Kernelization and Complexity Results for Connectivity Augmentation Problems

Jiong Guo* and Johannes Uhlmann**

Institut für Informatik, Friedrich-Schiller-Universität Jena,
Ernst-Abbe-Platz 2, D-07743 Jena, Germany
{guo,uhlmann}@minet.uni-jena.de

Abstract. Connectivity augmentation problems ask for adding a set of at most k edges whose insertion makes a given graph satisfy a specified connectivity property, such as bridge-connectivity or biconnectivity. We show that, for bridge-connectivity and biconnectivity, the respective connectivity augmentation problems admit problem kernels with $O(k^2)$ vertices and links. Moreover, we study partial connectivity augmentation problems, naturally generalizing connectivity augmentation problems. Here, we do not require that, after adding the edges, the entire graph should satisfy the connectivity property, but a large subgraph. In this setting, two polynomial-time solvable connectivity augmentation problems behave differently, namely, the *partial* biconnectivity augmentation problem remains polynomial-time solvable whereas the *partial* strong connectivity augmentation problem becomes W[2]-hard with respect to k.

1 Introduction

Connectivity augmentation problems on undirected and directed graphs have as input a graph $G = (V, E)$, a set E' of edges, and a non-negative integer k, and ask for a set E'' of at most k edges from E' such that $(V, E \cup E'')$ satisfies a specified connectivity property. The edges in E' are called the *links*. Eswaran and Tarjan [2] introduced connectivity augmentation problems and described their numerous applications.

We use $G = (V, E)$ and $D = (V, A)$ to denote undirected and directed graphs. A *path* from vertex u_1 to vertex u_l in $G = (V, E)$ (or $D = (V, A)$) is a sequence of edges $\{u_1, u_2\}$, $\{u_2, u_3\}$, ..., $\{u_{l-1}, u_l\}$ (or arcs (u_1, u_2), (u_2, u_3), ..., (u_{l-1}, u_l)). A *cycle* is a path with $u_1 = u_l$.

A vertex u in an undirected graph is called a *cut-vertex* if there are two vertices v, w with $v \neq u$ and $w \neq u$ such that every path from v to w contains u. If an undirected graph G is connected and has no cut-vertex, then G is *biconnected*. A *bridge* in an undirected graph is an edge $\{u, v\}$ such that every path

* Supported by the Deutsche Forschungsgemeinschaft (DFG), Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4.
** Partially supported by the Deutsche Forschungsgemeinschaft (DFG), research project OPAL (optimal solutions for hard problems in computational biology), NI 369/2.

F. Dehne, J.-R. Sack, and N. Zeh (Eds.): WADS 2007, LNCS 4619, pp. 484–495, 2007.

between u and v contains $\{u, v\}$. If G is connected and has no bridge, then G is *bridge-connected*. A directed graph $D = (V, A)$ is *strongly connected* if, for all pairs of vertices u and v, there is a path from u to v. The *connected (biconnected, bridge-connected, strongly connected) components* of a graph are its maximal connected (biconnected, bridge-connected, strongly connected) subgraphs. We consider here two connectivity augmentation problems, namely, Bridge-Connectivity Augmentation (BCA) and Biconnectivity Augmentation (BIA), where we are asked to add at most k links to make the given graph bridge-connected or biconnected.

There is a long history of research dealing with BCA and BIA starting in 1976 with the work of Eswaran and Tarjan [2]. They showed that, in the case that E' is *complete*, that is, graph (V, E') is a complete graph, both problems are polynomial-time solvable. In 1981, Frederickson and JáJá [4] proved the NP-completeness of both problems if E' is *incomplete*. Motivated by the NP-completeness, the approximability of the optimization versions of these problems has been extensively studied in the literature. Frederickson and JáJá [4] gave polynomial-time factor-2 approximation algorithms for both problems. For BCA, Nagamochi [8] improved the approximation factor to 1.875. Later, Even et al. [3] presented a factor-1.5 approximation algorithm for BCA. In case of BIA, Khuller and Thurimella [5] improved the running time of the factor-2 approximation algorithm in [4]. In an unpublished manuscript, Kortsarz and Nutov [7] claimed a polynomial-time factor-$\frac{12}{7}$ algorithm for BIA. On the negative site, Kortsarz et al. [6] proved that there exists an $\epsilon > 0$ for which it is NP-hard to approximate BCA and BIA within a factor of $1 + \epsilon$.

