it by deleting one of the edges (trying all possibilities). This is repeated until every shortest st-path has length at least ℓ. For each such shortest path, we branch into at most $\ell - 1$ possibilities to delete one of its edges, and the depth of the corresponding search tree is at most k (our "deletion budget") since otherwise we cannot find a solution with at most k edge deletions. The correctness is obvious. Hence, we arrive at a search tree of size at most $(\ell-1)^k$ where in each step we need to compute a shortest path. Using Dijkstra's algorithm, this can be done in $O(n \log n + m)$ time. The overall running time is thus $O((\ell-1)^k \cdot (n \log n + m))$. □

Using the search tree described in the proof of Proposition 2 to destroy all paths of length at most $2^{O(\sqrt{\log n})}$ yields the following.

Corollary 1. Max-Length SP-MVE *with unit-length edges is parameterized $n/2^{O(\sqrt{\log n})}$-approximable with respect to the parameter k.*

see Proof 3 (appendix) By deleting every edge on too short st-paths, we obtain an ℓ-approximation.

Proposition 3. Min-Cost SP-MVE *is polynomial-time ℓ-approximable.*

see Proof 4 (appendix) Combining the previous approximation algorithm with a tradeoff between running time and approximation factor [5, Lemma 2], we obtain the following.

Corollary 2. *For every increasing function r* Min-Cost SP-MVE *is parameterized $r(n)$-approximable with respect to the parameter ℓ.*

Parameter feedback edge set number. We next provide a linear-size problem kernel for SP-MVE parameterized by the feedback edge set number. Recall that an edge set $F \subseteq E$ is called *feedback edge set* for a graph $G = (V, E)$ if $G - F$ is a tree or a forest. The feedback edge set number of G is the size of a minimum feedback edge set. Computing a spanning tree, one can determine a minimum feedback edge set in linear time. Hence, we assume in the following that we are given a feedback edge set F with $|F| = f$ for our input instance $(G = (V, E), k, \ell, s, t, \tau)$. We start with two simple data reduction rules dealing with degree-one and degree-two vertices.

Rule 1 *Let $(G = (V, E), k, \ell, s, t, \tau)$ be an SP-MVE instance and let $v \in V \setminus \{s, t\}$ be a vertex of degree one. Then, delete v.*

The correctness of Rule 1 is obvious as no shortest path uses a degree-one vertex. We can deal with degree-two vertices as follows.

Rule 2 *Let $(G = (V, E), k, \ell, s, t, \tau)$ be an SP-MVE instance and let $v \in V \setminus \{s, t\}$ be a vertex of degree two with $N_G(v) = \{u, w\}$ and $\{u, w\} \notin E$. Then add the edge $\{u, w\}$ with the length $\tau(\{u, w\}) := \tau(\{u, v\}) + \tau(\{v, w\})$ and delete v.*

The correctness of Rule 2 follows from the fact that on an induced path at most one edge will be deleted and it does not matter which one will get deleted. Applying both rules exhaustively can clearly be done in polynomial time and leads to the following problem kernel.

Theorem 5. *SP-MVE admits a problem kernel with $5f + 2$ vertices and $6f + 2$ edges.*

see Proof 5 (appendix)

Corollary 3. *SP-MVE is fixed-parameter tractable with respect to the parameter feedback edge set number.*

Parameter cluster vertex deletion number. We now prove that SP-MVE restricted to unit-length edges is fixed-parameter tractable with respect to the parameter cluster vertex deletion number x. Recall that a graph G is a *cluster graph* if it is a union of disjoint cliques. A vertex set $X \subseteq V$ is called cluster vertex deletion set if $G[V \setminus X]$ is a cluster graph. The cluster vertex deletion number is the size of a minimum cluster vertex deletion set.

We assume in the following that for the input instance $(G = (V, E), k, \ell, s, t)$ we are given a cluster vertex deletion set X of size $|X| = x$. If X is not already given, then we can compute X in $O(1.92^x \cdot (n + m))$ time [6]. Our algorithm is based on the observation that twins can be handled equally in a solution. This follows from a more general statement provided in the following lemma. It shows that for any set $T \subseteq V \setminus \{s, t\}$ of vertices that have the same neighborhood in $V \setminus T$, we can assume that we do not delete edges in $G[T]$ and that the vertices in T behave the same, that is, one deletes either all edges or no edge between a vertex $v \in V \setminus T$ and the vertices in T.

