Noname manuscript No.
(will be inserted by the editor)

The $(2, k)$-connectivity augmentation problem: Algorithmic aspects

Florian Hörsch* · Zoltán Szigeti

the date of receipt and acceptance should be inserted later

Abstract Durand de Gevigney and Szigeti [9] have recently given a min-max theorem for the $(2, k)$-connectivity augmentation problem. This article provides an $O(n^3(m + n \log n))$ time algorithm to find an optimal solution for this problem.

Keywords Connectivity augmentation

1 Introduction

Let $\boldsymbol{G = (V, E)}$ be an undirected graph with $\boldsymbol{n}$ vertices and $\boldsymbol{m}$ edges and let $\boldsymbol{c}$: $E \to \mathbb{Z}_{>0}$ be an integer edge capacity function. For $X \subseteq V$, we denote the set of edges with exactly one endvertex in X by $\boldsymbol{\delta_G(X)}$. We say that G is *k-edge-connected* if $\sum_{e \in \delta_G(X)} c(e) \geq k$ for all nonempty, proper $X \subset V$.

For all problems treated in this article, the version with capacities and the version without capacities can be easily reduced to each other by replacing an edge with a capacity by multiple edges and vice-versa. Yet, this does not mean algorithmic equivalence. All the running times we give hold for the case with edge capacities, assuming that all basic operations can be executed in constant time. For this reason, all technical statements starting from Section 2 will be given in the capacitated form. During the introduction, however, we

Florian Hörsch
46 Avenue Félix Viallet
Grenoble, France
E-mail: florian.hoersch@grenoble-inp.fr
Corresponding author

Zoltán Szigeti
46 Avenue Félix Viallet
Grenoble, France
E-mail: zoltan.szigeti@grenoble-inp.fr

first describe the problems in the noncapacitated form for the sake of simplicity.

The theory of graph connectivity augmentation has seen significant progress during the last decades. The basic problem, the *global edge-connectivity augmentation* of undirected graphs, can be defined as follows: Given an undirected graph G and a positive integer k, find a set of edges of minimum cardinality whose addition results in a k-edge-connected graph. The problem was solved in terms of a min-max theorem by Cai and Sun [8] and a polynomial time algorithm was provided by Watanabe and Nakamura [18]. If we replace the minimum cardinality condition in the above problem by a minimum cost condition with respect to a given arbitrary cost function on V^2, the problem becomes NP-complete. For this reason, we only consider the minimum cardinality version for all augmentation problems treated in this article.

The method that is nowadays most commonly used when dealing with connectivity augmentation problems was introduced by Frank [11]. This method consists of two steps. In the first one a new vertex as well as edges connecting it to the given graph are added to have the required connectivity condition. Such a graph with the minimum even number of new edges is called a *minimal even extension.* The second step applies an operation called splitting off. A splitting off is the deletion of two edges incident to the new vertex and adding an edge between the other endvertices of these edges. A minimum augmentation of G can be obtained by repeatedly applying splitting offs maintaining the connectivity requirements and finally deleting the added vertex.

Besides the basic case, this method allows to handle several other versions of the problem such as the local edge-connectivity augmentation problem in undirected graphs and the global arc-connectivity augmentation problem for directed graphs. Frank [11] managed to provide both min-max theorems and efficient algorithms for these problems. Several further applications have been found, see for example [1], [2], [3], [4], [5], [6], [7] and [17].

Vertex-connectivity augmentation problems are more complicated than edge-connectivity augmentation problems. For the global vertex-connectivity augmentation problem in undirected graphs, no min-max theorem is known, however, a polynomial time algorithm for constant k was given by Jackson and Jordán [14]. For the global vertex-connectivity augmentation problem in directed graphs, a min-max theorem and an efficient algorithm for constant k were given by Frank and Jordán [13]. The main contribution of [13], besides the solution of the directed global vertex-connectivity augmentation problem, is the first application of bisets, a technique we also rely on in the present article.