Concerning the parameterized complexity of these problems, we are only aware of one result due to Nagamochi [8]. Since the bridge-connected components of a graph form a tree, we may assume that the input graph of a BCA-instance is a tree by contracting these components. Nagamochi [8] showed that BCA is fixed-parameter tractable with respect to the number of leaves ℓ in the given tree. More precisely, there is an algorithm solving this problem in $O(\ell^{\ell+1} \log \ell \cdot (|V| + |E'|))$ time. Since the number of leaves provides a lower bound on the solution size k, BCA is fixed-parameter tractable with respect to k. Nothing has been known concerning the kernelization of these problems. Problem kernelization is one of the most important contributions of fixed-parameter algorithmics to practical computing [1,9]. A *kernelization* is a polynomial-time algorithm that transforms a given instance I with parameter k of a problem P into a new instance I' with parameter $k' \leq k$ of P such that the original instance I is a yes-instance with parameter k iff the new instance I' is a yes-instance with parameter k' and $|I'| \leq g(k)$ for a function g. The instance I' is called the *problem kernel*. We complement the result of Nagamochi with a problem kernel with $O(k^2)$ vertices and links for BCA and BIA.

Furthermore, we study *partial* connectivity augmentation problems, a natural generalization of the connectivity augmentation problems, where we have as input a graph $G = (V, E)$, a set E' of links, and two non-negative integers k, Φ

and ask for a subset E'' of E' with $|E''| \leq k$ such that graph $(V, E \cup E'')$ has a subgraph that contains at least Φ vertices and satisfies the given connectivity property. Clearly, if $\Phi = |V|$, then we have the connectivity augmentation problems. We consider two partial connectivity augmentation problems, Partial Bridge-Connectivity Augmentation and Partial Strong Connectivity Augmentation. For both connectivity properties, their corresponding non-partial connectivity augmentation problems are solvable in polynomial-time if E' is complete [2]. We show in Sect. 5 that Partial Bridge-Connectivity Augmentation with a complete link set remains polynomial-time solvable but Partial Strong Connectivity Augmentation with a complete link set is W[2]-hard, that is, it is very unlikely that this problem is fixed-parameter tractable with respect to the parameter k.

Most proofs are deferred to a long version of this paper.

2 Preliminaries

Parameterized complexity is a two-dimensional framework for studying the computational complexity of problems [1,9]. One dimension is the input size n (as in classical complexity theory) and the other one the *parameter* k (usually a positive integer). A problem is called *fixed-parameter tractable* (fpt) if it can be solved in $f(k) \cdot n^{O(1)}$ time, where f is a computable function only depending on k. A core tool in the development of fixed-parameter algorithms is polynomial-time preprocessing by *data reduction rules*, often yielding a *kernelization*. Herein, the goal is, given any problem instance I with parameter k, to transform it in polynomial time into a new instance I' with parameter k' such that the size of I' is bounded from above by some function only depending on k, $k' \leq k$, and (I, k) is a yes-instance iff (I', k') is a yes-instance. A data reduction rule is *correct* if the new instance after an application of this rule is a yes-instance iff the original instance is a yes-instance. Throughout this paper, we call a problem instance *reduced* if the corresponding data reduction rules cannot be applied anymore. A formal framework to show *fixed-parameter intractability* was developed by Downey and Fellows [1] who introduced the concept of *parameterized reductions*. A parameterized reduction from a parameterized language L to another parameterized language L' is a function that, given an instance (I, k), computes in $f(k) \cdot n^{O(1)}$ time an instance (I', k') (with k' only depending on k) such that $(I, k) \in L \Leftrightarrow (I', k') \in L'$. The basic complexity class for fixed-parameter intractability is W[1] as there is good reason to believe that W[1]-hard problems are not fixed-parameter tractable [1].

Throughout this paper, we set $n := |V|$ and $m := |E'|$ for a given graph $G = (V, E)$ and a given link set E'. For a graph G, we also use $V(G)$ and $E(G)$ to denote its vertex and edge set, respectively. The *neighborhood* $N(v)$ of a vertex $v \in V$ is the set of vertices that are adjacent to v. The *degree* of a vertex v, denoted by $\deg(v)$, is the size of $N(v)$.

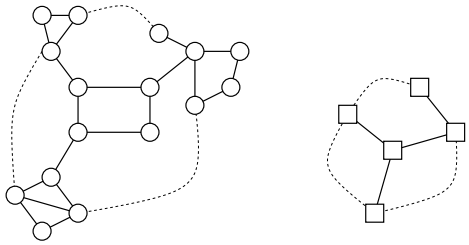

Fig. 1. Example of contracting bridge-connected components of a given graph. The links are drawn as dashed lines.

3 The Bridge-Connectivity Augmentation Problem

The main result of this section is a data reduction for Bridge-Connectivity Augmentation that leads to a quadratic-size problem kernel. Given an instance of BCA, we can assume that the input graph G is a tree [2,4,3]: Each bridge-connected component of $G = (V, E)$ can be contracted into a single vertex by contracting all edges in this component, resulting in a tree. The set of links has to be adapted accordingly. The contraction of the bridge-connected components can be done in $O(|V| + |E|)$ time [11]. See Fig. 1 for an example. Hence, in the following, the input instance is always denoted by T.