Lemma 1. *Let $G = (V, E)$ be an undirected graph with unit-length edges, let $s, t \in V$ be two vertices, and let $T = \{v_1, \ldots, v_t\} \subseteq V \setminus \{s, t\}$ be a set of vertices such that $N_G(v_1) \setminus T = N_G(v_2) \setminus T = \ldots = N_G(v_t) \setminus T$. Then, for every edge subset $S \subseteq E$, there exists an edge subset $S' \subseteq E$ such that $\mathrm{dist}_{G-S'}(s, t) \geq \mathrm{dist}_{G-S}(s, t)$, $|S'| \leq |S|$, and $N_{G[S']}(v_1) = N_{G[S']}(v_2) = \ldots = N_{G[S']}(v_t)$.*

see Proof 6 (appendix)

Theorem 6. *SP-MVE with unit-length edges is linear-time fixed-parameter tractable with respect to the parameter cluster vertex deletion number.*

Proof. Let $(G = (V, E), k, \ell, s, t)$ be the input instance of SP-MVE and let $X \subseteq V$ be a cluster vertex deletion set of size $|X| = x$. Hence, $G - X$ is a cluster graph and the vertex sets $C_1, \ldots, C_r$ form the cliques (clusters) for some $r \in \mathbb{N}$. We set $\mathcal{C} := \{C_1, \ldots, C_r\}$. Assume that there is an SP-MVE solution $S \subseteq E$ of size at most k; otherwise the algorithm will output no as it finds no solution. We describe an algorithm that finds S.

Our algorithm is based on the following observation. Let P be an arbitrary shortest st-path that goes through a clique $C \in \mathcal{C}$ in $G - S$. Then, P contains at most 2^x vertices from C: By Lemma 1, we can assume that the twins in G are still twins in $G - S$. Since P is a shortest path, P does not contain two vertices that are twins. As the vertices in C form a clique, they only differ in how they are connected to vertices in X. Thus, C contains at most 2^x "different" vertices, that is, vertices with pairwise different neighborhoods.

Now, consider two non-adjacent vertices $u, v \in X$. From the above considerations it follows that in $G - S$ an uv-path avoiding the vertices in X has length between one and $2^x + 1$ as it can pass through at most one clique. Our algorithm tries for each vertex pair from X all possibilities for the distance it has in $G - S$ and then tries to realize the current possibility. After the current possibility is realized, the cliques in $\mathcal{C}$ are obsolete and thus the instance size can be bounded in a function of x. More precisely, our algorithm works as follows:

1. Branch into all possibilities to delete edges contained in $G[X]$. Decrease the budget k accordingly.
2. Branch into all possibilities to add for each pair u, v of non-adjacent vertices in X an edge with a length of $\{2, 3, \ldots, 2^x, 2^x + 1, \infty\}$ indicating the length of a shortest path between u and v that does not contain any vertex in X.
3. Delete for each clique containing neither s nor t the *minimum number* of edges to ensure that a shortest path between each pair of vertices in X is completely contained in $G[X]$. Decrease the budget k accordingly.
4. Remove all cliques except the ones that contain s or t. Do *not* change the budget k.
5. Solve the problem on the remaining graph with the remaining budget (that was not spent in Steps 1 and 3).

Note that Step 2 is performed for each possibility in Step 1. Hence, in Steps 1 and 2 at most $2^{x^2} \cdot (2^x + 1)^{x^2}$ possibilities are considered and for each of these possibilities Step 3 is invoked.

In Step 3, the algorithm tries to realize the prediction made in Step 2. To this end, let $C \in \mathcal{C}$ be a clique containing neither s nor t. The algorithm branches into all possibilities to delete edges in $G[C]$ or edges with one endpoint in C and the other endpoint in X. Since $G[C]$ contains at most 2^x different vertices, it follows from Lemma 1 that at most $2^{(2^x)^2+2^x \cdot x} = 2^{(4^x)+2^x \cdot x}$ possibilities need to be considered to delete edges. For each possibility, the algorithm checks in $x^{O(1)}$ time whether all shortest paths between a pair of vertices of X go through C. If yes, then the algorithm discards the currently considered branch; if no, then the current branch is called valid. From all valid branches for C, the algorithm picks the one that deletes the minimum amount of edges and proceeds with the next clique. Observe that since X is a vertex separator for all cliques in $\mathcal{C}$, the algorithm can solve Step 3 for each clique independently of the outcome in the other cliques. Hence, the overall running time for Step 3 is $2^{2^{O(x)}} \cdot n$ as $|\mathcal{C}| \leq n$.

