Fixed-parameter algorithms for minimum cost edge-connectivity augmentation

DÁNIEL MARX, Institute for Computer Science and Control, Hungarian Academy of Sciences
LÁSZLÓ VÉGH, London School of Economics

We consider connectivity-augmentation problems in a setting where each potential new edge has a nonnegative cost associated with it, and the task is to achieve a certain connectivity target with at most p new edges of minimum total cost. The main result is that the minimum cost augmentation of edge-connectivity from $k-1$ to k with at most p new edges is fixed-parameter tractable parameterized by p and admits a polynomial kernel. We also prove the fixed-parameter tractability of increasing edge-connectivity from 0 to 2, and increasing node-connectivity from 1 to 2.

Categories and Subject Descriptors: F.2 [**Theory of Computation**]: Analysis of Algorithms and Complexity; G.2.1 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: fixed parameter algorithms, connectivity augmentation

ACM Reference Format:
Dániel Marx, László A. Végh, 2014. Fixed-parameter algorithms for minimum cost edge-connectivity augmentation *ACM Trans. Algor.* V, N, Article A (January YYYY), 23 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Designing networks satisfying certain connectivity requirements has been a rich source of computational problems since the earliest days of algorithmic graph theory: for example, the original motivation of Borůvka's work on finding minimum cost spanning trees was designing an efficient electricity network in Moravia [Nesetril et al. 2001]. In many applications, we have stronger requirements than simply achieving connectivity: one may want to have connections between (certain pairs of) nodes even after a certain number of node or link failures. Survivable network design problems deal with such more general requirements.

In the simplest scenario, the task is to achieve k-edge-connectivity or k-node-connectivity by adding the minimum number of new edges to a given directed or undirected graph G. This setting already leads to a surprisingly complex theory and, somewhat unexpectedly, there are exact polynomial-time algorithms for many of these questions. For example, there is a polynomial-time algorithm for achieving

An extended abstract of this paper appeared at the 40th International Colloquium on Automata, Languages, and Programming, Lecture Notes in Computer Science Volume 7965, 2013, pp 721-732.
The first author was supported by the European Research Council (ERC) grant "PARAMTIGHT: Parameterized complexity and the search for tight complexity results," reference 280152 and OTKA grant NK105645.
Author's addresses: D. Marx, Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary; L. A. Végh, Department of Management, London School of Economics & Political Science, London, UK.

DOI: http://dx.doi.org/10.1145/0000000.0000000

k-edge-connectivity in an undirected graph by adding the minimum number of edges (Watanabe and Nakamura [1987], see also Frank [1992]). For k-node-connectivity, a polynomial-time algorithm is known only for the special case when the graph is already $(k-1)$-node-connected; the general case is still open [Végh 2011]. We refer the reader to the recent book by Frank [2011] on more results of similar flavour. One can observe that increasing connectivity by one already poses significant challenges and in general the node-connectivity versions of these problems seem to be more difficult than their edge-connectivity counterparts.

For most applications, minimizing the number of new edges is a very simplified objective: for example, it might not be possible to realize direct connections between nodes that are very far from each other. A slightly more realistic setting is to assume that the input specifies a list of potential new edges ("links") and the task is to achieve the required connectivity by using the minimum number of links from this list. Unfortunately, almost all problems of this form turn out to be NP-hard: deciding if the empty graph on n nodes can be augmented to be 2-edge-connected with n new edges from a given list is equivalent to finding a Hamiltonian cycle (similar simple arguments can show the NP-hardness of augmenting to k-edge-connectivity also for larger k). Even though these problems are already hard, this setting is still unrealistic: it is difficult to imagine any application where all the potential new links have the same cost. Therefore, one typically tries to solve a minimum cost version of the problem, where for every pair u, v of nodes, a (finite or infinite) cost $c(u, v)$ of connecting u and v is given. When the goal is to achieve k-edge connectivity, we call this problem *Minimum Cost Edge-Connectivity Augmentation to* k (see Section 2 for a more formal definition). In the special case when the input graph is assumed to be $(k-1)$-edge-connected (as in, e.g., [Jordán 1995; Hsu 2000; Kortsarz and Nutov 2007; Végh 2011]), we call the problem *Minimum Cost Edge-Connectivity Augmentation by One*. Alternatively, one can think of this problem with the edge-connectivity target being the minimum cut value of the input graph plus one. The same terminology will be used for the node-connectivity versions and the minimum cardinality variants (where every cost is either 1 or infinite).

Due to the hardness of the more general minimum cost problems, research over the last two decades has focused mostly on the approximability of the problem. This field is also known as survivable network design, e.g., [Agrawal et al. 1995; Goemans and Williamson 1995; Jain 2001; Cheriyan et al. 2003; Kortsarz and Nutov 2003; Cheriyan and Végh 2014]; for a survey, see [Kortsarz and Nutov 2007]. In this paper, we approach these problems from the viewpoint of parameterized complexity. We say that a problem with parameter p is *fixed-parameter tractable (FPT)* if it can be solved in time $f(p)\cdot n^{O(1)}$, where $f(p)$ is an arbitrary computable function depending only on p and n is the size of the input [Downey and Fellows 1999; Flum and Grohe 2006]. The tool box of fixed-parameter tractability includes many techniques such as bounded search trees, color coding, bidimensionality, etc. The method that received most attention in recent years is the technique of kernelization [Lokshtanov et al. 2012; Misra et al. 2011]. A *polynomial kernelization* is a polynomial-time algorithm that produces an equivalent instance of size $p^{O(1)}$, i.e., polynomial in the parameter, but not depending on the size of the instance. Clearly, polynomial kernelization implies fixed-parameter tractability, as kernelization in time $n^{O(1)}$ followed by any brute force algorithm on the $p^{O(1)}$-size kernel yields a $f(p)\cdot n^{O(1)}$ time algorithm. The conceptual message of polynomial kernelization is that the hard problem can be solved by first applying a preprocessing to extract a "hard core" and then solving this small hard instance by whatever method available. An interesting example of fixed-parameter tractability in the context of connectivity augmentation is the result by Jackson and Jordán [2005], showing that for the problem of making a graph k-node-connected by adding a minimum number of ar-

bitrary new edges admits a $2^{O(k)} \cdot n^{O(1)}$ time algorithm (it is still open whether there is a polynomial-time algorithm for this problem).

As observed above, if the link between arbitrary pair of nodes is not always available (or if they have different costs for different pairs), then the problem for augmenting a $(k-1)$-edge-connected graph to a k-edge-connected one is NP-hard for any fixed $k \geq 2$. Thus for these problems we cannot expect fixed-parameter tractability when parameterizing by k. In this paper, we consider a different parameterization: we assume that the input contains an integer p, which is a upper bound on the number of new links that can be added. Assuming that the number p of new links is much smaller than the size of the graph, exponential dependence on p is still acceptable, as long as the running time depends only polynomially on the size of the graph. It follows from Nagamochi [2003, Lemma 7] that *Minimum Cardinality Edge-Connectivity Augmentation from 1 to 2* is fixed-parameter tractable parameterized by this upper bound p. Guo and Uhlmann [2010] showed that this problem, as well as its node-connectivity counterpart, admits a kernel of $O(p^2)$ nodes and $O(p^2)$ links. Neither of these algorithms seem to work for the more general minimum cost version of the problem, as the algorithms rely on discarding links that can be replaced by more useful ones. Arguments of this form cannot be generalized to the case when the links have different costs, as the more useful links can have higher costs. Our results go beyond the results of [Nagamochi 2003; Guo and Uhlmann 2010] by considering higher order edge-connectivity and by allowing arbitrary costs on the links.

We present a kernelization algorithm for the problem *Minimum Cost Edge-Connectivity Augmentation by One* for arbitrary k. The algorithm starts by doing the opposite of the obvious: instead of decreasing the size of the instance by discarding provably unnecessary links, we add new links to ensure that the instance has a certain closure property; we call instances satisfying this property *metric instances*. We argue that these changes do not affect the value of the optimum solution. Then we show that a metric instance has a bounded number of important links that are provably sufficient for the construction of an optimum solution. The natural machinery for this approach via metric instances is to work with a more general problem. Besides the costs, every link is equipped with a positive integer weight. Parallel links between pairs of nodes will be therefore allowed. Our task is to find a minimum cost set of links of total weight at most p whose addition makes the graph k-edge-connected. Our main result addresses the corresponding problem, *Weighted Minimum Cost Edge-Connectivity Augmentation*.

THEOREM 1.1. *Weighted Minimum Cost Edge-Connectivity Augmentation by One admits a kernel of $O(p)$ nodes, $O(p)$ edges, $O(p^3)$ links, with all costs being integers of $O(p^6 \log p)$ bits.*

Our result hence gives an $O(2^{O(p \log p)}|V|^{O(1)})$ time algorithm for the problem. Very recently, this was improved by Basavaraju et al. [2014] to $9^p|V|^{O(1)}$, by a reduction to a Steiner tree problem in a certain auxiliary graph.

The original problem is the special case when all links have weight one. Strictly speaking, Theorem 1.1 does not give a kernel for the original problem, as the kernel may contain links of higher weight even if all links in the input had weight one. Our next theorem, which can be derived from the previous one, shows that we may obtain a kernel that is an unweighted instance. However, there is a trade-off in the bound on the kernel size.

THEOREM 1.2. *Minimum Cost Edge-Connectivity Augmentation by One admits a kernel of $O(p^4)$ nodes, $O(p^4)$ edges and $O(p^4)$ links, with all costs being integers of $O(p^8 \log p)$ bits.*

Let us now outline the main ideas of the proof of Theorem 1.1. We first show that every input can be efficiently reduced to a metric instance, one with the closure property. We first describe our algorithm in the special case of increasing edge-connectivity from 1 to 2, where connectivity augmentation can be interpreted as covering a tree by paths. The closure property of the instance allows us to prove that there is an optimum solution where every new link is incident only to "corner nodes" (leaves and branch nodes). Either the problem is infeasible, or we can bound the number of corner nodes by $O(p)$. Hence we can also bound the number of potential links in the resulting small instance.

Augmenting edge connectivity from 2 to 3 is similar to augmenting from 1 to 2, but this time the graph we need to work on is no longer a tree, but a cactus graph. Thus the arguments are slightly more complicated, but generally go along the same lines. Finally, in the general case of increasing edge-connectivity from $k-1$ to k, we use the uncrossing properties of minimum cuts and a classical result of Dinits, Karzanov, and Lomonosov [1976] to show that (depending on the parity of k) the problem can be always reduced to the case $k=2$ or $k=3$.

In kernels for the weighted problem, a further technical issue has to be overcome: each finite cost in the produced instance has to be a rational number represented by $p^{O(1)}$ bits. As we have no assumption on the sizes of the numbers appearing in the input, this is a nontrivial requirement. It turns out that a technique of Frank and Tardos [1987] (used earlier in the design of strongly polynomial-time algorithms) can be straightforwardly applied here: the costs in the input can be preprocessed in a way that the each number is an integer of $O(p^6 \log p)$ bits long and the relative costs of the feasible solutions do not change. We believe that this observation is of independent interest, as this technique seems to be an essential tool for kernelization of problems involving costs.