Returning to the basic case of global undirected edge-connectivity augmentation, Frank [11] provided an $O(n^5)$ time algorithm based on the approach described above. In order to achieve this running time, he uses a slightly involved method for the splitting off part. Carefully choosing the pairs of edges to be split off, he manages to finish with a complete splitting off after a linear number of splitting off operations in n while the obvious approach results in a quadratic number. Nagamochi and Ibaraki [16] provided a more efficient implementation of the method described above. They managed to find a minimum k-edge-connected augmentation in $O((n \log n)(m + n \log n))$ time. An important ingredient in their work is an $O(n(m + n \log n))$ time mincut algorithm. We also make use of this mincut algorithm as a subroutine.

The problem we deal with in this article is $(2,k)$-connectivity augmentation. The concept of $(2,k)$-connectivity is a mixed form of edge-connectivity and vertex-connectivity and was first introduced in a more general form by Kaneko and Ota [15]. A graph $G = (V,E)$ is said to be $(2,k)$-connected if $|V| \geq 3$, G is $2k$-edge-connected and $G - v$ is k-edge-connected for every vertex $v \in V$. Bisets provide a convenient framework to treat this mixed-connectivity concept. The $(2,k)$-connectivity augmentation problem was considered by Durand de Gevigney and Szigeti [9], where a min-max theorem was proved. The authors characterized graphs admitting a complete splitting off maintaining $(2,k)$-connectivity in terms of an obstacle. They also worked out the minimal even extension step for $(2,k)$-connectivity. They showed that a minimal even extension for $(2,k)$-connectivity exists so that the obtained graph contains no obstacle yielding a complete admissible splitting off. This allowed the application of Frank's method to derive a min-max theorem. While a careful analysis of their proof yields a rather slow polynomial time algorithm for the noncapacitated version, no explicit mention of an algorithm is made.

The aim of the present article is to give an explicit algorithm of significantly improved running time that works for the more general capacitated version. Given a graph $G = (V,E)$ and $k \geq 2$, a *minimum $(2,k)$-connected augmentation* of G is a $(2,k)$-connected graph $G' = (V,E')$ such that $E \subseteq E'$ and $|E'|$ is minimum. The main result of this article is the following:

Theorem 1 *Given a graph G and an integer $k \geq 2$, we can compute a minimum $(2,k)$-connected augmentation of G in $O(n^3(m + n \log n))$ time.*

During the whole article we suppose that $k \geq 2$. For $k = 1$, observe that $(2,1)$-connectivity is equivalent to 2-vertex-connectivity and an $O(n+m)$ time algorithm for 2-vertex-connectivity augmentation has been found in [10] by Eswaran and Tarjan. We follow the proof technique of [9] and show how to turn it into an efficient algorithm with some more effort. An efficient algorithm for finding a minimal even extension for $(2,k)$-connectivity comes straight from its definition. Therefore, the main difficulty is to find the complete admissible splitting off yielding a $(2,k)$-connected augmentation of G whose existence is guaranteed by [9]. In comparison to the problems in [11], we seem to face another difficulty; graphs containing an obstacle need to be avoided during the splitting off process. We show that no obstacle can ever be created when splitting offs are executed from a minimal even extension for $(2,k)$-connectivity, so the aforementioned difficulty does not actually exist. We first use the mincut algorithm of [16] to compute the best splitting off for a given pair of edges incident to the new vertex in $O(n^2(m + n \log n))$ time. This yields a rather simple algorithm for the $(2,k)$-connectivity augmentation problem that runs in $O(n^4(m + n \ \log \ n))$ time. In the last part, we provide a more efficient algorithm for the splitting off part. In order to speed up the previous algorithm, we choose the splitting offs in a way that linearizes the number of required splitting offs. The overall running time of our algorithm for $(2, k)$-connectivity augmentation is $O(n^3(m + n \ \log \ n))$.

This algorithm is inspired by the method used by Frank in [11] for the case of k-edge-connectivity. However, there is a siginificant difference between our method and the method used by Frank. While Frank tries to execute several splitting offs all of which contain one common edge, we try to split off all pairs of edges included in a well-chosen triple of edges. This allows to avoid a significant amount of technicalities in comparison to an earlier version of this article.

