Fixed-parameter algorithms for minimum cost edge-connectivity augmentation[⋆]

Dániel Marx[1] and László A. Végh[2]

[1] Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SZTAKI) Budapest, Hungary dmarx@cs.bme.hu
[2] Department of Management, London School of Economics, London, UK l.vegh@lse.ac.uk

Abstract. 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.

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 efficient electricity network in Moravia [22]. 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 k-edge-connectivity in an undirected graph by adding the minimum number of edges (Watanabe and Nakamura [24], see also Frank [7]). 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 [23]. We refer the reader to the recent book by Frank [8] on more results

[⋆] Full version available on Arxiv:1304.6593. The first author was supported by the European Research Council (ERC) grant *"PARAMTIGHT: Parameterized complexity and the search for tight complexity results,"* reference 280152.

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. [16,13,18,23]), 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. [1,11,15,3,17,2]; for a survey, see [18]. 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 [5,6]. 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 [19,20]. 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 [14], showing that for the problem of making a graph k-node-connected by adding a minimum number of arbitrary 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 edges 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 [21, 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 [12] 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 [21,12] 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. The natural machinery for this approach is to work with a more general problem. Besides the costs, every link is equipped with a positive integer weight. 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 integers of $O(p^6 \log p)$ bits.*

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 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 [4] to show that we can assume 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 [10] (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 the full version), we first obtain a kernel by applying our weighted result to our unweighted instance; this kernel will however contain links of weight higher than one. Still, every link f in the (weighted) kernel can be replaced by a sequence of $w(f)$ original unweighted edges. This replaces the $O(p^2)$ links by $O(p^4)$ original ones.

We try to extend our results in two directions. The results described next are proved only in the full version of the paper. 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.

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 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. [24] and [7] to [9] and [23]), 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 additional branching arguments needed in Theorem 1.4 can show a glimpse of the difficulties one can encounter when trying to solve the problem 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, link f can be automatically substituted by link e. 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.

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 set $X \subseteq V$ and $F \subseteq \binom{V}{2}$, let $d_F(X)$ denote the number of edges $e = uv \in F$ with $u \in X$, $v \in V \setminus X$. When we are given a graph $G = (V, E)$ and it is clear from the context, $d(X)$ will denote $d_E(X)$. A 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 all edges in 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 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 $w(F) \le p$ 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*.

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 $c(e) \le c(f)$ and $w(e) \le w(f)$, then we may also ignore the link f. It is convenient to assume that for every value $1 \le t \le p$ 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, 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 $1 \le t \le p$. However, in the input links of infinite cost should not be listed.

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. If multiple t-links are created between s and another node, we keep only one with minimum cost.

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. 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.*

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. 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

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*

(i) $c(f) \leq c(e)$ holds whenever the link f is a shadow of link e.
(ii) 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) 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. That is, $c(h)$ is set to $\min\{c(h), c(e) + c(f)\}$. Then for every t-link f, we decrease $c(f)$ to the $\min\{c(e) : f \text{ is a shadow of } e\}$. 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; further, 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 $\bar{c}(e) \le c(e)$ for every link $e \in E^*$. Moreover, if for a link set $\bar{F} \subseteq E^*$, 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, $c(F) \le \bar{c}(\bar{F})$, and $w(F) \le w(\bar{F})$. Consequently, an optimal solution for $\bar{c}$ provides an optimal solution for c.*

The proof (see full version) proceeds by showing that after iteration t, all rank t inequalities are satisfied and they remain satisfied later on. 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.) 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 of the tree, that is, $\mathcal{D}$ is in one-to-one correspondence with 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 $|R| \le 4p - 2$, due to the following simple fact.

Proposition 3.4. *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 contracting each path of degree-2 nodes to a single edge. The number of links in the kernel will be $O(p^3)$.

Theorem 3.5. *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.*

The proof (see full version) analyses an optimal solution with the total number of links minimal, and subject to this, the total length of the paths in the tree between the endpoints of the links minimal. Such an optimal solution may contain no links incident to degree 2 nodes.

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 1 gives an example of a cactus.

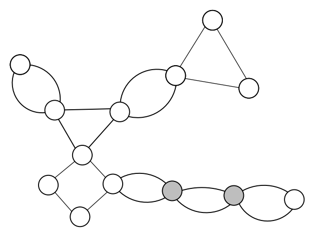

Fig. 1. A cactus graph. The shaded nodes are in the set T.

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 3.6. *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.*

In the rest of the section, we assume $G = (V, E)$ is a cactus. The set of minimum cuts $\mathcal{D}$ corresponds to arbitrary pairs of 2 edges on the same circuit.

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 1). The elements of R will be again called *corner nodes*. We can give the following simple bound:

Proposition 3.7. $|R_2| \leq 4|R_1| - 8$.

Observe that every node in R_1 forms a singleton minimum cut. Hence if $|R_1| > 2p$, we may conclude infeasibility. Otherwise, Proposition 3.7 gives $|R| \leq 10p - 8$.