In contrast to the tree edges, denoted by $\{u, v\}$, we denote links by (u, v). We use $p_{u,v}$ to denote the uniquely determined path between two vertices u and v in T. In the course of the data reduction process, if a link $(u, v) \in E'$ is added to a solution E'', then the vertices from the path $p_{u,v}$ form a bridge-connected component and we contract all edges in this component, obtaining a tree again. We say a link (u, v) *covers* an edge e if e lies on $p_{u,v}$. For an edge $e \in E$, we use $l(e)$ to denote the set of links covering e. A link $(u, v) \in E'$ is called a *shadow* if there exists a link $(u', v') \in E'$ with $V(p_{u,v}) \subsetneq V(p_{u',v'})$. Let $N_{E'}(u) := \{v \mid (u, v) \in E'\}$. For a vertex $v \in V$ and an edge $e \in E$, we use $T_{v,e}$ to denote the subtree of $(V, E \setminus \{e\})$ that contains v.

The following observation provides the starting point for our kernelization:

Lemma 1 ([2]). *Let $L(T)$ be the set of leaves of the tree T of a BCA-instance. Every solution of this instance contains at least $|L(T)|/2$ many links, that is, $k \geq |L(T)|/2$.*

We can conclude that every yes-instance of BCA contains at most $2k$ leaves and, also, at most $2k - 1$ internal vertices with degree at least three. It remains to upper-bound the number of internal vertices of degree two. If we can bound the maximum length of the paths that consist solely of degree-2 vertices, then we can achieve an upper bound on the number of degree-2 vertices. To this end, we apply four data reduction rules. We begin with three data reduction rules that are also used in [3,8] and whose correctness is easy to verify.

Shadow Deletion: Delete all shadows in E'.
Unit Link: If there is an edge $e \in E$ with $l(e) = \{(u, v)\}$, then contract $p_{u,v}$ and decrease the parameter k by one.
Covered Edge: If $l(e_1) \subseteq l(e_2)$ for two edges $e_1, e_2 \in E$, then contract e_2.

Lemma 2. *The above three rules can be executed in $O(n \cdot m^3 + n^3 \cdot m)$ time.*

Before we present the fourth data reduction rule, we show some structural properties of a BCA-instance that is reduced with respect to the above three rules. In particular, we show that, in a reduced instance, the links over a path consisting solely of degree-2 vertices have some "consecutiveness" property, which provides the basis for the fourth data reduction rule.

Lemma 3. *Let $(T = (V, E), E', k)$ be a reduced instance with respect to the above three rules, let $v \in V$ be a degree-2 vertex in T, and let e, e' be the edges incident to v. Then, there exists at least one link (v, x) in E' with $x \in V(T_{v,e})$ and at least one link (v, y) in E' with $y \in V(T_{v,e'})$.*

Lemma 4. *Let $(T = (V, E), E', k)$ be a reduced instance with respect to the above three rules. Then, for every link $(u, v) \in E'$, it holds that $|E(p_{u,v})| \geq 2$.*

Lemma 5. *Let $(T = (V, E), E', k)$ be a reduced instance with respect to the above three rules. Consider a path $P = \{u, v_1\}, \{v_1, v_2\}, \ldots, \{v_l, w\}$ in T with $deg(v_i) = 2$ for all $1 \leq i \leq l$, $deg(u) \geq 3$, and $deg(w) \geq 3$. Let E'_v denote the set of links with both endpoints in $\{v_1, v_2, \ldots, v_l\}$. If $E'_v \neq \emptyset$, then there exists an integer N with $1 \leq N \leq l - 1$ such that $E'_v = \{(v_i, v_{i+N}) \mid 1 \leq i \leq l - N\}$ and there exists no link (x, y) with $x \in V(T_{u,\{u,v_1\}})$ and $y \in V(T_{w,\{v_l,w\}})$.*

The fourth data reduction rule restricts the length of paths that consist solely of degree-2 vertices. By Lemma 5, the links with both endpoints from such a path admit a "consecutiveness" property. By making use of this property, the next data reduction rule replaces a long degree-2 path by a shorter "equivalent" degree-2 path.

Degree-2-Path: Let $(T = (V, E), E', k)$ be a reduced instance with respect to the above three rules. Let $P = \{u, v_1\}, \{v_1, v_2\}, \ldots, \{v_l, w\}$ be a path in T such that $\deg(v_i) = 2$ for all $1 \leq i \leq l$, $\deg(u) \geq 3$, and $\deg(w) \geq 3$ and let E'_v denote the set of links with both endpoints from $v_1, v_2, \ldots, v_l$. If there exists an integer N with $l \geq 2N$ such that $E'_v = \{(v_i, v_{i+N}) \mid 1 \leq i \leq l - N\}$, then proceed as follows: Let $c := \lfloor \frac{l}{N} \rfloor - 1$ and $d := (l \mod N)$. Replace P by a path $P' = \{u, x_1\}, \{x_1, x_2\}, \ldots, \{x_{N+d}, w\}$. Remove all links in E'_v from E' and add the links (x_i, x_{i+N}) for $1 \leq i \leq d$ to E'. Replace every link (v_i, y) with $y \in V(T_{u,\{u,v_1\}})$ by the link (x_i, y) and replace every link (v_i, y) with $y \in V(T_{w,\{v_l,w\}})$ by the link $(x_{N+d-(l-i)}, y)$. Finally, decrease parameter k by c.