This article is organized as follows: In Section 2, we give an overview of the results we use and show how to turn them into subroutines for our algorithm. In Section 3, we show how to efficiently compute a minimal even extension for $(2, k)$-connectivity and give some of its important properties. We show how to turn this into a simple algorithm that runs in $O(n^4(m + n \ \log \ n))$ time. In Section 3, we present a more involved algorithm that runs in $O(n^3(m + n \ \log \ n))$ time.

2 Basic definitions and previous results

2.1 Capacitated graphs

All the graphs considered are undirected and loopless. On the other hand, our graphs have nonnegative integer edge capacities. Let $(G = (V, E), c)$ be a capacitated graph, i.e. a graph with an integer edge capacity function $c : E \to \mathbb{Z}_{>0}$. Given a set $F \subseteq E$ of edges, we will use $\boldsymbol{c(F)}$ for $\sum_{e \in F} c(e)$. Observe that we require all our capacitated graphs to have integer edge capacities, so we shall not create any capacitated graphs with noninteger edge capacities during our algorithm. We also do not consider capacitated graphs with multiple edges as they can easily be replaced by a single edge with the sum of the corresponding capacities. For our algorithms, we will suppose that additions and subtractions of real numbers can be executed in constant time.

For $X, Y \subseteq V$, we use $\boldsymbol{\delta_G(X, Y)}$ to denote the set of edges between $X - Y$ and $Y - X$. We use $\boldsymbol{\overline{\delta}_G(X, Y)}$ for $\delta_G(X \cap Y, V - (X \cup Y))$. For a vertex $v \in V$, we use $\boldsymbol{\delta_G(v)}$ for $\delta_G(\{v\})$ and $\boldsymbol{N_G(v)}$ for the set of neighbors of v.

We require the following well-known result that can be found in Proposition 1.2.1 in [12].

Proposition 1 *Let $(G = (V, E), c)$ be a capacitated graph. For all $X, Y \subseteq V$, the following hold:*

(a) $c(\delta_G(X)) + c(\delta_G(Y)) = c(\delta_G(X \cap Y)) + c(\delta_G(X \cup Y)) + 2c(\delta_G(X, Y))$,
(b) $c(\delta_G(X)) + c(\delta_G(Y)) = c(\delta_G(X - Y)) + c(\delta_G(X - Y)) + 2c(\overline{\delta}_G(X, Y)$.

2.2 Minimum cuts

Given a capacitated graph $(G=(V,E),c)$, the mincut problem consists of finding a set $\emptyset \neq S \subsetneq V$ that minimizes $c(\delta_G(S))$. This problem has been widely studied. Due to the specific nature of our application, we are interested in a slight variation of this problem: given a capacitated graph $(H=(V+s,E),c)$ with a distinguished vertex $s \notin V$, we want to find a set $\emptyset \neq S \subsetneq V$ that minimizes $c(\delta_H(S))$. In other words, we additionally require that no side of the cut consists of s only. We denote the capacity of such a minimum cut by $\boldsymbol{\lambda_{(H,c)}(V)}$. We say that $(H=(V+s,E),c)$ is *k-edge-connected in V* for some positive integer k if $\lambda_{(H,c)}(V) \geq k$. We strongly rely on the following algorithmic result which is due to Nagamochi and Ibaraki [16].

Lemma 1 *Given a capacitated graph $(H=(V+s,E),c)$, we can compute $\lambda_{(H,c)}(V)$ and a set $\emptyset \neq S \subsetneq V$ that minimizes $c(\delta_H(S))$ in $O(n(m+n\ \log\ n))$ time.*