To prove Theorem 1.2 (see Section 3.6), we first obtain a kernel by applying our weighted result to the unweighted instance; this kernel will however contain links of weight higher than one. Still, every link f of weight $w(f)$ in the (weighted) kernel can be replaced by a sequence of $w(f)$ original unweighted edges. This replaces the $O(p^3)$ links by $O(p^4)$ original ones.

We try to extend our results in two directions. First, we show that in the case of increasing connectivity from 1 to 2, the node-connectivity version can be directly reduced to the edge-connectivity version (see Section 3.7).

THEOREM 1.3. *Weighted Minimum Cost Node-Connectivity Augmentation from* 1 *to* 2 *admits a a kernel of* $O(p)$ *nodes,* $O(p)$ *edges,* $O(p^3)$ *links, with all costs being integers of* $O(p^6 \log p)$ *bits.*

For higher connectivities, we do not expect such a clean reduction to work. Polynomial-time exact and approximation algorithms for node-connectivity are typically much more involved than for edge-connectivity (compare e.g., [Watanabe and Nakamura 1987] and [Frank 1992] to [Frank and Jordán 1995] and [Végh 2011]), and it is reasonable to expect that the situation is similar in the case of fixed-parameter tractability.

A natural goal for future work is trying to remove the assumption of Theorems 1.1 and 1.2 that the input graph is $(k-1)$-connected. In the case of 2-edge-connectivity, we show that the problem is fixed-parameter tractable even if the input graph is not connected. However, the algorithm uses nontrivial branching and it does not provide a polynomial kernel.

THEOREM 1.4. *Minimum Cost Edge-Connectivity Augmentation to* 2 *can be solved in time* $2^{O(p \log p)} \cdot n^{O(1)}$.

The proof is given in Section 4. The additional branching arguments needed in Theorem 1.4 can show a glimpse of the difficulties one can encounter when trying to solve the problem for larger k, especially with respect to kernelization. For augmentation by one, the following notion of shadows was crucial to define the metric closure of the instances: f is a shadow of link e if the weight of e is at most that of f, and e covers every k-cut covered by f — in other words, substituting link f by link e retains the same connectivity. When the input graph is not assumed to be connected, we cannot extend the shadow relation to links connecting different components, only in special, restricted situations. Therefore, we cannot prove the existence of an optimal solution with all links incident to corner nodes only. Instead, we prove that there is an optimal solution such that all leaves are adjacent to either corner nodes or certain other special nodes; this enables the branching in the FPT algorithm. A further difficulty arises if we want to avoid using two copies of the same link. This was automatically excluded for augmentation by one, whereas now further efforts are needed to enforce this requirement.

2. PRELIMINARIES

For a set V, let $\binom{V}{2}$ denote the edge set of the complete graph on V. Let $n = |V|$ denote the number of nodes. For a node set $X \subseteq V$ and a set of edges (or links) $F \subseteq \binom{V}{2}$, let $d_F(X)$ denote the number of edges (or links) in F with endpoints $u \in X$ and $v \in V \setminus X$. When we are given a graph $G = (V, E)$ and it is clear from the context, $d(X)$ will denote $d_E(X)$. A node set $\emptyset \neq X \subsetneq V$ will be called a *cut*, and *minimum cut* if $d(X)$ takes the minimum value. For a function $z : V \to \mathbb{R}$, and a set $X \subseteq V$, let $z(X) = \sum_{v \in X} z(v)$ (we use the same notation with functions on edges as well). For $u, v \in V$, a set $X \subseteq V$ is called an $u\bar{v}$-set if $u \in X$, $v \in V \setminus X$.

Let us be given an undirected graph $G = (V, E)$ (possibly containing parallel edges), a connectivity target $k \in \mathbb{Z}_+$, and a cost function $c : \binom{V}{2} \to \mathbb{R}_+ \cup \{\infty\}$. For a given nonnegative integer p, our aim is to find a minimum cost set of edges $F \subseteq \binom{V}{2}$ of cardinality at most p such that $(V, E \cup F)$ is k-edge-connected.

We will work with a more general version of this problem. Let E^* denote an edge set on V, possibly containing parallel edges. We call the elements of E *edges* and the elements of E^* *links*. Besides the cost function $c : E^* \to \mathbb{R}_+ \cup \{\infty\}$, we are also given a positive integer weight function $w : E^* \to \mathbb{Z}_+$. We restrict the total weight of the augmenting edge set to be at most p instead of restricting its cardinality. Let us define our main problem.

Weighted Minimum Cost Edge Connectivity Augmentation	
Input:	Graph $G = (V, E)$, set of links E^*, integers $k, p > 0$, weight function $w : E^* \to \mathbb{Z}_+$, cost function $c : E^* \to \mathbb{R}_+ \cup \{\infty\}$.
Find:	minimum cost link set $F \subseteq E^*$ such that $w(F) \leq p$ and $(V, E \cup F)$ is k-edge-connected.

A problem instance is thus given by (V, E, E^*, c, w, k, p). An $F \subseteq E^*$ for which $(V, E \cup F)$ is k-edge-connected is called an *augmenting link set*. If all weights are equal to one, we simply refer to the problem as *Minimum Cost Edge Connectivity Augmentation*.

As defined above, an optimal solution to *Weighted Minimum Cost Edge Connectivity Augmentation* does not allow using the same link in E^* twice. Motivated by the original (unweighted) problem, a natural further restriction is to forbid using multiple links (of possibly different weights) between the same two nodes u and v. If the input graph is already $(k-1)$-edge-connected, neither of these restrictions makes a difference, since

given an augmenting edge set, deleting all but one links from a parallel bundle is still an augmenting edge set. In Section 4 we investigate the problem of augmenting an arbitrary (possibly disconnected) graph to 2-edge-connected, where using parallel links may result in a cheaper solution. We first solve here the problem with allowing multiple copies of the same link, and in Section 4.3, we show how the problem can be solved if parallel links are forbidden.

For a set $S \subseteq V$, by G/S we mean the contraction of S to a single node s. That is, the node set of the contracted graph is $(V - S) \cup \{s\}$, and every edge uv with $u \notin S$, $v \in S$ is replaced by an edge us (possibly creating parallel edges); edges inside S are removed. Note that S is not assumed to be connected. We also contract the links to E^*/S accordingly.

We say that two nodes x and y are *k-inseparable* if there is no $x\bar{y}$-set X with $d(X) < k$. By Menger's theorem, this is equivalent to the existence of k edge-disjoint paths between x and y; this property can be tested in polynomial time by a max flow-min cut computation. Let us say that the node set $S \subseteq V$ is *k-inseparable* if any two nodes $x, y \in S$ are k-inseparable. It is easy to verify that being k-inseparable is an equivalence relation.[1] The maximal k-inseparable sets hence give a partition of the node set V. The following proposition provides us with a preprocessing step that can be used to simplify the instance:

PROPOSITION 2.1. *For a problem instance (V, E, E^*, c, w, k, p), let $S \subseteq V$ be a k-inseparable set of nodes. Let us consider the instance obtained by the contraction of S. Assume $\bar{F} \subseteq E^*/S$ is an optimal solution to the contracted problem. Then the pre-image of $\bar{F}$ in E^* is an optimal solution to the original problem.*

PROOF. We claim that for a link set $F \subseteq E^*$, $(V, E \cup F)$ is k-edge-connected if and only if adding the image $\bar{F}$ of F to the contracted graph is k-edge-connected. It is straightforward that if F is an augmenting link set, then so is $\bar{F}$. Conversely, assume for a contradiction that $\bar{F}$ is an augmenting link set but F is not. This means that there exists a set $X \subseteq V$ with $d_E(X) + d_F(X) < k$. Since S is k-inseparable, either $S \subseteq X$ or $S \cap X = \emptyset$. This implies that under the contraction the image of X will violate k-edge-connectivity in the augmented graph, a contradiction. □

Note that contracting a k-inseparable set S does not affect whether $x, y \notin S$ are k-inseparable. Thus by Proposition 2.1, we can simplify the instance by contracting each class of the partition given by the k-inseparable relation. Observe that after such a contraction, there are no longer any k-inseparable pair of nodes any more. Thus we may assume in our algorithms that every pair of nodes can be separated by a cut of size smaller than k.

3. AUGMENTING EDGE CONNECTIVITY BY ONE

Assume that the input graph is already $(k-1)$-edge-connected. It is easy to see that in an augmenting link set, it is sufficient to keep only one link from every bundle of parallel links. Therefore, we can exclude parallel links of the same weight. This motivates the following notation.

An edge between $x, y \in V$ will be denoted as xy. For a link f, we use $f = (x, y)$ if it is a link between x and y; note that there might be several links between the same nodes with different weights. We may ignore all links of weight $> p$. If for a pair of nodes $u, v \in V$, there are two links e and f between u and v such that $c(e) \leq c(f)$ and $w(e) \leq w(f)$, then we may also ignore the link f, as discussed above.

[1] To see transitivity, observe that if x and y are k-inseparable and y and z are k-inseparable, then a cut X separating x and z would either separate x and y, or y and z, a contradiction.

Subroutine METRIC-COMPLETION(c)
for $t = 1, 2, \ldots, p$ **do**
  for every 3 links $e = (u, v)$, $f = (v, z)$, $h = (u, z)$ with $w(h) = t \geq w(e) + w(f)$ **do**
    $c(h) \leftarrow \min\{c(h), c(e) + c(f)\}$
  for every link f with $w(f) = t$ **do**
    $c(f) \leftarrow \min\{c(e) : f$ is a shadow of $e\}$.

Fig. 1. The algorithm for computing the metric completion

It is convenient to assume that for every value $1 \leq t \leq p$ and every two nodes $u, v \in V$, there is exactly one link e between u and v with $w(e) = t$ (if there is no such link in the input E^*, we can add one of cost ∞). This e will be referred to as the *t-link between u and v*. With this convention, in this section we will assume that E^* consists of exactly p copies of $\binom{V}{2}$: a t-link between any two nodes $u, v \in V$ for every $1 \leq t \leq p$. However, in the input links of infinite cost should not be listed. (We avoid the discussion of exactly how the links are represented in the input: as we express the size of the kernel in terms of the number of nodes/edges/links, the exact representation does not matter for our results.)