See Fig. 2 for an example of the application of the Degree-2-Path rule.

Lemma 6. *The Degree-2-Path rule is correct and can be executed in $O(n^2 + nm)$ time.*

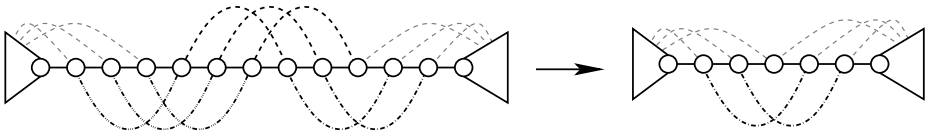

Fig. 2. Example for the Degree-2-Path rule for $N = 3$, $l = 11$, $c = 2$, and $d = 2$

Proof. Let $(T = (V, E), E', k)$ be a BCA-instance reduced with respect to the first three rules and let (T_a, E'_a, k_a) be the resulting instance after one application of the Degree-2-Path rule. Let $P = \{u, v_1\}, \{v_1, v_2\}, \ldots, \{v_l, w\}$ be the path for which the conditions of the Degree-2-Path rule are fulfilled and let E'_v be the set of links with both endpoints from $v_1, \ldots, v_l$. By Lemma 5, either $E'_v = \emptyset$ or there is an integer N with $E'_v = \{(v_i, v_{N+i}) \mid 1 \leq i \leq l - N\}$ and there is no link between a vertex in $V(T_{u,\{u,v_1\}})$ and a vertex in $V(T_{w,\{v_l,w\}})$. Since the Degree-2-Path rule is applicable to P, we know $E'_v \neq \emptyset$ and $l \geq 2N$. Then, we have $c = \lfloor \frac{l}{N} \rfloor - 1$, $d = (l \mod N)$, and $k_a = k - c$. We show that (T, E', k) is a yes-instance if and only if (T_a, E'_a, k_a) is a yes-instance. The correctness of the Degree-2-Path rule then follows by induction on the number of applications of the rule. Here, we give only the proof of "⇒"-direction. The proofs of "⇐"-direction and the running time are omitted due to lack of space.

"⇒": Let $E'' \subseteq E'$ with $|E''| \leq k$ be a solution for the original instance. We first show some properties of $E'' \cap E'_v$ and then construct a solution E''_a for the new instance with $|E''_a| \leq k_a$ from E''.

We can assume that $E'' \cap E'_v$ contains only pairwise "non-overlapping" links, that is, there are no two links $(v_i, v_{i+N}), (v_j, v_{j+N}) \in E'' \cap E'_v$ with $i < j < i+N$: If there are two links $(v_i, v_{i+N}), (v_j, v_{j+N}) \in E'' \cap E'_v$ with $i < j < i + N$, then we construct another solution from E'' by replacing (v_j, v_{j+N}) by (v_{i+N}, z), where $z = v_{i+2N}$ or z is a vertex from $T_{w,\{v_l,w\}}$. Link (v_{i+N}, z) exists due to Lemma 3. Obviously, this yields a solution of the same size (or even of smaller size if (v_{i+N}, z) is already part of the solution).

Next, we show that $c \leq |E'' \cap E'_v| \leq c + 1$. On the one hand, since every link in E'_v covers exactly N edges and the links in $E'' \cap E'_v$ are pairwise non-overlapping, there can be at most $\lfloor \frac{l-1}{N} \rfloor = \lfloor \frac{(c+1)N+d-1}{N} \rfloor = c + 1$ pairwise non-overlapping links in $E'' \cap E'_v$. On the other hand, all edges of path P which lie between vertex v_N and vertex v_{l-N+1} have to be covered by links in E'_v. This means that E'' has cardinality at least $\lceil \frac{l-2N+1}{N} \rceil = \lceil \frac{(c+1)N+d-2N+1}{N} \rceil = (c-1) + \lceil \frac{d+1}{N} \rceil = c$.

In the following, let $i_l := \max\{i \mid (y, v_i) \in E'' \wedge y \in V(T_{u,\{u,v_1\}})\}$ and $i_r := \min\{i \mid (v_i, z) \in E'' \wedge z \in V(T_{w,\{v_l,w\}})\}$. We distinguish two cases, namely, $|E'' \cap E'_v| = c$ and $|E'' \cap E'_v| = c + 1$, and construct in both cases a solution for the new instance.