As discussed above, the cliques in $\mathcal{C}$ containing neither s nor t are now obsolete as there is always a shortest path avoiding these cliques. Hence, the algorithm removes these cliques (Step 4). This can be done in linear time. The remaining instance consists of the vertices in X and the at most two cliques containing s and t. As the algorithm deleted the edges within $G[X]$ in Step 1, it remains to consider deleting edges within the two cliques or between the two cliques and the vertices in X. Again, by Lemma 1, the algorithm only needs to branch into $2^{2 \cdot (4^x + x \cdot 2^x)}$ possibilities to delete edges and check for each branch whether s and t have distance at least ℓ and the overall budget k is not exceeded. If one branch succeeds, then the algorithm found a solution and returns it. If no branch succeeds, then there exists no solution of size k since the algorithm performed an exhaustive search. Overall, the running time is $2^{2^{O(x)}} \cdot (n+m)$. □

6 Conclusion

Shortest Path Most Vital Edges (SP-MVE) is a natural edge deletion problem that clearly deserves further study from a parameterized complexity perspective. While we showed that SP-MVE remains NP-hard for even constant values of the parameters b and ℓ relating to the length increase, we left open whether SP-MVE is fixed-parameter tractable with respect to the "standard parameter" k (number of edge deletions). Even fixed-parameter tractability with respect to the combined parameter (b, k) remains open. Figure 1 in the introductory section depicts a wide range of structural graph parameters for which the parameterized complexity status of SP-MVE is unknown. Also concerning the approximation point of view not much is known. There is a huge gap between the known lower and upper bounds of the approximation factor achievable in polynomial time. Further, from a practical point of view it would make sense to extend our studies by restricting the input to planar graphs [25]. Finally, also in terms of parameterized approximability SP-MVE offers a number of interesting challenges, altogether making it an excellent candidate problem for a full-fledged multivariate complexity analysis.

Bibliography

[1] S. Arora and C. Lund. Hardness of approximations. In *Approximation algorithms for NP-hard problems*, pages 399–446. PWS Publishing Company, 1996.

[2] A. Bar-Noy, S. Khuller, and B. Schieber. The complexity of finding most vital arcs and nodes. Technical report, College Park, MD, USA, 1995.

[3] C. Bazgan, S. Toubaline, and D. Vanderpooten. Efficient determination of the k most vital edges for the minimum spanning tree problem. *Computers and Operations Research*, 39(11):2888–2898, 2012.

[4] C. Bazgan, S. Toubaline, and D. Vanderpooten. Critical edges/nodes for the minimum spanning tree problem: complexity and approximation. *Journal of Combinatorial Optimization*, 26(1):178–189, 2013.

[5] C. Bazgan, M. Chopin, A. Nichterlein, and F. Sikora. Parameterized approximability of maximizing the spread of influence in networks. *Journal of Discrete Algorithms*, 27:54–65, 2014.

[6] A. Boral, M. Cygan, T. Kociumaka, and M. Pilipczuk. A fast branching algorithm for cluster vertex deletion. In *Proc. 9th CSR*, volume 8476 of *LNCS*, pages 111–124. Springer, 2014.

[7] M. Doucha and J. Kratochvíl. Cluster vertex deletion: A parameterization between vertex cover and clique-width. In *Proc. 37th MFCS*, volume 7464 of *LNCS*, pages 348–359. Springer, 2012.

[8] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.

[9] M. R. Fellows, B. M. P. Jansen, and F. A. Rosamond. Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. *European Journal of Combinatorics*, 34(3):541–566, 2013.

[10] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[11] G. N. Frederickson and R. Solis-Oba. Increasing the weight of minimum spanning trees. *Proc. 7th SODA*, pages 539–546, 1996.

[12] D. Fulkerson and G. C. Harding. Maximizing the minimum source-sink path subject to a budget constraint. *Mathematical Programming*, 13:116–118, 1977.

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

[14] J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, 38(1):31–45, 2007.