2.3 Bisets

Given a ground set Ω, a *biset* X consists of two sets $\boldsymbol{X_O, X_I} \subseteq \Omega$ with $X_I \subseteq X_O$. We call X_I the *inner set* of X, X_O the *outer set* of X and $\boldsymbol{w(\mathsf{X})}= X_O - X_I$ the *wall* of X. We also say $\mathsf{X}=(X_O,X_I)$. We define the *complement* $\overline{\mathsf{X}}$ of X by $\overline{X}_O = \Omega - X_I$ and $\overline{X}_I = \Omega - X_O$. Observe that $w(\overline{\mathsf{X}}) = w(\mathsf{X})$. We say that X of Ω is *trivial* if $X_I = \emptyset$ or $X_O = \Omega$, *nontrivial* otherwise. Given two bisets X and Y, we define $\mathsf{X} \cup \mathsf{Y}= (X_O \cup Y_O, X_I \cup Y_I)$, $\mathsf{X} \cap \mathsf{Y}= (X_O \cap Y_O, X_I \cap Y_I)$ and $\mathsf{X} - \mathsf{Y}= \mathsf{X} \cap \overline{\mathsf{Y}}$.

Given a capacitated graph $(G=(V,E),c)$ and a positive integer k, we define a function f on the bisets on V by $\boldsymbol{f(\mathsf{X})}= k|w(\mathsf{X})| + c(\delta_G(X_I, V - X_O))$. Observe that $f(\mathsf{X}) = f(\overline{\mathsf{X}})$. This function will play a crucial role throughout the article.

We can now rephrase the definition of $(2,k)$-connectivity in terms of bisets. A capacitated graph $(G=(V,E),c)$ with $|V| \geq 3$ is $(2,k)$-connected if for every nontrivial biset X on V, we have $f(\mathsf{X}) \geq 2k$. We also need the following slightly more advanced notion: A capacitated graph $(H=(V+s,E),c)$ with $|V| \geq 3$ is called *$(2,k)$-connected in V* if for every biset X on V which is nontrivial with respect to V, we have $f(\mathsf{X}) \geq 2k$. Observe that in $H=(V+s,E)$ the vertex s belongs to $\overline{X}$ for any $X \subseteq V$.

2.4 Splitting off

Let $(H=(V+s,E),c)$ be a capacitated graph. For $v \in N_H(s)$ and a nonnegative integer $\alpha \leq c(sv)$, we denote by $\boldsymbol{(H,c)^{\alpha}_{v}}$ the capacitated graph obtained from (H,c) by decreasing the capacity of sv by α. If $c(sv)=0$ after the operation, we delete sv from H. For (H,c) that is $(2,k)$-connected in V, we denote $\boldsymbol{U_{(H,c)}}= \{v \in V |\ (H,c)^1_v \text{ is } (2,k)\text{-connected in } V\}$, a set that will play a significant role later on. For a vertex $v \in V$, we denote by $\boldsymbol{(H,c)_v^{max}}$ the capacitated graph $(H,c)_v^\alpha$ where α is the maximum integer such that $(H,c)_v^\alpha$ is well-defined and $(2,k)$-connected in V.

For $u, v \in N_H(s)$ and a positive integer $\alpha \leq c(su), c(sv)$, we denote by $\boldsymbol{(H,c)_{u,v}^\alpha}$ the capacitated graph obtained from (H,c) by decreasing $c(su)$ and $c(sv)$ by α and increasing $c(uv)$ by α. We delete edges of capacity 0 and create the edge uv if it does not exist yet. We also delete the arising loop if $u = v$. We call this operation the *α-multiple splitting off* of su and sv and say that this α-multiple splitting off contains su and sv. We abbreviate 1-multiple splitting off to *splitting off*. Suppose that (H,c) is $(2,k)$-connected in V. We say that a pair (su, sv) is *admissible* if $(H,c)_{u,v}^1$ is $(2,k)$-connected in V. For $u, v \in N_H(s)$, let α be the maximum integer such that $(H,c)_{u,v}^\alpha$ is well-defined and $(2,k)$-connected in V. We call an α-mutiple splitting off of (su, sv) a *maximal splitting off* of (su, sv) and denote $\boldsymbol{(H,c)_{u,v}^{max}} = (H,c)_{u,v}^\alpha$. Observe that every maximal splitting off can be viewed as a series of splitting offs.

We next give an important characterization of admissible pairs. Given a pair (su, sv), a nontrivial biset X with either $f(\mathsf{X}) \leq 2k+1$ and $u, v \in X_I$ or $f(\mathsf{X}) = 2k, u, v \in X_O$ and $\{u, v\} \cap X_I \neq \emptyset$ is said to *block* (su, sv). The following result can be found as Lemma 3.1 in [9] and will be frequently used.