In the first case, we can assume that $E'' \cap E'_v = \{(v_{i_l}, v_{i_l+N}), (v_{i_l+N}, v_{i_l+2N}), \ldots, (v_{i_l+(c-1)N}, v_{i_l+cN})\}$. We construct the new solution E''_a from E'' as follows: We replace every link $(y, v_i) \in E''$ by a link (y, x_i). Note that, since the given instance is reduced with respect to the first three rules, the existence

of the link (y, v_i) implies that $i < N \leq N + d$ and, thus, the link (y, x_i) exists in E'_a. The links $(v_i, z) \in E''$ with $z \in V(T_{w,\{v_l,w\}})$ are also replaced by links $(x_{N+d-(l-i)}, z)$. With the same reason as above, links (x_i, z) exist in E'_a. Finally, we remove all links in $E'' \cap E'_v$ from E''. The resulting set E''_a is then a solution for the new instance. Since $|E'' \cap E'_v| = c$ in this case, we have $|E''_a| \leq k_a$. Obviously, all edges of T_a that are not between the new vertices $x_1, \ldots, x_{N+d}$ are covered. To show that the edges between the new vertices are also covered, observe that the edges on the path between x_1 and x_{i_l} and the edges on the path between $x_{N+d-(l-i_r)}$ and x_{N+d} are covered by the links (y, x_{i_l}) with $y \in V(T_{u,\{u,v_1\}})$ and $(x_{N+d-(l-i_r)}, z)$ with $z \in V(T_{w,\{v_l,w\}})$ that replace the links (y, v_{i_l}) and (v_{i_r}, z), respectively. Then, it suffices to show that $N + d - (l - i_r) \leq i_l$. Since E'' is a solution of the non-reduced instance, we get $i_l + cN \geq i_r$. This is equivalent to $i_l \geq i_r - cN = N + d - cN - N - d + i_r = N + d - (l - i_r)$.

In the second case, we assume that $E'' \cap E'_v = \{(v_{i_l}, v_{i_l+N}), (v_{i_l+N}, v_{i_l+2N}), \ldots, (v_{i_l+cN}, v_{i_l+(c+1)N})\}$. We construct the solution E''_a for the new instance from E'' as follows: We replace every link $(y, v_i) \in E''$ with $y \in V(T_{u,\{u,v_1\}})$ by a link (y, x_i). The links $(v_i, z) \in E''$ with $z \in V(T_{w,\{v_l,w\}})$ are also replaced by links $(x_{N+d-(l-i)}, z)$. We remove all links in $E'' \cap E'_v$ from E''. Finally, we add link (x_{i_l}, x_{i_l+N}) to E''. With a similar argument, we can show that the resulting set is a solution of the new instance. □

Next, we show that a BCA-instance reduced with respect to the four data reduction rules has $O(k^2)$ vertices and $O(k^2)$ links. The key point in the following is to upper-bound the number of the internal vertices with degree two. Herein, we consider the paths formed by degree-two vertices. The next two lemmas are used to show the upper bounds on the number and the length of such paths. The first one is due to Even et al. [3] and shows that there is no such degree-two vertex path between a leaf and an internal vertex of degree at least three.

Lemma 7 ([3]). *Let (T, E', k) be a BCA-instance to which the Shadow Deletion rule, the Unit Link rule, and the Covered Edge rule cannot be applied. Let v be a leaf of T and u be the parent of v. Then, $deg(u) \geq 3$.*

In the next lemma, we upper-bound the length of a path in a reduced instance that consists of degree-two vertices. Herein, we use $L(T)$ to denote the set of leaves in tree T.

Lemma 8. *Let (T, E', k) be a reduced BCA-instance and let $P = \{u, v_1\}, \{v_1, v_2\}, \ldots, \{v_l, w\}$ be a path in T with $deg(u) \geq 3$, $deg(w) \geq 3$, and $deg(v_i) = 2$ for all $1 \leq i \leq l$. Let $L_u := L(T_{u,\{u,v_1\}})$ and $L_w := L(T_{w,\{v_l,w\}})$. Then,*

(1) there are at most $2|L_u| + 2|L_w|$ links that have exactly one endpoint in $V_P := \{v_1, v_2, \ldots, v_l\}$.
(2) $l \leq 4 \cdot \min(|L_u|, |L_w|)$.

Proof. In the following, we use T_u and T_w to denote $T_{u,\{u,v_1\}}$ and $T_{w,\{v_l,w\}}$, respectively, and consider them as two rooted trees with roots u and w, respectively. For a vertex x in T_u or T_w, we use T_x to denote the subtree of T_u or T_w

rooted at x. Moreover, we use A_u and A_w to denote the set of internal vertices of degree at least 3 in T_u and T_w.

The key for proving the lemma is the following observation: There exist at most $|A_u| + |L_u|$ links in E' with one endpoint in T_u and one endpoint in V_P and there exist at most $|A_w| + |L_w|$ links in E' with one endpoint in T_w and one endpoint in V_P. Here, we prove this observation for T_u. Consider a degree-2 vertex x in T_u. By Lemma 3, there exists a link $(x, y) \in E'$ with $y \in V(T_x)$. The existence of the link (x, y) then excludes any link (a, b) with $a \in V(T_y)$ and $b \in V_P$, since, otherwise, (x, y) would be a shadow and the Shadow Deletion rule would be applied. In this way, every degree-2 vertex x in T_u "blocks" at least one vertex from T_x from building links with the vertices in V_P. Thus, by a simple calculation, we arrive at the $|A_u| + |L_u|$-bound on the number of links between the vertices in T_u and the vertices in V_P.