[15] J. Guo and Y. R. Shrestha. Parameterized complexity of edge interdiction problems. In *Proc. 20th COCOON*, volume 8591 of *LNCS*, pages 166–178. Springer, 2014.

[16] E. Israeli and R. K. Wood. Shortest-path network interdiction. *Networks*, 40(2): 97–111, 2002.

[17] L. Khachiyan, E. Boros, K. Borys, K. M. 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.

[18] C. Komusiewicz and R. Niedermeier. New races in parameterized algorithmics. In *Proc. 37th MFCS*, volume 7464 of *LNCS*, pages 19–30. Springer, 2012.

[19] S. Kratsch. Recent developments in kernelization: A survey. *Bulletin of the EATCS*, 113:58–97, 2014.

[20] W. Liang. Finding the k most vital edges with respect to minimum spanning trees for fixed k. *Discrete Applied Mathematics*, 113(2-3):319–327, 2001.

[21] K. Malik, A. Mittal, and S. K. Gupta. The k most vital arcs in the shortest path problem. *Operations Research Letters*, 8:223–227, 1989.

[22] E. Nardelli, G. Proietti, and P. Widmayer. A faster computation of the most vital edge of a shortest path. *Information Processing Letters*, 79(2):81–85, 2001.

[23] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[24] R. Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In *Proc. 27th STACS*, volume 5 of *LIPIcs*, pages 17–32. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2010.

[25] F. Pan and A. Schild. Interdiction problems on planar graphs. In *Proc. 16th APPROX-RANDOM*, volume 8096 of *LNCS*, pages 317–331. Springer, 2013.

[26] M. Sorge and M. Weller. The graph parameter hierarchy. Manuscript, 2013. URL `http://fpt.akt.tu-berlin.de/msorge/parameter-hierarchy.pdf`.

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

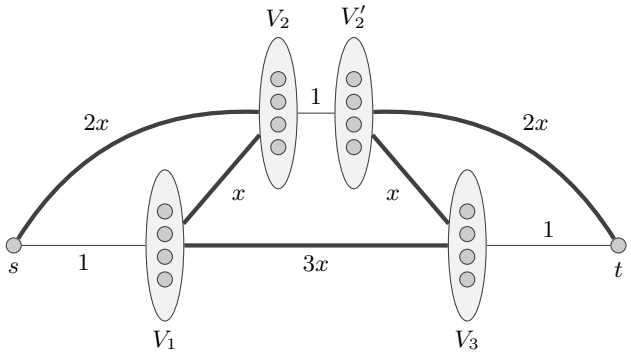

Fig. 4. A schematic representation of the graph G' constructed from the tripartite graph $G = (V_1 \cup V_2 \cup V_3, E)$. The vertices are grouped to the used sets. Each edge in G' is represented by a length-one edge in the picture. A bold edge indicates an edge-gadget and the corresponding number denotes its length.

A Proofs

A.1 Proof 1 (Theorem 2)

Proof. We construct a gap-reduction [1] from Vertex Cover on three-partite graphs to Max-Length SP-MVE.

In this proof we use a *gap*-reduction from a decision problem to a maximization problem. A decision problem Π is called *gap-reducible* to a maximization problem Π' with gap $\rho > 1$ if for any instance I of Π we can construct an instance I' of Q in polynomial time while satisfying the following properties.

- If I is a yes-instance, then $opt(I') \geq c(|I|)$.
- If I is a no-instance, then $opt(I') < \frac{c(|I|)}{\rho(|I|)}$.

The idea behind a *gap*-reduction is that if Π is NP-hard then Π' is not approximable within a factor ρ provided that P $\neq$ NP.

Starting with an instance $(G = (V, E), h)$ of Vertex Cover on three-partite graphs we construct an instance $I' = (G' = (V', E'), k, s, t)$ of Max-Length SP-MVE as in the proof of Theorem 1. We only change some lengths as follows (see also Figure 4): For each edge $\{u, v\} \in (V_1 \times V_2) \cap E$ we add the edge-gadget $e_{u,v}$ of length x, for each edge $\{u, v\} \in (V_2 \times V_3) \cap E$ we add the edge-gadget $e_{u',v}$ of length x, and for each edge $\{u, v\} \in (V_1 \times V_3) \cap E$ we add the edge-gadget $e_{u,v}$ of length $3x$. We add edge-gadgets of length $2x$ between s and every vertex $v \in V_2$ and between t and every vertex $v' \in V_2'$. The value x could be any polynomial function in $|V| = n$. Observe that we have $\text{dist}_{G'}(s, t) = 3x + 2$.