3.1. Metric instances

The following notions will be used for augmenting edge-connectivity from 1 to 2 and from 2 to 3. We formulate them here in a generic way. Assume the input graph is $(k-1)$-edge-connected. Let $\mathcal{D}$ denote the set of all minimum cuts, represented by the node sets. That is, $X \in \mathcal{D}$ if and only if $d(X) = k - 1$. Note that, by the minimality of the cut, both X and $V \setminus X$ induce connected graphs if $X \in \mathcal{D}$. For a link $e = (u, v) \in E^*$, let us define $\mathcal{D}(e) \subseteq \mathcal{D}$ as the subset of minimum cuts *covered* by e. That is, $X \in \mathcal{D}$ is in $\mathcal{D}(e)$ if and only if X is an $u\bar{v}$-set or a $v\bar{u}$-set. Clearly, augmenting edge-connectivity by one is equivalent to covering all the minimum cuts of the graph.

PROPOSITION 3.1. *Assume (V, E) is $(k-1)$-edge-connected. Then $(V, E \cup F)$ is k-edge-connected if and only if $\cup_{e \in F} \mathcal{D}(e) = \mathcal{D}$.*

The following definition identifies the class of metric instances that plays a key role in our algorithm.

Definition 3.2. We say that the link f is a *shadow* of link e, if $w(f) \geq w(e)$ and $\mathcal{D}(f) \subseteq \mathcal{D}(e)$. The instance (V, E, E^*, c, w, k, p) is *metric*, if

(1) $c(f) \leq c(e)$ holds whenever the link f is a shadow of link e.
(2) Consider three links $e = (u, v)$, $f = (v, z)$ and $h = (u, z)$ with $w(h) \geq w(e) + w(f)$. Then $c(h) \leq c(e) + c(f)$.

Whereas the input instance may not be metric, we can create its metric completion with the following simple subroutine. Let us call the inequalities in *(i) shadow inequalities* and those in *(ii) triangle inequalities*. Let us define the *rank* of the inequality $c(f) \leq c(e)$ to be $w(f)$, and the rank of $c(h) \leq c(e) + c(f)$ to be $w(h)$. By *fixing* the triangle inequality $c(h) > c(e) + c(f)$, we mean decreasing the value of $c(h)$ to $c(e) + c(f)$.

The subroutine METRIC-COMPLETION(c) (see Figure 1) consists of p iterations, one for each $t = 1, 2, \ldots, p$. In the t'th iteration, first all triangle inequalities of rank t are taken in an arbitrary order, and the violated ones are fixed. Then for every t-link f, we decrease $c(f)$ to the minimum cost of links e such that f is a shadow of e. Note that we perform these steps one after the other for every violated inequality: in each

step, we decrease the cost of a single link f only (this will be important in the analysis of the algorithm). The first part of iteration 1 is void as there are no rank 1 triangle inequalities. The subroutine can be implemented in polynomial time: the number of triangle inequalities is $O(p^3n^3)$, and they can be efficiently listed; furthermore, every link is the shadow of $O(pn^2)$ other ones.

LEMMA 3.3. *Consider a problem instance (V, E, E^*, c, w, k, p) with the graph (V, E) being $(k-1)$-edge-connected.* METRIC-COMPLETION(c) *returns a metric cost function $\bar{c}$ with $\bar{c}(e) \leq c(e)$ for every link $e \in E^*$. Moreover, if for a link set $\bar{F} \subseteq E^*$, the graph $(V, E \cup \bar{F})$ is k-edge-connected, then there exists an $F \subseteq E^*$ such that $(V, E \cup F)$ is k-edge-connected, $c(F) \leq \bar{c}(\bar{F})$, and $w(F) \leq w(\bar{F})$. Consequently, an optimal solution for $\bar{c}$ provides an optimal solution for c.*

PROOF. Inequality $\bar{c}(e) \leq c(e)$ clearly holds for all links since the algorithm only decreases the costs. To verify the metric property, we prove that at the end of iteration t, all rank t inequalities are satisfied. This implies that the final cost function is metric, as the costs of the edges participating in rank t inequalities are not modified during any later iteration.

Consider a triangle inequality with links $t = w(h) \geq w(e) + w(f)$. As $w(e), w(f) < t$, the costs of e and f are not modified in iteration t. After fixing this inequality if necessary, we have $c(h) \leq c(e) + c(f)$. In the second part of the iteration, $c(h)$ may only decrease. Consequently, all triangle inequalities of rank t must be valid at the end of iteration t.

Let $\tilde{c}$ denote the cost function at the end of the first part of iteration t, after fixing all triangle inequalities. Using the fact that the shadow relation is transitive, it is easy to see that the values $c(f)$ after the second part of iteration t equal

$$c(f) = \min\{\tilde{c}(e) : f \text{ is a shadow of } e\}. \tag{1}$$

Consider now two links e and f with f being a shadow of e, and let $t = w(f) \geq w(e)$. We have to show $c(f) \leq c(e)$ at the end of iteration t. This is straightforward if $w(e) < t$: the new value of $c(f)$ is defined as a minimum value taken over a set containing $c(e)$; $c(e)$ itself is not modified. Assume now $w(e) = t$. Let h be the link giving the minimum in (1) for the link e, that is, the new value is $c(e) = \tilde{c}(h)$ with e being the shadow of h. Again by the transitivity of the shadow relation, f is also a shadow of h, and consequently, $c(f) \leq \tilde{c}(h) = c(e)$, as required.

For the second part of the lemma, it is enough to verify the statement for the case when $\bar{c}$ arises by a single modification step from c (i.e., fixing a triangle inequality or taking a minimum). First, assume we fixed a triangle inequality $c(h) > c(e) + c(f)$ by setting $\bar{c}(h) = c(e) + c(f)$ and $\bar{c}(g) = c(g)$ for every $g \neq h$. Consider an edge set $\bar{F}$ such that $(V, E \cup \bar{F})$ is k-edge-connected. If $h \notin \bar{F}$, then $F = \bar{F}$ satisfies the conditions. If $h \in \bar{F}$, then let us set $F = (\bar{F} \setminus \{h\}) \cup \{e, f\}$. We have $c(F) \leq \bar{c}(F)$, $w(F) \leq w(\bar{F})$. Furthermore, every cut covered by h must be covered by either e or h, implying that $(V, E \cup F)$ is also k-edge-connected.

Next, assume $\bar{c}(f) = c(e)$ was set for a link e such that f is a shadow of e, and $\bar{c}(g) = c(g)$ for every $g \neq f$. Now $F = (\bar{F} \setminus \{f\}) \cup \{e\}$ clearly satisfies the conditions: recall that by the definition of shadows, $\mathcal{D}(f) \subseteq \mathcal{D}(e)$. □

The proof also provides an efficient way for transforming an augmenting link set $\bar{F}$ to another F as in the lemma. For this, in every step of METRIC-COMPLETION(c) we have to keep track of the inequalities responsible for cost reductions.

By Lemma 3.3, we may restrict our attention to metric instances. In what follows, we show how to construct a kernel for metric instances for cases $k = 2$ and $k = 3$. (The case $k = 2$ could be easily reduced to $k = 3$, but we treat it separately as it is somewhat

simpler and more intuitive.) Section 3.4 then shows how the case of general k can be reduced to either of these cases depending on the parity of k.

3.2. Augmentation from 1 to 2

In this section, we assume that the input graph (V, E) is connected. By Proposition 2.1, we may assume that it is a tree: after contracting all the 2-inseparable sets, there are no two nodes with two edge-disjoint paths between them, implying that there is no cycle in the graph.

The minimum cuts are given by the edges, that is, $\mathcal{D}$ is in one-to-one correspondence with E. For a link e between two nodes $u, v \in V$, let $P(e) = P(u, v)$ denote the unique path between u and v in this tree. Then the link f is a shadow of the link e if $P(f) \subseteq P(e)$ and $w(f) \geq w(e)$. Now Proposition 3.1 simply amounts to the following.

PROPOSITION 3.4. *Graph $(V, E \cup F)$ is 2-edge-connected if and only if $\cup_{e \in F} P(e) = E$.*

Based on Lemma 3.3, it suffices to solve the problem assuming that the instance $(V, E, E^*, c, w, 2, p)$ is metric. The main observation is that in a metric instance we only need to use links that connect certain special nodes, whose number we can bound by a function of p.

Let us refer to the leaves and nodes of degree at least 3 as *corner nodes*; let $R \subseteq V$ denote their set. Every leaf in the tree (V, E) requires at least one incident edge in F. If the number of leaves is greater than $2p$, we may conclude that the problem is infeasible. (Formally, in this case we may return the following kernel: a single edge as the input graph with an empty link set.) If there are at most $2p$ leaves, then $|R| \leq 4p-2$, due to the following simple fact.

PROPOSITION 3.5. *The number of nodes of degree at least 3 in a tree is at most the number of leaves minus 2.*

Based on the following theorem, we can obtain a kernel on at most $4p - 2$ nodes by deleting all links incident to degree-2 nodes, and then contracting each path of degree-2 nodes to a single edge. The number of links in the kernel will be $O(p^3)$: there are $O(p^2)$ possible edges and p possible weights for each edge.

THEOREM 3.6. *For a metric instance $(V, E, E^*, c, w, 2, p)$, there exists an optimal solution F such that every edge in F is only incident to corner nodes.*

PROOF. For every link f, let $\ell(f) = |P(f)|$ denote the length of the path in the tree between its endpoints. Consider an optimal solution F such that $|F|$ is minimal, and subject to this, $\ell(F) = \sum_{f \in F} \ell(f)$ is minimal. We show that no link in this set F can be incident to a degree 2 node.

For a contradiction, assume that $f = (u, y) \in F$ has an endnode y having degree 2 in E; let x and z denote the two neighbors of y, with $xy \in P(f)$. Since $(V, E \cup F)$ is 2-edge-connected, there must be a link $e \in F$ with $yz \in P(e)$. We distinguish two cases, as illustrated in Figure 2.

Case I. $xy \in P(e)$. In this case, we may replace the link $f = (u, y)$ by a link $f' = (u, x)$ with $w(f') = w(f)$. By property (i) of metric instances, we have $c(f') \leq c(f)$ as f' is a shadow of f. By Proposition 3.4, $(V, E \cup F')$ is still 2-edge-connected for the resulting solution F', yet $|F'| = |F|$, $c(F') \leq c(F)$ and $\ell(F') < \ell(F)$, a contradiction to the choice of F.