From the above observation, we know that there are at most $|A_u| + |L_u| + |A_w| + |L_w|$ links with exactly one endpoint in V_P. Since $|A_u| \leq |L_u| - 1$ and $|A_w| \leq |L_w| - 1$, the first part of the lemma follows.

To prove the second part of the lemma, we distinguish two cases. First, suppose that there exists at least one link $(x, y) \in E'$ with $x \in V(T_u)$ and $y \in V(T_w)$. Then, since the instance is reduced with respect to the Shadow Deletion rule, there is no link in E' between two vertices in V_P. By Lemma 3, for every vertex v_i in V_P, there are at least two links $(v_i, a), (v_i, b) \in E'$ with $a \in V(T_u)$ and $b \in V(T_w)$. According to the above observation, there are at most $2|L_u| - 1$ (or $2|L_w| - 1$) links between V_P-vertices and the vertices in $V(T_u)$ (or $V(T_w)$). Therefore, $|V_P| \leq \min(2|L_u| - 1, 2|L_w| - 1)$.

In the second case, there is no link $(x, y) \in E'$ with $x \in V(T_u)$ and $y \in V(T_w)$. Let v_{i_l} be the vertex in V_P such that there exist a link $(v_{i_l}, z) \in E'$ with $z \in V(T_u)$ and, for all $i_l \leq i \leq l$, there is no link $(v_i, z) \in E'$ with $z \in V(T_u)$. From the above observation, we know $i_l \leq 2|L_u| - 1$. By Lemma 3 and the fact that the instance is reduced with respect to the Degree-2-Path rule, for every vertex v_i with $i_l \leq i \leq l$, there is a link $(v_i, v_j) \in E'$ with $1 \leq j \leq i_l$. Since the instance is reduced with respect to the Shadow Deletion rule, there cannot be two vertices from $\{v_{i_l+1}, \ldots, v_l\}$ which form two links with one vertex from $\{v_1, \ldots, v_{i_l}\}$, we can conclude $l \leq 2i_l \leq 4|L_u| - 2$. Obviously, the same argument can also be applied to obtain $l \leq 4|L_w| - 2$. The second part of the lemma follows. □

Now, we prove the size bound of the problem kernel for BCA.

Theorem 1. Bridge-Connectivity Augmentation *admits a problem kernel with $O(k^2)$ vertices and $O(k^2)$ links.*

4 The Biconnectivity Augmentation Problem

In this section, by studying Biconnectivity Augmentation (BIA), we deal with a more general problem setting than in the previous section. Hence, based on the previous section, we extend and refine our kernelization technique introduced there. As shown by Frederickson and JáJá [4], we can assume that the input

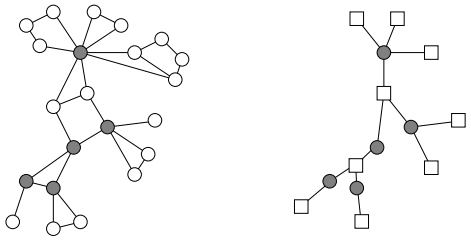

Fig. 3. A graph together with its block tree. The cut-vertices are colored gray and the block-vertices are drawn as rectangles.

graph is a so-called *block tree*. A block tree $T = (V_T, E_T)$ is a tree over the vertex set $V_T := B \cup C$ with $B \cap C = \emptyset$ where the leaves of T form a subset of B and the edges in E_T have one endpoint from B and one endpoint from C.

We can easily compute a block tree from a given undirected and connected graph $G = (V, E)$: Identify B as the set of biconnected components of G and C as the set of cut-vertices of G. Insert an edge between a biconnected component and a cut-vertex into E_T if the cut-vertex belongs to the biconnected component. In the following, the vertices in B are called *block-vertices* and the vertices in C are called *cut-vertices*. See Fig. 3 for an example of a graph and its block tree.

Eswaran and Tarjan [2] gave a lower bound on the size of the solutions of a BIA-instance.

Lemma 9 ([2]). *If (T, E', k) is a yes-instance of BIA, then $k \geq \lceil |L|/2 \rceil$ where L is the set of leaves of T.*

By Lemma 9, the number of leaves and the number of internal vertices of degree at least three of a given block tree can be easily bounded from above by $2k$ and $2k - 1$. Again, we focus on the internal vertices of degree two of T. The decisive difference between a BIA-instance and a BCA-instance lies in the partition of the tree vertices into two subsets, the block-vertices and the cut-vertices. A block-vertex can only have cut-vertices as neighbors and vice versa. In the following, we present first a preprocessing, which ensures that the links in E' are all between block-vertices.