We now show that if G has a vertex cover of size at most h then $opt(I') = 4x + 1$, otherwise $opt(I') \leq 3x + 2$.

Let $V'' \subseteq V$ be a vertex cover of size at most h in G. It is not hard to verify that for the set $E'' = \{\{s, v\} : v \in V_1 \cap V''\} \cup \{\{v, v'\} : v \in V_2 \cap V''\} \cup \{\{v, t\} : v \in V_3 \cap V''\}$ it holds that $\text{dist}_{G'-E''}(s, t) = 4x + 1$ and $|E''| = |V''| \leq h$.

Suppose now that G has no vertex cover of size h. Let $E'' \subseteq E'$ be a set of h edges. We can assume as in the proof of Theorem 1 that E'' does not contain any edge from an edge-gadget. Thus $E'' \subseteq (\{s\} \times V_1) \cup (V_2 \times V_2') \cup (V_3 \times \{t\})$. We construct a vertex set V'' for G as follows: For each edge $\{s, v\} \in E''$, we add v to V'' and for each edge $\{v, t\} \in E''$, we add v to V''. Finally, for each edge $\{v, v'\} \in E'' \cap (V_2 \times V_2')$, we add v to V''.

Since V'' is not a vertex cover in G, there exists an edge $\{u, v\} \in E$ with $u, v \notin V''$. If $v \in V_1$ and $u \in V_2$, then the st-path s-v-u-u'-t of length $3x+2$ is contained in $G - E''$. If $v \in V_1$ and $u \in V_3$, then the st-path s-v-u-t of length $3x + 2$ is contained in $G - E''$. Finally, if $v \in V_2$ and $u \in V_3$, then the st-path s-v-v'-u-t of length $3x + 2$ is contained in $G - E''$.

Since Vertex Cover is NP-hard on three-partite graphs [13, GT1], Max-Length SP-MVE is not $\frac{4x+1}{3x+2} = 4/3 - 1/poly(n)$-approximable in polynomial time. □

A.2 Proof 2 (Proposition 1)

Proof. Recall that we assume k to be smaller than the size of a minimum st-cut and, thus, $k < \Delta$ as otherwise we could simply delete all edges incident to s. The straightforward algorithm branching in all m^k cases to delete at most k edges and checking with Dijkstra's algorithm whether the distance between s and t is high enough runs in $O(m^k(m + n \log n)) = O(m^{\Delta-1}(m + n \log n))$ time. □

A.3 Proof 3 (Corollary 1)

Proof. We employ the search tree algorithm behind Proposition 2; it has size $O((\ell-1)^k)$. The idea now is to either compute an optimal solution in fpt-time or to derive the stated approximation in polynomial time.

Our parameterized approximation algorithm works as follows. Trying $\ell = 1, 2, \ldots, f(n)$ we employ the search tree to detect whether there is an optimal solution of length smaller than $f(n)$. Namely, if the search tree for some ℓ-value says no, then we know that we found an optimal solution with the previous search tree and output this. Otherwise, we reach $\ell = f(n)$ and thus, since the optimal value is at most $n - 1$, this means that we have a factor-$n/f(n)$-approximation.

Clearly, the overall running time of this whole procedure is a polynomial times $f(n)^k$. It remains to determine for which (maximum) function $f(n)$ this still yields an fpt running time for parameter k. First, if $k > \log(f(n))$, then $f(n)^k$ can be upper-bounded by a function only in k and we are done. Second, if $k \leq \log(f(n))$, then for $f(n) \leq 2^{O(\sqrt{\log n})}$ we have that $f(n)^k \leq f(n)^{\log(f(n))} = 2^{(\log(f(n))^2}$. The latter term is polynomial iff $f(n) = 2^{O(\sqrt{\log n})}$. □

A.4 Proof 4 (Proposition 3)