Case II. $xy \notin P(e)$. This is only possible if e is incident to y, say $e = (y, v)$. For $t = w(f)+w(e)$, consider the t-link h between u and v. By property (ii), $c(h) \leq c(f)+c(e)$. Furthermore, $P(h) = P(f) \cup P(e)$. For the resulting solution $F' = F \setminus \{e, f\} \cup \{h\}$, the

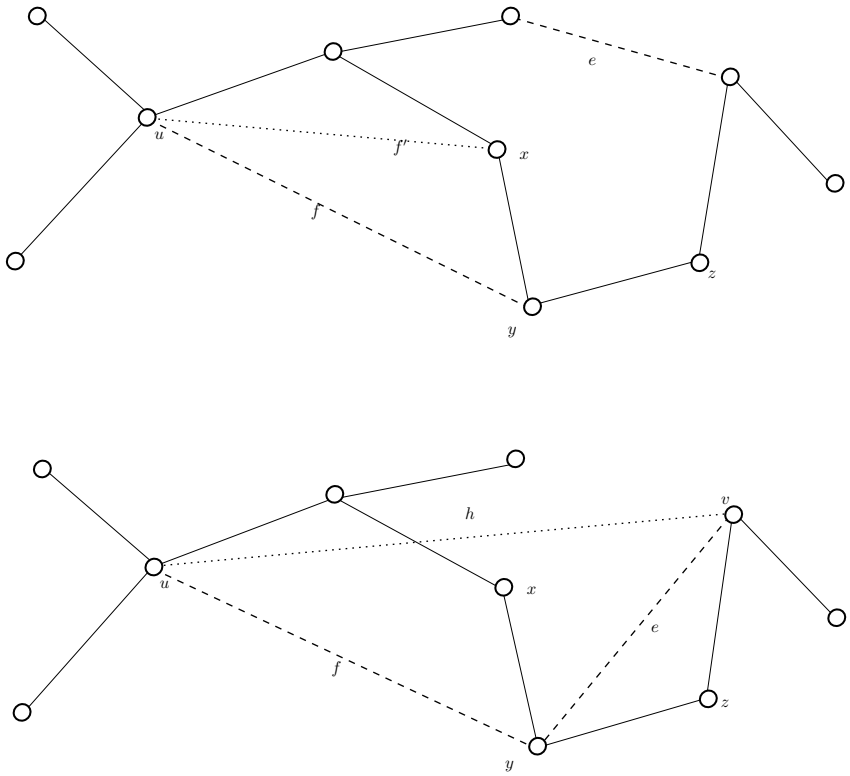

Fig. 2. Illustration of Cases I and II in the proof of Theorem 3.6.

graph $(V, E \cup F')$ is 2-edge-connected, $c(F') \leq c(F)$ and $|F'| < |F|$, a contradiction again. □

3.3. Augmentation from 2 to 3

In this section, we assume that the input graph is 2-edge-connected but not 3-edge-connected. Let us call a 2-edge-connected graph $G = (V, E)$ a *cactus*, if every edge belongs to exactly one circuit. This is equivalent to saying that every block (maximal induced 2-node-connected subgraph) is a circuit (possibly of length 2, using two parallel edges). Figure 3 gives an example of a cactus.

By Proposition 2.1, we may assume that every 3-inseparable set in G is a singleton, that is, there are no two nodes in the graph connected by 3 edge-disjoint paths.

PROPOSITION 3.7. *Assume that $G = (V, E)$ is a 2-edge-connected graph such that every 3-inseparable set is a singleton. Then G is a cactus.*

PROOF. By 2-edge-connectivity, every edge must be contained in at least one circuit. For a contradiction, assume there is an edge e contained in two different circuits C_1 and C_2. Pick an edge $f \in C_1 \setminus C_2$, and take the maximal path P in C_1 containing f such that the nodes incident to both P and C_2 are precisely the endpoints of P, say x and y. The edge $e \in C_1 \cap C_2$ guarantees the existence of such a path, that is, $x \neq y$. Now there are three edge-disjoint paths connecting x and y: P and the two $x - y$ paths contained in C_2. This contradicts our assumption. □

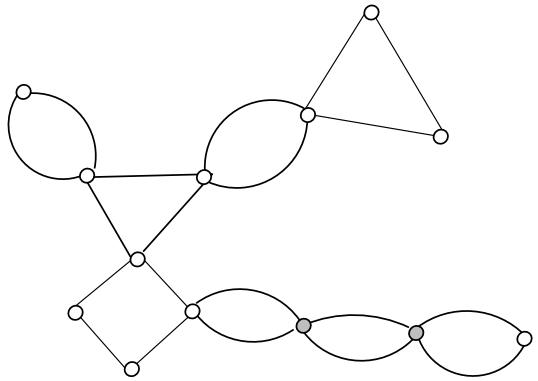

Fig. 3. A cactus graph. The shaded nodes are in the set T.

In the rest of the section, we assume that $G = (V, E)$ is a cactus. The set of minimum cuts $\mathcal{D}$ corresponds to arbitrary pairs of 2 edges on the same circuit. We say that the node b *separates* the nodes a and c, if every path between a and c must traverse b (we allow $a = b$ or $b = c$).

PROPOSITION 3.8. *Consider links $e = (u, v)$ and $f = (x, y)$ with $w(f) \geq w(e)$. Then f is a shadow of e if and only if both x and y separate u and v.*

PROOF. To see sufficiency, assume that both x and y separate u and v, and consider an $x\bar{y}$-set $X \in \mathcal{D}(f)$. We have to show that $X \in \mathcal{D}(e)$, that is, one of u and v is in X and the other in $V \setminus X$. Indeed, assume for a contradiction that $u, v \in X$. Since X is connected, it contains a path between u and v avoiding y, a contradiction. The case $u, v \in V \setminus X$ is symmetric.

For necessity, assume w.l.o.g. x does not separate u and v, that is, there exists a path Q between u and v not containing x. Pick two edges incident to x that are contained in the same cycle, and such that they separate x and y. They correspond to a minimum cut $X \in \mathcal{D}(f)$ (they are the two edges between X and $V - X$). The path Q is either entirely contained in X or in $V - X$ (as it cannot traverse the edges incident to x), and therefore $e = (u, v)$ cannot cover X. This contradicts $\mathcal{D}(f) \subseteq \mathcal{D}(e)$. □

Again by Lemma 3.3, we may restrict our attention to metric instances. Let us call a circuit of length 2 a *2-circuit* (that is, a set of two parallel edges between two nodes). Let R_1 denote the set of nodes of degree 2, or equivalently, the set of nodes incident to exactly one circuit. Let R_2 denote the set of nodes incident to at least 3 circuits, or at least two circuits not both 2-circuits. Let $R = R_1 \cup R_2$ and let $T = V \setminus R$ denote the set of remaining nodes, that is, the set of nodes that are incident to precisely two circuits, both 2-circuits (see Figure 3). The elements of R will be again called *corner nodes*. We can give the following simple bound:

PROPOSITION 3.9. $|R_2| \leq 4|R_1| - 8$.

PROOF. The proof is by induction on $|V|$. If all circuits in G are 2-circuits, that is, G is created by duplicating every edge of a tree, R_1 corresponds to the leaves and R_2 to the branching nodes. The claim follows by Proposition 3.5, as $|R_1| \geq 2$. Assume now G has at least one circuit C of length $r \geq 3$, and has $t \leq r$ nodes incident to other circuits. Consider the graph after removing the edges of C and the $r-t$ isolated nodes. We obtain t cacti; let a_i and b_i denote the corresponding $|R_1|$ and $|R_2|$ values for $i = 1, \ldots, t$. By

induction, $b_i \leq 4a_i - 8$ holds for each of them, giving

$$\sum_{i=1}^{t} b_i \leq \sum_{i=1}^{t}(4a_i - 8) = 4\sum_{i=1}^{t}(a_i - 1) - 4t. \tag{2}$$

Observe that $|R_2| \leq \sum_{i=1}^{t} b_i + t$, since the only nodes of R_2 that are possibly not accounted for in any of the smaller cacti are the t nodes where these cacti are incident to C. Also, $|R_1| \geq \sum_{i=1}^{t}(a_i - 1) + r - t$, since we remove at most one node of degree 2 from each component and add $r - t$ new ones. Adding up the inequalities we obtain

$$|R_2| \leq \sum_{i=1}^{t} b_i + t \leq 4\sum_{i=1}^{t}(a_i - 1) - 3t$$
$$\leq 4(|R_1| + t - r) - 3t = 4|R_1| + t - 4r \leq 4|R_1| - 8$$

The second inequality holds by (2), and the last one uses $8 \leq 4r - t$ that is valid since $t \leq r$ and $r \geq 3$. □

Observe that every node in R_1 forms a singleton minimum cut. Hence if $|R_1| > 2p$, we may conclude infeasibility. Otherwise, Proposition 3.9 gives $|R| \leq 10p - 8$.

We prove the analogue of Theorem 3.6: we show that it is sufficient to consider only links incident to R. It follows that we can obtain a kernel on at most $10p - 8$ nodes by replacing every path consisting of 2-circuits by a single 2-circuit. The number of links in the kernel will again be $O(p^3)$.

THEOREM 3.10. *For a metric instance $(V, E, E^*, c, w, 3, p)$, there exists an optimal solution F such that every edge in F is only incident to corner nodes.*

PROOF. The proof goes along the same lines as that of Theorem 3.6. For every link f, let $\ell(f) = |\mathcal{D}(f)|$. Consider an optimal solution F such that $|F|$ is minimal, and subject to this, $\ell(F) = \sum_{f \in F} \ell(f)$ is minimal. We show that no link in this set F can be incident to a node in T.

For a contradiction, assume $f = (u, y) \in F$ has an endnode $y \in T$. Node y is incident to two 2-circuits; let us denote these by C_x and C_z, with C_x consisting of two parallel edges between x and y and C_z between y and z. Clearly, f covers exactly one of the corresponding two cuts. W.l.o.g. assume that the cut corresponding to C_x is in $\mathcal{D}(f)$; note that this implies that x separates u and y. Since $(V, E \cup F)$ is 3-edge-connected, there must be a link $e \in F$ such that the cut corresponding to C_z is in $\mathcal{D}(e)$. The two cases whether the cut corresponding to C_x is in $\mathcal{D}(e)$ lead to contradictions the same way as in the proof of Theorem 3.6, using Proposition 3.8. □

3.4. Augmenting edge-connectivity for higher values

In this section, we assume that the input graph $G = (V, E)$ is already $(k-1)$-connected, where k is the connectivity target. We show that for even or odd k, the problem can be reduced to the $k = 2$ or the $k = 3$ case, respectively.

Assume first that k is even. We use the following simple structure theorem, which is based on the observation that if the minimum cut value in a graph is odd, then the family of minimum cuts is cross-free. (A set system on V is cross-free if it does contain two elements A and B such that $A \cap B \neq \emptyset$, $A \setminus B \neq \emptyset$, $B \setminus A \neq \emptyset$, and $V \setminus (A \cup B) \neq \emptyset$.)

THEOREM 3.11 ([FRANK 2011, THM 7.1.2]). *Assume that the minimum cut value $k - 1$ in the graph $G = (V, E)$ is odd. Then there exists a tree $H = (U, L)$ along with a map $\varphi : V \to U$ such that the min-cuts of G and the edges of H are in one-to-one correspondence: for every edge $e \in L$, the pre-images of the two components of $H - e$ are*

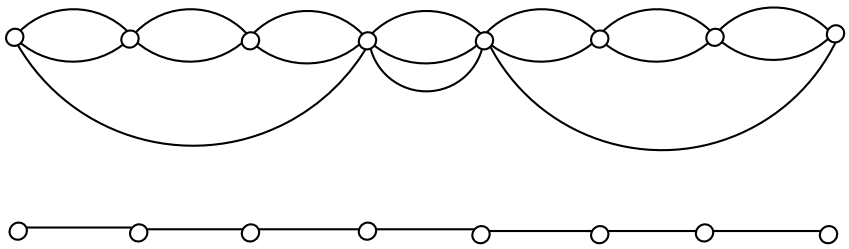

Fig. 4. Illustration of Theorem 3.11 for $k = 4$. The above graph is mapped to the path below with a bijection between the nodes.

the sides of the corresponding min-cut, and every minimum cut can be obtained this way.