Preprocessing: While there exists a link $(u, v) \in E'$ with $u \in C$ and $v \in B \cup C$, replace (u, v) by the link (w, v) where $w \in B$ is the neighbor of u that lies on the path between u and v in T. Finally, for all $u \in B \cup C$, remove all links (u, u) from E'.

To see the correctness of the preprocessing, the following equivalent formulation of BIA is helpful: Given a block tree $T = (B \cup C, E_T)$, a set of links E', and a non-negative integer k, find a subset E'' of links with $|E''| \leq k$ such that, for every $c \in C$, if c is removed from the graph $(B \cup C, E_T \cup E'')$, then the resulting graph is still connected.

Lemma 10. *The preprocessing is correct and can be executed in $O(n \cdot m)$ time.*

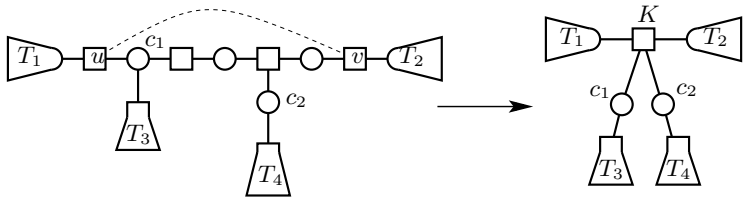

Fig. 4. An illustration of the modification made after adding a link (u, v) to the solution set

Next, we present the data reduction rules for BIA which generalize the data reduction rules in Sect. 3. Herein, if we add a link (u, v) to the solution E'', then, following Rosenthal and Goldner [10], we modify the instance as follows: Let P denote the path in T between u, v, let C be the set of cut-vertices on P that have degree at least three, and let N be the set of cut-vertices which are neighbors of the block-vertices in P and do not lie on P. Replace P by a single block-vertex K. Every link (u, v) with at least one endpoint being in P, say u, is replaced by link (K, v). For every vertex $v \in N$, add edge $\{K, v\}$ and, for every $c \in C$, add edge $\{K, c\}$. An illustration is given in Fig. 4.

The data reduction rules use the following terms and notations: For a vertex u, we use E'_u to denote the links in E' which cover u or have u as one of its endpoints. We call a path between two cut vertices a *degree-2-cut-path* if all vertices on this path are degree-two vertices. A degree-2-cut-path is *maximal* if it is not a proper subpath of another degree-2-cut-path.

Shadow Deletion: Delete all shadows.
Unit Link: If there exists a cut-vertex u with $E'_u = \{(x, y)\}$, then add (x, y) to E'' and decrease the parameter k by one.
Covered Cut-Vertex: If there are two cut-vertices u and v with $E'_u \subseteq E'_v$ and $N(v) = \{w_1, w_2\}$, then add a new block-vertex w and make it adjacent to the vertices in $(N(w_1) \cup N(w_2)) \setminus \{v\}$ and replace every link of the form (w_1, x) or (w_2, x) by a link (w, x). Finally, remove v, w_1, w_2 from T.
Degree-2-Cut-Path: Let (T, E', k) be a BIA-instance to which the first three rules do not apply, let $P = \{c_1, b_1\}, \{b_1, c_2\}, \{c_2, b_2\}, \ldots, \{c_l, b_l\}, \{b_l, c_{l+1}\}$ be a maximal degree-2-cut-path in T with $\{b_1, \ldots, b_l\} \subseteq B$ and $\{c_1, \ldots, c_{l+1}\} \subseteq C$, and E'_b be the set of links with both endpoints from $\{b_1, \ldots, b_l\}$. If there exists an integer N with $2N \leq l$ such that $E'_b = \{(b_i, b_{i+N}) \mid 1 \leq i \leq l - N\}$, then proceed as follows: Let $c := \lfloor \frac{l}{N} \rfloor - 1$ and $d = l \mod N$. Replace P by a path $P' = \{c'_1, b'_1\}, \ldots, \{c'_{N+d}, b'_{N+d}\}, \{b'_{N+d}, c'_{N+d+1}\}$. Remove all links in E'_b from E' and add links (b'_i, b'_{i+N}) for $1 \leq i \leq d$ to E'. Replace every link (b_i, y) with $y \in V(T_{b_1, \{b_1, c_1\}})$ by link (b'_i, y) and replace every link (b_i, y) with $y \in V(T_{b_l, \{c_{l-1}, b_l\}})$ by link $(b'_{N+d-(l-i)}, y)$. Finally, decrease the parameter k by c.

Due to the similarity of the four rules to the ones in Sect. 3, the running time follows from Lemmas 2 and 6.