Lemma 2 *Given a capacitated graph $(H = (V+s, E), c)$ that is $(2,k)$-connected in V and $u, v \in N_H(s)$, (su, sv) is admissible if and only if there is no biset blocking it.*

A biset that blocks a pair of edges (su, sv) with $u \neq v$ is called *horrifying*. Note that the wall of a horrifying biset contains at most one vertex.

While the following result is not explicitly proven in [9], its proof is almost literally the same as the one of Lemma 3.4 in [9]. We therefore omit it. The result nevertheless plays a key role in our algorithm.

Lemma 3 *Let $(H = (V+s, E), c)$ be a capacitated graph that is $(2,k)$-connected in V with $c(\delta_H(s))$ even. Let X be a horrifying biset, $u \in X_I \cap N_H(s)$ and $v \in N_H(s) - X_I$. If the biset Y blocks (su, sv), then either $\mathsf{X} \cup \mathsf{Y}$ is horrifying or X and Y have the same wall of size 1.*

We also require the following result that can be found in [9] as Proposition 3.2.

Lemma 4 *Let $(H = (V+s, E), c)$ be a capacitated graph that is $(2,k)$-connected in V and let X be a horrifying biset. Then $N_H(s) - X_O \neq \emptyset$.*

A series of splitting offs at s that results in a capacitated graph in which s is an isolated vertex is called a *complete splitting off* of (H,c). It is easy to see that a complete splitting off exists if and only if $c(\delta_H(s))$ is even. A complete splitting off is *admissible* if each of the splitting offs it contains is admissible in the current capacitated graph when being chosen. This is equivalent to the finally obtained capacitated graph being $(2,k)$-connected after deleting s. Finding such a complete admissible splitting off is the main difficulty in our algorithm. We strongly rely on a characterization of capacitated graphs having a complete admissible splitting off that can be found in [9]. Before stating it, we need the following definition:

Let $(H=(V+s,E),c)$ be a capacitated graph that is $(2,k)$-connected in V and with $c(\delta_H(s))$ even. An *obstacle* is a collection $\mathcal{B}$ of bisets in V and a vertex $t \in N_H(s)$ satisfying the following:

$$c(st) \text{ is odd and } t \in U_{(H,c)}, \tag{1}$$

$$w(\mathsf{B}) = \{t\} \text{ and } f(\mathsf{B}) = 2k \text{ for all } \mathsf{B} \in \mathcal{B}, \tag{2}$$

$$B_I \cap B'_I = \emptyset \text{ for all distinct } \mathsf{B}, \mathsf{B}' \in \mathcal{B}, \tag{3}$$

$$N_H(s) - \{t\} \subseteq \bigcup_{\mathsf{B} \in \mathcal{B}} B_I. \tag{4}$$

We say that t is the *special* vertex of the obstacle. It is easy to see that a capacitated graph containing an obstacle does not have a complete admissible splitting off, as every splitting off of (st, su) for some $u \in N_H(s)$ is blocked by the biset $\mathsf{B} \in \mathcal{B}$ with $u \in B_I$ and by Lemma 2. In [9], it is proved that the converse is also true:

Theorem 2 *Let $(H=(V+s,E),c)$ be a capacitated graph that is $(2,k)$-connected in V for some $k \geq 2$ with $c(\delta_H(s))$ even. Then (H,c) has a complete admissible splitting off at s if and only if (H,c) does not contain an obstacle.*

2.5 Basic algorithms

In this section, we show that we can efficiently compute the maximum decrease of the capacity of an edge and the maximum multiplicity of a splitting off of an edge pair that maintain certain connectivity requirements. We first show this for the case of edge-connectivity and then apply this for the case of $(2,k)$-connectivity. All of these results are simple consequences of Lemma 1.