We prove the analogue of Theorem 3.5: 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 be $O(p^3)$.

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 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.

Theorem 3.8 ([8, Thm 7.1.2]). *Assume 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 the sides of the corresponding min-cut, and every minimum cut can be obtained this way.*

For odd k, the following theorem shows that the minimum cuts can be represented by a cactus.

Theorem 3.9 (Dinits, Karzanov, Lomonosov [4], [8, 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 3.8 and 3.9 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 3.8 and 3.9 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.

Lemma 3.10. *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);*

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 3.10 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 [10] is applicable to replace the cost by integers whose size is polynomial in p and the instance remains equivalent to the original one.

Theorem 3.11 ([10]). *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 $||\bar{c}||_\infty \le 2^{4n^3} N^{n(n+2)}$ and sign$(c \cdot b) =$ sign$(\bar{c} \cdot b)$, where b is an arbitrary integer vector with $||b||_1 \le N-1$. Such a vector $\bar{c}$ can be constructed in polynomial time.*

In our setting, $n = O(p^3)$ is the length of the vector. What we need to guarantee is that for c and $\bar{c}$, $c(F) < c(F')$ if and only if $\bar{c}(F) < \bar{c}(F')$ for arbitrary two sets of links F, F' with $|F|, |F'| \le p$. Hence we need to guarantee the property for vectors b with $||b||_1 \le 2p$, giving $N = 2p+1$. Therefore the theorem provides a guarantee $||\bar{c}||_\infty \le 2^{O(p^6)}(2p+1)^{O(p^6)}$, meaning that the entries 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.

References

1. A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, 1995.
2. J. Cheriyan and L. A. Végh. Approximating minimum-cost k-node connected subgraphs via independence-free graphs. *arXiv preprint arXiv:1212.3981*, 2012.

3. J. Cheriyan, S. Vempala, and A. Vetta. An approximation algorithm for the minimum-cost k-vertex connected subgraph. *SIAM J. Comput.*, 32(4):1050–1055, 2003.
4. E. Dinits, A. Karzanov, and M. Lomonosov. On the structure of a family of minimal weighted cuts in graphs. In A. Fridman, editor, *Studies in Discrete Mathematics*, pages 290–306. Nauka, Moscow, 1976. In Russian.
5. R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Monographs in Computer Science. Springer, New York, 1999.
6. J. Flum and M. Grohe. *Parameterized Complexity Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin, 2006.
7. A. Frank. Augmenting graphs to meet edge-connectivity requirements. *SIAM J. Discret. Math.*, 5(1):25–53, 1992.
8. A. Frank. *Connections in combinatorial optimization.* Number 38 in Oxford lecture series in mathematics and its applications. Oxford Univ Pr, 2011.
9. A. Frank and T. Jordán. Minimal edge-coverings of pairs of sets. *Journal of Combinatorial Theory, Series B*, 65(1):73–110, 1995.
10. A. Frank and É. Tardos. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.
11. M. Goemans and D. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.
12. J. Guo and J. Uhlmann. Kernelization and complexity results for connectivity augmentation problems. *Networks*, 56(2):131–142, 2010.
13. T. Hsu. On four-connecting a triconnected graph. *Journal of Algorithms*, 35(2):202–234, 2000.
14. B. Jackson and T. Jordán. Independence free graphs and vertex connectivity augmentation. *Journal of Combinatorial Theory, Series B*, 94(1):31–77, 2005.
15. K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001.
16. T. Jordán. On the optimal vertex-connectivity augmentation. *Journal of Combinatorial Theory, Series B*, 63(1):8–20, 1995.
17. G. Kortsarz and Z. Nutov. Approximating node connectivity problems via set covers. *Algorithmica*, 37(2):75–92, 2003.
18. G. Kortsarz and Z. Nutov. Approximating minimum cost connectivity problems. In T. Gonzalez, editor, *Handbook on Approximation Algorithms and Metaheuristics*. Chapman & Hall/CRC, London, 2007.
19. D. Lokshtanov, N. Misra, and S. Saurabh. Kernelization - preprocessing with a guarantee. In *The Multivariate Algorithmic Revolution and Beyond*, pages 129–161, 2012.
20. N. Misra, V. Raman, and S. Saurabh. Lower bounds on kernelization. *Discrete Optimization*, 8(1):110–128, 2011.
21. H. Nagamochi. An approximation for finding a smallest 2-edge-connected subgraph containing a specified spanning tree. *Discrete Applied Mathematics*, 126(1):83–113, 2003.
22. J. Nesetril, E. Milková, and H. Nesetrilová. Otakar Boruvka on minimum spanning tree problem translation of both the 1926 papers, comments, history. *Discrete Mathematics*, 233(1-3):3–36, 2001.
23. L. A. Végh. Augmenting undirected node-connectivity by one. *SIAM Journal on Discrete Mathematics*, 25(2):695–718, 2011.
24. T. Watanabe and A. Nakamura. Edge-connectivity augmentation problems. *J. Comput. Syst. Sci.*, 35(1):96–144, 1987.