Lemma 11. *The four data reduction rules are correct and can be executed in* $O(n \cdot m^3 + n^3 \cdot m)$ *time.*

Theorem 2. Biconnectivity Augmentation *admits a problem kernel with* $O(k^2)$ *vertices and* $O(k^2)$ *links.*

5 Partial Augmentation Problems

In this section, we study partial augmentation problems. Here, given a graph, a set of links, and two non-negative integers Φ, k, one asks for a set of at most k links whose insertion in the graph results in a graph that has a subgraph with at least Φ vertices that satisfies a given connectivity property. We show that, in the case that the link set is complete, that is, it contains all possible edges or arcs, Partial Bridge-Connectivity Augmentation (PBCA) is polynomial-time solvable and Partial Strong Connectivity Augmentation (PSCA) is W[2]-hard. Note that the non-partial versions of both problems are polynomial-time solvable if the link set is complete.

5.1 Partial Bridge-Connectivity Augmentation

The Partial Bridge-Connectivity Augmentation problem (PBCA) we study here is defined as follows: Given an undirected and connected graph $G = (V, E)$, a set of links E', and two non-negative integers Φ, k, find a set E'' of at most k links such that the graph $(V, E \cup E'')$ contains a bridge-connected component with at least Φ vertices. Note that in the case that E' is incomplete, Bridge-Connectivity Augmentation is NP-complete, which implies that PBCA is also NP-complete in this case. We show here that PBCA becomes polynomial-time solvable if E' is complete. This extends a result by Eswaran and Tarjan [2] saying that BCA is polynomial-time solvable in the case of a complete link set. Our solving strategy consists of two steps: The first step reduces the augmentation problem to a special subtree problem and the second step applies a dynamic programming approach to solve the subtree problem. The special subtree problem, Maximum d-Leaves Subtree (MLST), is defined as follows: Given a tree $T = (V_T, E_T)$, two non-negative integers N, d, and a weight function $w : V_T \to \mathbb{N}$, find a subtree of T with at most d leaves such that the total weight of the vertices in this subtree is at least N.

Theorem 3. Maximum d-Leaves Subtree *can be solved in* $O(|V_T| \cdot d^2)$ *time.*

Theorem 4. *In the case of a complete link set,* Partial Bridge-Connectivity Augmentation *is solvable in* $O(|V| \cdot k^2)$ *time.*

5.2 Partial Strong Connectivity Augmentation

Now, we show that Partial Strong Connectivity Augmentation (PSCA) is W[2]-hard, which is defined as follows: Given a directed graph $D = (V, A)$, a set of links $A' \subseteq V \times V$, and two non-negative integers Φ, k, find a subset A'' of A'

with $|A''| \leq k$ such that $(V, A \cup A'')$ contains a strongly connected component with at least Φ vertices. We give a parameterized reduction from the W[2]-hard Set Cover problem [1].

Theorem 5. *In both incomplete and complete link set cases,* Partial Strong Connectivity Augmentation *is W[2]-hard.*

6 Open Problems

The most interesting open problem is to study the parameterized complexity of the Strong Connectivity Augmentation problem, where we are given a directed graph $D = (V, A)$, a set of links $A \subseteq V \times V$, and a non-negative integer k and ask for a subset A'' of links such that graph $(V, A \cup A'')$ is strongly connected. We conjecture that this problem is fixed-parameter tractable with respect to k. Improving the size bounds of the problem kernels for Bridge-Connectivity Augmentation and Biconnectivity Augmentation to a linear function in k is another interesting open problem. Further opportunities for future work include investigating the approximability and fixed-parameter tractability of the Partial Bridge-Connectivity Augmentation and Partial Biconnectivity Augmentation problems for the case that the link set E' is incomplete.

References

1. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Heidelberg (1999)
2. Eswaran, K.P., Tarjan, R.E.: Augmentation problems. SIAM Journal on Computing 5(4), 653–665 (1976)
3. Even, G., Feldman, J., Kortsarz, G., Nutov, Z.: A 3/2-approximation algorithm for augmenting the edge-connectivity of a graph from 1 to 2 using a subset of a given edge set. In: Goemans, M.X., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) RANDOM 2001 and APPROX 2001. LNCS, vol. 2129, pp. 90–101. Springer, Heidelberg (2001)
4. Frederickson, G.N., JáJá, J.: Approximation algorithms for several graph augmentation problems. SIAM Journal on Computing 10(2), 270–283 (1981)
5. Khuller, S., Thurimella, R.: Approximation algorithms for graph augmentation. Journal of Algorithms 14(2), 214–225 (1993)
6. Kortsarz, G., Krauthgamer, R., Lee, J.R.: Hardness of approximation for vertex-connectivity network design problems. SIAM Journal on Computing 33(3), 704–720 (2004)
7. Kortsarz, G., Nutov, Z.: A 12/7-approximation algorithm for the vertex-connectivity of a graph from 1 to 2, Manuscipt (2002)
8. Nagamochi, H.: An approximation for finding a smallest 2-edge-connected subgraph containing a specified spanning tree. Discrete Applied Mathematics 126, 83–113 (2003)
9. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford University Press, Oxford (2006)
10. Rosenthal, A., Goldner, A.: Smallest augmentations to biconnect a graph. SIAM Journal on Computing 6(1), 55–66 (1977)
11. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM Journal on Computing 1(2), 146–160 (1972)