Note that Theorem 3.11 *does not* say that G is somehow a tree with duplicated edges: it is possible x and y are adjacent in G even if $\phi(x)$ and $\phi(y)$ are not adjacent in the tree H (see Figure 4).

For even $k-1$, the following theorem shows that the minimum cuts can be represented by a cactus. Note that the theorem also holds for odd $k-1$; however, in this case it is easy to see that the cactus arises from a tree by doubling all edges and hence obtaining Theorem 3.11.

THEOREM 3.12 ([DINITS ET AL. 1976], [FRANK 2011, THM 7.1.8]). *Consider a loopless graph $G = (V, E)$ with minimum cut value $k-1$. Then there exists a cactus $H = (U, L)$ along with a map $\varphi : V \to U$ such that the min-cuts of G and the edges of H are in one-to-one correspondence. That is, for every minimum cut $X \subseteq U$ of H, $\varphi^{-1}(X)$ is a minimum cut in G, and every minimum cut in G can be obtained in this form.*

Observe that if G does not contain k-inseparable pairs (e.g., it was obtained by contracting all the maximal k-inseparable sets), then φ in Theorems 3.11 and 3.12 is one-to-one: $\varphi(x) = \varphi(y)$ would mean that there is no minimum cut separating x and y. Therefore, in this case Theorems 3.11 and 3.12 imply that we can replace the graph with a tree or cactus graph H in a way that the minimum cuts are preserved. Note that the *value* of the minimum cut does change: it becomes 1 (if H is a tree) or 2 (if H is a cactus), but $X \subseteq V$ is a minimum cut in G if and only if it is a minimum cut in H. The proofs of the above theorems also give rise to polynomial time algorithms that find the tree or cactus representations efficiently. Let us summarize the above arguments.

LEMMA 3.13. *Let $G = (V, E)$ be a $(k-1)$-edge-connected graph containing no k-inseparable pairs. Then in polynomial time, one can construct a graph $H = (V, L)$ on the same node set having exactly the same set of minimum cuts such that*

(1) if k is even, then H is a tree (hence the minimum cuts are of size 1), and
(2) if k is odd, then H is a cactus (hence the minimum cuts are of size 2).

Now we are ready to show that if G is $(k-1)$-edge-connected, then a kernel containing $O(p)$ nodes, $O(p)$ edges, and $O(p^3)$ links is possible for every k. First, we contract every maximal k-inseparable set; if multiple links are created between two nodes with the same weight, let us only keep one with minimum cost. By Proposition 2.1, this does not change the problem. Then we can apply Lemma 3.13 to obtain an equivalent problem on graph H having a specific structure. If k is even, then covering the $(k-1)$-cuts of G is equivalent to covering the 1-cuts of the tree H, that is, augmenting the connectivity of G to k is equivalent to augmenting the connectivity of H to 2. Therefore, we can use the algorithm described in Section 3.2 to obtain a kernel. If k is odd, then

covering the $(k-1)$-cuts of G is equivalent to covering the 2-cuts of the cactus H, that is, augmenting the connectivity of G to k is equivalent to augmenting the connectivity of H to 3. In this case, Section 3.3 gives a kernel.

3.5. Decreasing the size of the cost

We have shown that for arbitrary instance (V, E, E^*, c, w, k, p), if (V, E) is $(k-1)$-edge-connected, then there exists a kernel on $O(p)$ nodes and $O(p^3)$ links. However, the costs of the links in this kernel can be arbitrary rational numbers (assuming the input contained rational entries).

We show that the technique of Frank and Tardos [1987] is applicable to replace the cost by integers whose size is polynomial in p and the instance remains equivalent to the original one.

THEOREM 3.14 ([FRANK AND TARDOS 1987]). *Let us be given a rational vector $c = (c_1, \ldots, c_n)$ and an integer N. Then there exists an integral vector $\bar{c} = (\bar{c}_1, \ldots, \bar{c}_n)$ such that $||\bar{c}||_\infty \leq 2^{4n^3} N^{n(n+2)}$ and sign$(c \cdot b)$ = sign$(\bar{c} \cdot b)$, where b is an arbitrary integer vector with $||b||_1 \leq N-1$. Such a vector $\bar{c}$ can be constructed in polynomial time.*

In our setting, $n = O(p^3)$ is the length of the vector. We want to modify the cost function c to obtain a new cost function $\bar{c}$ with the following property: for arbitrary two sets of links F, F' with $|F|, |F'| \leq p$, we have $c(F) < c(F')$ if and only if $\bar{c}(F) < \bar{c}(F')$. This can be guaranteed by requiring that $\text{sign}(c \cdot b) = \text{sign}(\bar{c} \cdot b)$ for every vector b containing at most $2p$ nonzero coordinates, all of them being 1 or -1. Thus it is sufficient to consider vectors b with $||b||_1 \leq 2p$, giving $N = 2p+1$. Therefore Theorem 3.14 provides a guarantee $||\bar{c}||_\infty \leq 2^{O(p^6)}(2p+1)^{O(p^6)}$, meaning that each entry of $\bar{c}$ can be described by $O(p^6 \log p)$ bits. An optimal solution for the cost vector $\bar{c}$ will be optimal for the original cost c. This completes the proof of Theorem 1.1.

Remark 3.15. The above construction works for *Weighted Minimum Cost Edge Connectivity Augmentation* defined as an optimization problem. However, parametrized complexity theory traditionally addresses decision problems. The corresponding decision problem further includes a value $\alpha \in \mathbb{R}$ in the input, and requires to decide whether there exists an augmenting edge set of weight at most p and cost at most α. For this setting, we can apply the Frank-Tardos algorithm for the vector (c, α) instead of c; this gives the same complexity bound $O(p^6 \log p)$.

3.6. Unweighted problems (Proof of Theorem 1.2)

In this section we show how Theorem 1.2 for unweighted instances can be deduced from Theorem 1.1.

Consider an instance of *Minimum Cost Edge-Connectivity Augmentation by One*: let $G = (V, E)$ be a $(k-1)$-edge-connected graph, and let E_0^* be a set of (unweighted) links with cost vector c. We may take it as an instance of *Weighted Minimum Cost Edge-Connectivity Augmentation by One*, setting the weights of all links to 1. Theorem 1.1 then returns a kernel with $O(p)$ nodes and $O(p^3)$ links.

The first step in constructing the kernel was Lemma 3.13, which obtained an equivalent problem instance with the input $G = (V, E)$ being a tree or a cactus, and the connectivity target $k = 2$ or $k = 3$, respectively. Let $R \subseteq V$ denote the set of corner nodes as in Sections 3.2 and 3.3, respectively; let $T = V \setminus R$. The kernel graph is obtained from G by contracting all paths of degree 2 nodes to single edges in trees, and all paths of 2-circuits to single 2-circuits in cacti. This was possible because in the metric closure, we can always find an optimal solution using links between corner nodes only (Theorems 3.6 and 3.10).

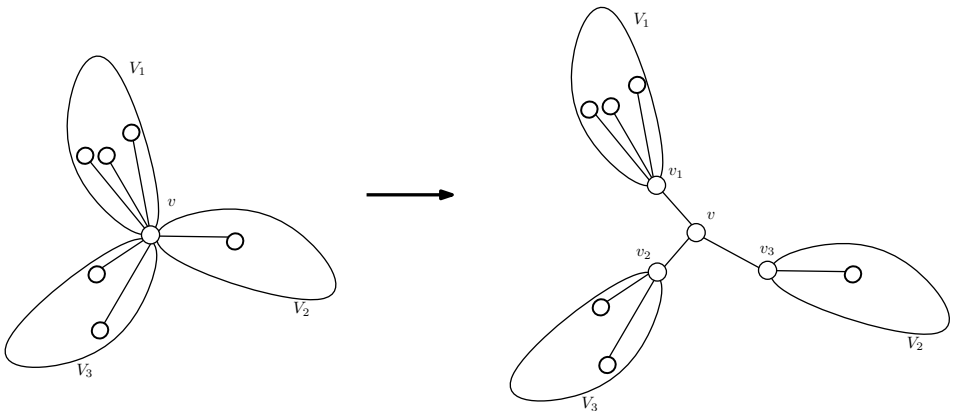

Fig. 5. The node splitting operation.

Let c denote the original cost function and $\bar{c}$ the one obtained by *Metric-Closure*(c). Consider now a link in the kernel; it corresponds to a link f in the metric closure in G. Let us say that a set of (unweighted) links $A \subseteq E_0^*$ *emulates* a link f in the metric closure, if

— $|A| \leq w(f)$,
— $\sum_{e \in A(f)} c(f) \leq \bar{c}(f)$, and
— $\cup_{e \in A(f)} \mathcal{D}(e) \supseteq \mathcal{D}(f)$.

We show that for every link f in the metric closure, there exists a set $A(f)$ emulating it. Indeed, we follow the steps of algorithm *Metric-Closure*(c), and maintain a set $A(f)$ emulating every link f. This is initialized as $A(f) = \{f\}$ for every link. If $c(h)$ is replaced by $c(e) + c(f)$, then replace $A(h)$ by $A(e) \cup A(f)$. If f is a shadow of e and $c(f)$ is replaced by $c(e)$, then replace $A(f)$ by $A(e)$. By induction it is easy to see that $A(f)$ will be a set emulating f in every step. In every optimal solution, we may replace f by the set of links $A(f)$ maintaining optimality. Then $|A(f)| \leq p$ follows from $w(f) \leq p$.

We have shown that the $O(p^3)$ links in the weighted kernel may be replaced by $O(p^4)$ original links. This also increases the number of nodes and edges in the kernel, as we must keep all nodes in T incident to these links. The bound $O(p^8 \log p)$ on the bit sizes easily follows as in Section 3.5.

3.7. Node-connectivity augmentation

Consider an instance $(V, E, E^*, c, w, 2, p)$ of *Weighted Minimum Cost Node-Connectivity Augmentation from 1 to 2*. We reduce it to an instance of *Weighted Minimum Cost Edge-Connectivity Augmentation from 1 to 2* via a simple and standard construction.

Let $N \subseteq V$ denote the set of cut nodes in $G = (V, E)$. Let us perform the following operation for every $v \in N$ (illustrated on Figure 5). Let $V_1, \ldots, V_r$ denote the node sets of the connected components of $G - v$; $r \geq 2$ as v is a cut node. Let us add r new nodes $v_1, v_2, \ldots, v_r$, connected to v. Replace every edge $uv \in E$ with $u \in V_i$ by uv_i and similarly every link (u, v) with $u \in V_i$ by a link (u, v_i) of the same cost and weight. Note that there are exactly r edges and no links incident to v after this operation. Let us call the vv_i edges *special edges*.