Proof. Let $I = (G = (V, E), \ell, s, t, \tau)$ be an instance of Min-Cost SP-MVE. We repeat the following algorithm until the shortest st-path has length at least ℓ. Set $G' := G$ and let P be a shortest st-path in G'. If $\tau(P) < \ell$ then set $G' := G' - E(P)$ and proceed. Denote by i the number of iterations the algorithm realizes. Let E'' be the set of all edges of the i shortest paths removed from G. The size of E'' is $|E''| \leq i\ell$ since at each step at most ℓ edges are deleted. Moreover, $opt(I) \geq i$ since an optimal solution contains at least one edge of each of these i paths. The algorithm clearly runs in polynomial time. □

A.5 Proof 5 (Theorem 5)

Proof. Let $(G = (V, E), k, \ell, s, t, \tau)$ be the input instance of SP-MVE. First, we exhaustively apply Rules 1 and 2. Clearly, this can be done in polynomial time. It remains to bound the size of the reduced graph G'. To this end, first observe that G' contains at most f degree-two vertices as every degree-two vertex that is not deleted by Rule 2 has two adjacent neighbors and thus induces together with its neighbors a cycle. Thus, G' contains at most f vertices of degree two. It remains to bound the number of vertices with degree at least three. To this end, let r denote the number of leaves in the tree $G' - F$. Thus, $G' - F$ contains at most $r - 2$ vertices of degree at least three. Due to Rule 1, G' contains at most two degree-one vertices (s and t) and, hence, $r \leq 2f + 2$. Furthermore, there are at most $2f$ degree-three vertices in G' that are incident to an edge in F. Hence, G' contains at most $4f + 2$ vertices of degree at least three. In total, G' contains at most $5f + 2$ vertices and, thus, $6f + 2$ edges. □

A.6 Proof 6 (Lemma 1)

Proof. Starting from the edge subset $S \subseteq E$, we construct S' having the desired properties. To this end, we abbreviate $\ell := \text{dist}_{G-S}(s, t)$. Let $T \in V \setminus \{s, t\}$ be a set of vertices such that $N(u) \setminus T = N(v) \setminus T$ for each pair $u, v \in T$. Assume that the vertices in T do not have the same neighborhood in $G[S]$; otherwise we simply set $S' := S$. Let $u \in T$ be a vertex that has in the graph (V, S) the smallest degree of all vertices in T, that is, the vertex in T that is incident to the least number of edges in S. Now, construct S' as follows. First, initialize S' as a copy of S. Second, remove all edges of S' that have both endpoints in T. Third, for each $v \in T \setminus \{u\}$ remove all edges incident to v from S and add for each edge $\{u, w\} \in S$ the edge $\{v, w\}$. Summarizing, S' is composed as follows:

$$S' := (S \setminus \{\{v, w\} \mid v \in T \setminus \{u\} \wedge w \in V\}) \cup \{\{v, w\} \mid v \in T \setminus \{u\} \wedge w \in V \setminus T \wedge \{u, w\} \in S\}.$$

By construction of S' we have $|S'| \leq |S|$. Furthermore, we have $N_{G[S']}(v) = N_{G[S]}(u) \setminus T$ for all $v \in T$ and thus $N_{G[S']}(v) = N_{G[S']}(v')$ for each pair $v, v' \in T$.

It remains to show that in $G - S'$ the distance between s and t is at least ℓ. To this end, assume by contradiction that $G - S'$ contains an st-path P of length less than ℓ. Since, by construction of S', each edge in $S \setminus S'$ has at least one endpoint in T, it follows that P contains at least one vertex of T. Let v and v' be the first respectively last vertex of T on P (possibly $v = v'$) and let w, w' be the vertices before v respectively after v' on P, that is,

$$P = s\text{-}\ldots\text{-}w\text{-}v\text{-}\ldots\text{-}v'\text{-}w'\text{-}\ldots\text{-}t.$$

Since $w, w' \notin T$, $N_G(v) \setminus T = N_G(v') \setminus T$, and $N_{G[S']}(v) = N_{G[S']}(v')$, it follows that $Pw\text{-}v\text{-}w'P$ is also an st-path with length less than ℓ in $G - S'$. Similarly, it follows that $P' := Pw\text{-}u\text{-}w'P$ is also an st-path with length less than ℓ in $G - S'$ (where u is the vertex used in the construction of S'). Since $N_{G[S']}(u) = N_{G[S]}(u) \setminus T$ it follows that $\{u, w\}, \{u, w'\} \notin S$, implying that P' is an st-path of length less than ℓ in $G - S$; a contradiction to the assumption that $\text{dist}_{G-S}(s, t) = \ell$. □