Let $G' = (V', E')$ denote the resulting graph after performing this for every $v \in N$. For a link set F, let $\varphi(F)$ denote its image after these operations. The following lemma shows the reduction to the *Weighted Minimum Cost Edge-Connectivity Augmentation from 1 to 2* problem.

LEMMA 3.16. *Graph $(V, E \cup F)$ is 2-node-connected if and only if $(V', E' \cup \varphi(F))$ is 2-edge-connected.*

PROOF. Consider first a link set F such that $(V, E \cup F)$ is 2-node-connected. Assume that there is a cut edge in $(V', E' \cup \varphi(F))$. If it is an edge $e \in E'$ that is an image of an original edge from E, then it is easy to verify that e must also be a cut edge in $(V, E \cup F)$. If the cut edge is some edge vv_i added in the construction, then V_i is disconnected from the rest of the graph in $(V, E \cup F) - v$. The converse direction follows by the same argument. □

It is left to prove that a kernel (V'', E'') for the edge-connectivity augmentation problem can be transformed to a kernel of the node-connectivity augmentation problem. Graph (V'', E'') was obtained by first contracting the maximal 2-inseparable sets, then contracting all paths of degree 2 nodes in the resulting tree. In the first step, no special edges can be contracted, since v and v_i are not 2-inseparable. Also, if v was an original cut node, then, after the transformation, no link is incident to v. Is is not difficult to see that contracting all special edges in (V'', E'') gives an equivalent node-connectivity augmentation problem.

4. AUGMENTING ARBITRARY GRAPHS TO 2-EDGE-CONNECTIVITY

In this section, we allow an arbitrary input graph; by Proposition 2.1, we may assume that $G = (V, E)$ is a forest with $r > 1$ components, denoted by (V_1, E_1), (V_2, E_2), $\ldots$, (V_r, E_r) (we also consider the isolated nodes as separate components, hence $V = \cup_{i=1}^{r} V_i$). There are two types of links in E^*: $e = (u, v)$ is an *internal link* if u and v are in the same component and *external link* otherwise.

In the following, we allow adding multiple copies of the same link. Doing this can make sense if the link connects two different components: then the two copies of the same link provides 2-edge-connectivity between the two components. However, the problem was originally defined such that multiple copies of the same link cannot be taken into the solution. In Section 4.3, we describe a clean reduction how to enforce that there can be only one copy of each link in the solution.

On a high level, we follow the same strategy as in Section 3.2: we define an appropriate notion of metric instances, and show that every input instance can be reduced efficiently to an equivalent metric one. However, this reduction is more involved than the reduction for connected inputs. We are only able to establish a fixed-parameter algorithm for metric instances, but we are unable to construct a polynomial kernel. In Section 4.1, we will show how to reduce the problem from arbitrary instances to metric ones. Then in Section 4.2, we exhibit the FPT algorithm for metric instances. The following propositions and definitions are needed for the definition of metric instances.

PROPOSITION 4.1. *Graph $(V, E \cup F)$ is 2-edge-connected if and only if it is connected and for every edge $e \in E \cup F$, there is a circuit in $E \cup F$ containing it.*

As before, if f is an internal link connecting two nodes in V_i, let $P(f)$ denote the unique path between the endpoints of f in E_i. We also say that the node y *lies between* the nodes x and z if x, y and z are in the same component, and y is contained in the unique path between x and z in this component ($y = x$ or $y = z$ is possible). Furthermore, the edge $uv \in E$ is between x and z, if it lies on the unique path between x and y in E (equivalently, both u and v are between x and z). We will use the following fundamental property of circuits in a graph, the so-called strong circuit axiom in matroid theory.

PROPOSITION 4.2. *Let C and C' be two circuits in a graph with $f \in C \cap C'$ and $g \in C \setminus C'$. Then there exists a circuit C'' with $C'' \subseteq C \cup C'$, $g \in C''$ and $f \notin C''$.*

Assume that the graph $(V, E \cup F)$ is 2-edge-connected. By Proposition 4.1, for every $e \in E \cup F$ there exists a circuit in $E \cup F$ containing e. Let $C(e)$ denote such a circuit containing e with $|C(e) \cap F|$ minimum (that is, $C(e)$ contains a minimum number of links); if there are more than one, pick such a circuit arbitrarily.

PROPOSITION 4.3. *For every $e \in E \cup F$, consider the circuit $C(e)$. Then for every $1 \leq i \leq r$, if $C(e)$ intersects (V_i, E_i), then the intersection is a path (possibly a single node), and $C(e)$ contains either a single internal link between two nodes in V_i or exactly two external links incident to V_i.*

PROOF. First, assume $C(e)$ contains an internal link f incident to V_i. If $e = f$ is itself this internal link, then $C(e)$ must consist of e and the unique path $P(e)$ in E_i connecting the two endpoints of e. Indeed, this circuit contains the minimum number of links (one), and furthermore there is no other circuit in $E \cup \{e\}$; hence $C(e)$ is uniquely defined in this case.

Assume therefore $e \neq f$, and consider the circuit $C(f)$. The previous argument shows that $C(f)$ consists of f and a path in E_i. If $e \in C(f) \cap E_i$, then either $C(e) = C(f)$, or by the minimal choice, $C(e)$ is another circuit containing only one link. This is only possible if $C(e)$ also comprises an internal link and the path in E_i between its endpoints, proving the claim. If $e \notin C(f)$, then we can apply Proposition 4.2 to $C(e)$ and $C(f)$. This gives a circuit $C \subseteq C(e) \cup C(f)$, $e \in C$, $f \notin C$, contradicting the fact that $C(e)$ contained the minimum number of links.

Hence we may assume that $C(e)$ contains no internal links; assume it has some external links incident to V_i. Let $C(e)$ be of the form $P_1 - f_1 - P_2 - f_2 - \ldots - P_t - f_t$, where $f_1, \ldots, f_t$ are the external links incident to V_i, and $P_1, \ldots, P_j$ are the paths on $C(e)$ between two subsequent f_j's. If $t = 2$, then the intersection between $C(e)$ and (V_i, E_i) must clearly be a path and hence the claim follows. Assume now $t > 2$, and that $e \in P_1 \cup \{f_1\}$. Let Q denote the path in E_i between the endpoints of f_1 and f_t. Now $f_1 - Q - f_t - P_1$ gives a circuit in $E \cup F$ containing e, a contradiction to the choice of $C(e)$. □

To define the notion of shadows in this setting, we first need the analogues of $P(f)$ for external links. This motivates our next definition. Consider a leaf u in a tree (V_i, E_i) and let (V_j, E_j) be a different component. For some $1 \leq t \leq p$, let $S_t(u, V_j)$ denote the endpoint of a cheapest link between u and a node in V_j of weight at most t, that is

$$S_t(u, V_j) = \text{argmin}_z\{c(f) : f \text{ is an } (u, z) \text{ link}, z \in V_j, w(f) \leq t\}.$$

If no such (u, z) link exists, then $S_t(u, V_j)$ will not be defined. If there are multiple possible choices, pick one arbitrarily. We say that the external link $f = (u, v)$ is *foliate* if one of its endpoints, say u, is a leaf in one of the components. Shadows will be defined for internal links and foliate external links only. All other external links are only shadows of themselves.

Definition 4.4. Consider two links e and f, with $w(f) \geq w(e)$. We say that f is a *shadow* of e in either of the following cases.

— $e = f$;
— e and f are both internal links in the same component and $P(f) \subseteq P(e)$;
— $e = (u, x)$, $f = (u, y)$ are two foliate external links for a leaf u, and y is between x and $S_t(u, V_j)$, where $t = w(e)$, and $x, y \in V_j$.

The definition is illustrated in Figure 6. Given this notion, the definition of metric instances is identical as in Section 3.1. We say that the instance is *metric*, if

(1) $c(f) \leq c(e)$ holds whenever the link f is a shadow of link e.

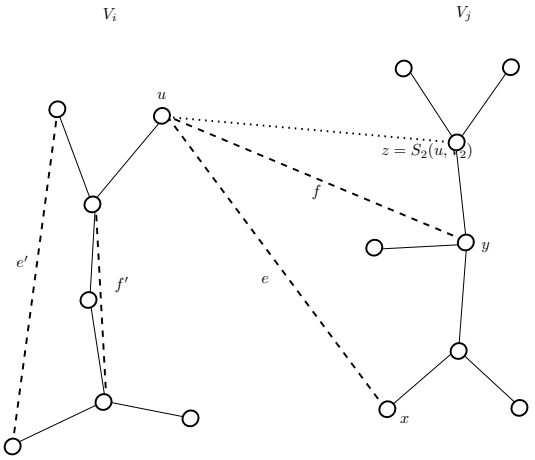

Fig. 6. The external link f is a shadow of the external link e, and the internal link f' is a shadow of the internal link e'.

(2) Consider three links $e = (u, v)$, $f = (v, z)$ and $h = (u, z)$ with $w(h) \geq w(e) + w(f)$. Then $c(h) \leq c(e) + c(f)$.

4.1. Computing the metric completion

We use the algorithm *Metric-Completion*(c) identical to the one in Figure 1, with the meaning of shadows modified. A technical difficulty is that the definition of shadow for external links involve the nodes $S_t(u, V_j)$, whose definition depends on the cost function, hence can change during the computation of the metric completion. Moreover, the definition of $S_t(u, V_j)$ might involve an arbitrary choice if there are multiple cheapest t-links. We use the following convention: while modifying the cost function c, we modify the nodes $z = S_t(u, V_j)$ only if necessary. That is, only if after the modification, link (u, z) is not among the cheapest t-links between u and V_j anymore. We next prove that Lemma 3.3 is still valid. The algorithm *Metric-Completion*(c) will again run in polynomial time, since the number of triangle inequalities will be $O(p^3n^3)$ and every link may be a shadow of at most $O(pn^2)$ other ones.

LEMMA 4.5. *Consider a problem instance* $(V, E, E^*, c, w, 2, p)$. *The algorithm* Metric-Completion(c) *returns a metric cost function* $\bar{c}$ *with* $\bar{c}(e) \leq c(e)$ *for every link* $e \in E^*$. *Moreover, if for a link set* $\bar{F} \subseteq E^*$, $(V, E \cup \bar{F})$ *is 2-edge-connected, then there exists an* $F \subseteq E^*$ *such that* $(V, E \cup F)$ *is 2-edge-connected,* $c(F) \leq \bar{c}(\bar{F})$ *and* $w(F) \leq w(\bar{F})$. *Consequently, and optimal solution for* $\bar{c}$ *provides an optimal solution for* c.

PROOF. The proof of the metric property of $\bar{c}$ is almost identical to that in Lemma 3.3. We need only one additional observation: after fixing the triangle inequalities in iteration t, the nodes $S_t(u, V_j)$ cannot change anymore. This is because all shadows of links between u and V_j are also links between u and V_j, hence we cannot decrease the cost of the cheapest such link in the second part of phase t. Therefore, it follows that the shadow relations for links of weight $\leq t$ are unchanged during and after the second part of iteration t and this relation is transitive.

For the second part, it is again enough to verify the claim for the case when $\bar{c}$ arises by a single modification from c. First, assume the modification is fixing a triangle inequality $c(h) > c(e) + c(f)$ by setting $\bar{c}(h) = c(e) + c(f)$ and $\bar{c}(g) = c(g)$ for every $g \neq h$. We again set $F = \bar{F}$ if $h \notin \bar{F}$ and $F = (\bar{F} \setminus \{h\}) \dot{\cup} \{e, f\}$ otherwise. The only difference is that F is a multiset (as in this section we assume that a link can be selected into

the solution twice) and $\dot\cup$ denotes disjoint union, i.e. if e or f was already present in F, then we keep the old copies as well; but the same analysis carries over.

Next, assume $\bar{c}(f) = c(e)$ was set in the second part of iteration t, and $\bar{c}(g) = c(g)$ for every $g \neq f$. If f is an internal link, it is easy to verify that replacing f by e retains 2-edge-connectivity.

Let us now focus on the case when f is a foliate external link, and $f \in \bar{F}$. Let $e = (u, x)$, $f = (u, y)$, $t = w(f)$, with u being a leaf, and $x, y \in V_j$ for a component not containing u; let $z = S_t(u, V_j)$. Let $h = (u, z)$ denote a cheapest t-link between u and V_j. As f is a shadow of e, the node y appears on the path between x and z in E_j.

Let $F_e = (\bar{F} \dot\cup \{e\}) \setminus \{f\}$ and $F_h = (\bar{F} \dot\cup \{h\}) \setminus \{f\}$. We aim to prove that either $E \cup F_e$ or $E \cup F_h$ is 2-edge-connected. Let us say that an edge in $(E \cup \bar{F}) \setminus \{f\}$ is *e-critical* or *h-critical*, if it is a cut edge in $E \cup F_e$ or in $E \cup F_h$, respectively. We call an edge *critical* if it is either of the two.

CLAIM 4.6. *If g is e-critical, then it must lie on the path in E_j between x and y. If g is h-critical, then it must lie on the path in E_j between y and z.*

PROOF. We prove for the e-critical case; the same argument works when e is h-critical. Consider the circuit $C(g)$ containing a minimum number of links as in Proposition 4.3. For g to become a cut edge in $E \cup F_e$, we must have $f \in C(g)$. Let C' denote the circuit consisting of the links $e = (u, x)$, $f = (u, y)$ and the $x - y$ path on E_j. If the latter does not contain g, then we may use Proposition 4.2 for $C(g)$ and C' to obtain a circuit $C'' \subseteq C(g) \cup C'$ with $g \in C''$, $f \notin C''$. The existence of such a C'' contradicts our assumption that g is a cut edge in $E \cup F_e$. □

CLAIM 4.7. *Either there exist no e-critical edges or there exist no h-critical edges.*

PROOF. For a contradiction, assume that there exists an e-critical edge g_e and an h-critical g_h. Consider the circuits $C(g_e)$ and $C(g_h)$ containing the minimum number of links as in Proposition 4.3; for the critical property, both of them must contain f. By Claim 4.6, both $g_e, g_h \in E_j$; g_e lies on the $x - y$ path, and g_h lies on the $y - z$ path. Then Proposition 4.3 implies that circuit $C(g_e)$ must be disjoint from the $y - z$ path in E_j and $C(g_h)$ must be disjoint from the $x - y$ path. Hence $g_h \notin C(g_e)$ and $g_e \notin C(g_h)$. Using Proposition 4.2, we get a circuit $C \subseteq (C(g_e) \cup C(g_h)) \setminus \{f\}$ and $g_e \in C$. This circuit is contained in $E \cup F_e$, a contradiction to the fact that g_e is e-critical. □

This claim completes the proof, showing that f can be exchanged to either e or h. (It is easy to check that e or h itself cannot become a cut edge, as it would imply that f was a cut edge in $E \cup F$). □

4.2. FPT algorithm for metric instances

In this section, we assume that problem instance $(V, E, E^*, c, w, 2, p)$ is metric. Let R again denote the set of *corner nodes*, that is, nodes of degree not equal to 2. Again, if there are more than $2p$ leaves, then the problem is infeasible; otherwise, $|R| \leq 4p - 2$. For a leaf u in the tree (V_1, E_1), let

$$\mathcal{S}_u = \{v \in V : v = S_t(u, V_j) \text{ for some } 1 \leq t \leq p, 2 \leq j \leq r\}.$$

Note that $|\mathcal{S}_u| < p^2$, since $r \leq p$ (every component contains at least two leaves). The following theorem gives rise to a straightforward FPT algorithm.

THEOREM 4.8. *Consider a metric instance $(V, E, E^*, c, w, 2, p)$, and let u be a leaf in the tree (V_1, E_1). There exists an optimal solution solution F such that for every link $f = (u, v) \in F$, it holds that $v \in R \cup \mathcal{S}_u$.*

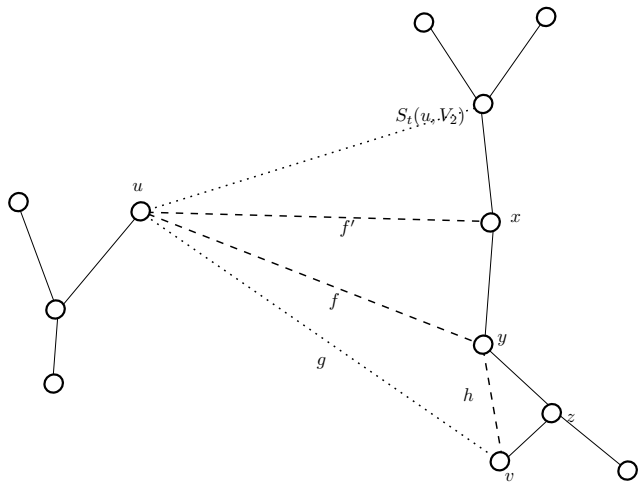

Fig. 7. Illustration of the proof of Theorem 4.8.

Given this theorem, the FPT algorithm is as follows. If the number of leaves is more than $2p$, we terminate by concluding infeasibility. Otherwise, we pick an arbitrary leaf u in the first tree. We branch according to all possible incident links connecting it to one of the corner nodes or to the elements of $\mathcal{S}_u$. This is altogether $O(p)$ nodes with p possible links connecting them to u, giving $O(p^2)$ branches. This gives an algorithm with running time $(p^2)^p = 2^{O(p \log p)}$, proving Theorem 1.4.

PROOF OF THEOREM 4.8. For an internal link $f = (u, v)$ incident to u, let $\ell(f) = |P(f)|$. For a foliate link $f = (u, v)$ incident to u, let (V_j, E_j) be the component containing v. Let $\ell(f)$ denote the length of the unique path in E_j between v and $S_t(u, V_j)$. For all other external links, let $\ell(f) = 0$. Consider an optimal solution F such that $|F|$ is minimal, and subject to this, $\ell(F) = \sum_{f=(u,v)\in F} \ell(f)$ is minimal.

For a contradiction, consider a link $f = (u, y)$ with $y \notin R \cup \mathcal{S}_u$. Let $t = w(f)$. If f is an internal link, let x be the neighbour of y between u and y. If f is external, w.l.o.g. assume $y \in V_2$; in this case, let x be the neighbour of y closer to $S_t(u, V_2)$. In both cases, let z be the other neighbour of y in E_1 or in E_2, which is uniquely defined since y has degree 2. Note that $\ell(f)$ is the length of the path between u and y in E_1 or between $S_t(u, V_2)$ and y in E_2. The external case is illustrated in Figure 7; for the internal case, see Figure 2 in Section 3.2.

CLAIM 4.9. *For any circuit $C \subseteq E \cup F$ with $xy \in C$, we must have $f \in C$.*

PROOF. For a contradiction, assume there exists a circuit C with $xy \in C$, $f \notin C$. Let $f' = (u, x)$ be a t-link, and consider $F' = (F \setminus \{f\}) \cup \{f'\}$. Link f' is a shadow of f and hence $c(f') \leq c(f)$, that is, $c(F) \leq c(F')$; further, $\ell(f') = \ell(f) - 1$. We claim that $E \cup F'$ is also 2-edge-connected, thereby contradicting the minimal choice of $\ell(F)$. The edge xy is not a cut edge, as witnessed by the circuit C not containing f. For any other edge $e \in (E \cup F) \setminus \{f\}$, we know that there is a circuit $C(e) \subseteq E \cup F$ containg e. If $f \notin C(e)$, then $C(e) \subseteq E \cup F'$ as well. In the sequel, assume $f \in C(e)$. If $xy \in C(e)$, then f and xy can be replaced in $C(e)$ by f', giving a circuit in $E \cup F'$ containing e. On the other hand, if $xy \notin C(e)$, then we can replace f by f' and xy. We can show in a similar way that f' cannot be a cut edge either: given a circuit of $E \cup F$ containing f, we can either replace f by f' and xy, or replace f and xy by f' to obtain a circuit in $E \cup F'$ containing f'. □

Consider now the edge $yz \in E$, and let $C(yz)$ be a circuit in $E \cup F$ containing yz and having a minimal number of links. Let $C(xy)$ be the analogous circuit for xy; the previous claim implies $f \in C(xy)$.

CLAIM 4.10. *We have $xy, f \notin C(yz)$, and there is a link $h = (y, v) \in C(yz) \cap F$.*

PROOF. By Proposition 4.3, $C(yz)$ intersects the component of yz (E_1 or E_2) in a single path P and there are at most two incident links. If $xy \in P$, then by the previous claim, $f \in C(yz)$. Then y has degree 3 in the circuit $C(yz)$, a contradiction. Consequently, the path P must end in y, and hence $C(yz)$ must contain a link h incident to y. The proof is complete by showing $h \neq f$. Indeed, if $h = f$, then we can apply Proposition 4.2 for $C(xy)$ and $C(yz)$ to obtain a circuit C' with $xy \in C'$ and $f \notin C'$, a contradiction to the previous claim. □

The rest of the proof is dedicated to showing that 2-edge-connectivity is maintained if we replace f and h by a (u, v)-link g of weight $w(f) + w(h)$. Since the instance is metric, we must have $c(g) \leq c(f) + c(h)$. Let $F' = (F \cup \{g\}) \setminus \{f, h\}$. Showing that $E \cup F'$ is 2-edge-connected yields a contradiction to the minimal choice of $|F|$. By Proposition 4.1, we have to show that $E \cup F'$ is connected and for each edge there is a circuit containing it. Connectivity follows easily: if $E \cup F'$ became disconnected by removing links $f = (u, y)$ and $h = (y, v)$, and adding (u, v), then node y must lie in a different component than u and v. However, as $f \in C(xy)$ by Claim 4.9, the path $C(xy) \setminus \{f\}$ still appears in $E \cup F'$ and connects the endpoints u and y of f. To verify the existence of a circuit for each edge, we need the following.

CLAIM 4.11. *The only common node of the circuits $C(xy)$ and $C(yz)$ is y.*

PROOF. For a contradiction, assume the two circuits intersect in nodes other than y. Let us start moving on the path $P_0 = C(xy) \setminus \{f\}$ from y until we hit the first node on $C(yz)$; let a be this intersection point and let P_1 be the part of P_0 between y and a. Let P_2 be one of the two parts of $C(yz)$ between a and y. Now $P_1 \cup P_2$ is a circuit containing xy but not f, a contradiction to Claim 4.9. □

Consequently, $\hat{C} = (C(xy) \cup C(yz) \cup \{g\}) \setminus \{f, h\}$ is a circuit in $E \cup F'$ containing g. For an arbitrary $e \in (E \cup F') \setminus \{g\}$, consider the circuit $C(e)$ in $E \cup F$. We are done if $C(e)$ contains neither of f and h. If $C(e)$ contains both f and h, then we can replace these two edges in the circuit with g. Assume $C(e)$ contains exactly one of them, say $f \in C(e)$ (the case $h \in C(e)$ can be proved similarly). If $e \in C(xy)$, then $\hat{C}$ does contain e. Otherwise, if $e \notin C(xy)$, then we may use Proposition 4.2 to obtain a circuit $C' \subseteq C(e) \cup C(xy)$, $f \notin C'$, $e \in C'$. Also, $h \notin C'$ as it was contained in neither $C(e)$ nor $C(xy)$. Now $e \in C' \subseteq E \cup F'$, completing the proof. □

4.3. Forbidding using links twice

The algorithm presented in the previous section solves the version of the problem where we allow taking the same link twice in a solution. Here we show how to solve the original version of the problem, where this is not allowed. We present a solution for the restriction when we do not even allow adding parallel links of different weights between two nodes. The argument can be easily modified to the weaker restriction when we may allow parallel links with different weight.

As before, let $(V_1, E_1), \ldots, (V_r, E_r)$ be the components of the input graph. Note that the components can contain cycles and hence they are not necessarily trees; however, this will not cause any complications for the arguments presented in this section. Let us construct the set S the following way. Start with $S = \emptyset$, and for every $1 \leq i < j \leq r$ and $1 \leq t \leq p$, consider the t-links between V_i and V_j; if there is a *unique* t-link of

minimum cost between these components, then add this link into the set S. If $r > p$, then there exists no feasible solution (as we would need more than p links to connect the components). If $r \leq p$, we have $|S| \leq p^3$.

As a first step of the algorithm, we branch on which subset of S appears in the solution. That is, for every subset $S' \subseteq S$ with $w(S') \leq p$ and not containing any parallel links, we obtain a new graph G' by adding the links in S' to the graph G. Note that adding the set S' can decrease the number of components and can create further cycles. We define a new parameter $p' = p - w(S')$ and define a new cost function c', whose only difference from c is that the cost of every link in $S \setminus S'$ and of every link parallel to a link in S' is ∞. We solve the modified instance for the graph $G' = (V', E') = (V, E \cup S')$, parameter p', and cost function c' using the algorithm of the previous section. If F' is the solution obtained this way, then we return the solution $F = S' \cup F'$. The branching step adds a factor of $O((p^3)^p) = 2^{O(p \log p)}$ to the running time of the algorithm.

It is clear that if the original instance has a solution not containing duplicated links, then no matter which subset of the links S it uses, our algorithm returns a solution with not larger cost. More importantly, we claim that if our algorithm returns a solution using some links twice, then it can be modified such that it does not use any link twice and the cost does not increase. These two statements prove that this algorithm indeed finds an optimum solution for the problem where duplicated links are not allowed. We observe first the following simple lemma:

LEMMA 4.12. *Let $G = (V, E \cup F)$ be a 2-edge-connected graph.* (i) *If F contains two parallel edges with the same endpoints x and y in the same component of (V, E), we may remove one of them without destroying 2-edge-connectivity.* (ii) *Suppose that $e_1, e_2 \in F$ are two parallel links with endpoints x and y in different components of (V, E). Suppose that F contains another link e^* (different from e_1, e_2) whose endpoints are in the same connected components of (V, E) as x and y, respectively. Then $G^* = (V, E \cup (F \setminus e_1))$ is also 2-edge-connected.*

PROOF. The first statement is straightforward. For the second, observe that the edge e_2 is not a cut edge in G^*: there is a circuit containing xy formed by e^*, a path in the component of x, a path in the component of y, and e_2 itself. Moreover, if G^* has a cut edge other than e_2, then it is a cut edge of G as well, a contradiction. □

Note that F' cannot contain links parallel to S' (as the cost of every such link is ∞ in c'), hence parallel links can appear only in F' itself. Suppose that the algorithm finds a multiset F' of links, containing parallel pairs. Consider two links $e_1, e_2 \in F'$ between x and y; let $t = w(e_1)$. If x and y are in the same component of V_i of G, then Lemma 4.12(i) implies that e_1 can be safely removed. Assume therefore that x and y are in two different connected components V_i and V_j of G, respectively. As $e_1 \notin S$, link e_1 is not the unique minimum cost t-link between V_i and V_j in the original instance. Therefore, there is a t-link e^* between V_i and V_j with $c(e^*) \leq c(e_1)$ (note that possibly $e^* \in S \setminus S'$). Link e^* connects the same two connected components of G as e_1. Therefore, if e^* is already in $S' \cup F'$ or there is a link parallel to it in $S' \cup F'$, then Lemma 4.12(ii) implies that removing e_1 from $S' \cup F'$ does not destroy 2-edge-connectivity. Otherwise, we replace e_1 with e^*; the cost of the new solution $S' \cup (F' \setminus e_1) \cup e^*$ obtained this way is not larger than the cost of $S' \cup F'$. Again by Lemma 4.12(ii), removing e_1 from $(V, E \cup S' \cup (F' \cup e^*))$ does not destroy 2-edge-connectivity, i.e., $(V, E \cup S' \cup (F' \setminus e_1) \cup e^*)$ is 2-edge-connected. We repeat this replacement for every duplicated link. Note that this process does not create new duplicated links: we add e^* only if there is no link in $F = S' \cup F'$ with the same endpoints as e^*. Therefore, we obtain a solution having not larger cost and containing no duplicated links.

REFERENCES

A. Agrawal, P. Klein, and R. Ravi. 1995. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *SIAM J. Comput.* 24, 3 (1995), 440–456.

M. Basavaraju, F. V. Fomin, P. Golovach, P. Misra, M.S. Ramanujan, and S. Saurabh. 2014. Parameterized Algorithms to Preserve Connectivity. In *Automata, Languages, and Programming*. Springer, 800–811.

J. Cheriyan and L. A. Végh. 2014. Approximating Minimum-Cost k-Node Connected Subgraphs via Independence-Free Graphs. *SIAM J. Comput.* 43, 4 (2014), 1342–1362.

J. Cheriyan, S. Vempala, and A. Vetta. 2003. An Approximation Algorithm for the Minimum-Cost k-Vertex Connected Subgraph. *SIAM J. Comput.* 32, 4 (2003), 1050–1055.

E.A. Dinits, A.V. Karzanov, and M.V. Lomonosov. 1976. On the structure of a family of minimal weighted cuts in graphs. In *Studies in Discrete Mathematics*, A.A. Fridman (Ed.). Nauka, Moscow, 290–306. In Russian.

R. G. Downey and M. R. Fellows. 1999. *Parameterized Complexity*. Springer, New York. xvi+533 pages.

J. Flum and M. Grohe. 2006. *Parameterized Complexity Theory*. Springer, Berlin. xiv+493 pages.

András Frank. 1992. Augmenting graphs to meet edge-connectivity requirements. *SIAM J. Discret. Math.* 5, 1 (1992), 25–53.

A. Frank. 2011. *Connections in combinatorial optimization*. Number 38 in Oxford lecture series in mathematics and its applications. Oxford Univ Pr.

A. Frank and T. Jordán. 1995. Minimal edge-coverings of pairs of sets. *J. Combin. Theory Ser B* 65, 1 (1995), 73–110.

A. Frank and É. Tardos. 1987. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica* 7, 1 (1987), 49–65.

M.X. Goemans and D.P. Williamson. 1995. A general approximation technique for constrained forest problems. *SIAM J. Comput.* 24, 2 (1995), 296–317.

Jiong Guo and Johannes Uhlmann. 2010. Kernelization and complexity results for connectivity augmentation problems. *Networks* 56, 2 (2010), 131–142.

T. Hsu. 2000. On four-connecting a triconnected graph. *Journal of Algorithms* 35, 2 (2000), 202–234.

B. Jackson and T. Jordán. 2005. Independence free graphs and vertex connectivity augmentation. *J. Combin. Theory Ser B* 94, 1 (2005), 31–77.

K. Jain. 2001. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica* 21, 1 (2001), 39–60.

T. Jordán. 1995. On the optimal vertex-connectivity augmentation. *Journal of Combinatorial Theory, Series B* 63, 1 (1995), 8–20.

G. Kortsarz and Z. Nutov. 2003. Approximating node connectivity problems via set covers. *Algorithmica* 37, 2 (2003), 75–92.

G. Kortsarz and Z. Nutov. 2007. Approximating minimum cost connectivity problems. In *Handbook on Approximation Algorithms and Metaheuristics*, T.F. Gonzalez (Ed.). Chapman & Hall/CRC, London.

Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. 2012. Kernelization - Preprocessing with a Guarantee. In *The Multivariate Algorithmic Revolution and Beyond*. 129–161.

Neeldhara Misra, Venkatesh Raman, and Saket Saurabh. 2011. Lower bounds on kernelization. *Discrete Optimization* 8, 1 (2011), 110–128.

Hiroshi Nagamochi. 2003. An approximation for finding a smallest 2-edge-connected subgraph containing a specified spanning tree. *Discrete Appl. Math.* 126, 1 (2003), 83–113.

Jaroslav Nesetril, Eva Milková, and Helena Nesetrilová. 2001. Otakar Boruvka on minimum spanning tree problem Translation of both the 1926 papers, comments, history. *Discrete Math.* 233, 1-3 (2001), 3–36.

L. A. Végh. 2011. Augmenting undirected node-connectivity by one. *SIAM J. Discrete Math.* 25, 2 (2011), 695–718.

Toshimasa Watanabe and Akira Nakamura. 1987. Edge-connectivity augmentation problems. *J. Comput. Syst. Sci.* 35, 1 (1987), 96–144.