Lemma 5 *Given a capacitated graph $(H=(V+s,E),c)$ that is k-edge-connected in V and a vertex $v \in N_H(s)$, we can compute the maximal α such that $(H',c') = (H,c)^{\alpha}_{v}$ is k-edge-connected in V in $O(n(m+n \log n))$ time. Further, if $c'(sv) \neq 0$, we can compute a set S with $v \in S \subsetneq V$ and $c'(\delta_{H'}(S)) = k$ in $O(n(m+n \log n))$ time.*

Proof Let $\gamma := c(sv)$ and $(H_\gamma, c_\gamma) = (H,c)^{\gamma}_{v}$. Obviously $\alpha \leq \gamma$. The other condition α needs to satisfy is that $c'(\delta_{H'}(X)) \geq k$ for every $\emptyset \neq X \subsetneq V$. If $v \notin X$, we obtain $c'(\delta_{H'}(X)) = c(\delta_H(X)) \geq k$. If $v \in X$, the condition is satisfied if and only if $0 \geq k - c'(\delta_{H'}(X)) = k - (c_\gamma(\delta_{H_\gamma}(X)) - \alpha + \gamma) = \alpha - (\gamma - k + c_\gamma(\delta_{H_\gamma}(X)))$. It follows that $\alpha = \min\{\gamma, \gamma - k + \lambda_{(H_\gamma, c_\gamma)}(V)\}$. By Lemma 1, we can compute $\lambda_{(H_\gamma,c_\gamma)}(V)$, and hence α, and also a set $S \subsetneq V$ with $c_\gamma(\delta_{H_\gamma}(S)) = \lambda_{(H_\gamma,c_\gamma)}(V)$ in $O(n(m+n \log n))$ time. If $c'(sv) \neq 0$, we have $c'(\delta_{H'}(S)) = k$. □

Lemma 6 *Given a capacitated graph $(H = (V + s, E), c)$ that is k-edge-connected in V and vertices $u, v \in N_H(s)$, we can compute the maximal α such that $(H', c') = (H, c)^{\alpha}_{u,v}$ is k-edge-connected in V in $O(n(m + n \log n))$ time. Further, if $c'(su), c'(sv) \neq 0$, we can compute a set S with $u, v \in S \subsetneq V$ and $c'(\delta_{H'}(S)) \leq k + 1$ in $O(n(m + n \log n))$ time.*

Proof Let $\gamma = \min\{c(su), c(sv)\}$ and $(H_\gamma, c_\gamma) = (H, c)^{\gamma}_{u,v}$. Obviously $\alpha \leq \gamma$. The other condition α needs to satisfy is that $c'(\delta_{H'}(X)) \geq k$ for every $\emptyset \neq X \subsetneq V$. If $\{u, v\} - X \neq \emptyset$, we obtain $c'(\delta_{H'}(X)) = c(\delta_H(X)) \geq k$. If $u, v \in X$, the condition is satisfied if and only if $0 \geq k - c'(\delta_{H'}(X)) = k - (c_\gamma(\delta_{H_\gamma}(X)) - 2\alpha + 2\gamma) = 2(\alpha - (\gamma + \frac{1}{2}c_\gamma(\delta_{H_\gamma}(X)) - \frac{1}{2}k))$. It follows that $\alpha = \min\{\gamma, \lfloor \gamma + \frac{1}{2}\lambda_{(H_\gamma, c_\gamma)}(V) - \frac{1}{2}k \rfloor\}$. By Lemma 1, we can compute $\lambda_{(H_\gamma, c_\gamma)}(V)$, and hence α, and also a set $S \subsetneq V$ with $c_\gamma(\delta_{H_\gamma}(S)) = \lambda_{(H_\gamma, c_\gamma)}(V)$, in $O(n(m + n \log n))$ time. If $c'(su), c'(sv) \neq 0$, we have $c'(\delta_{H'}(S)) \leq k + 1$. $\square$

Lemma 7 *Given a capacitated graph $(H = (V+s, E), c)$ that is $(2, k)$-connected in V and a vertex $v \in N_H(s)$, we can compute $(H, c)^{max}_v$ in $O(n^2(m + n \log n))$ time.*

Proof Let α be the maximum integer such that $(H, c)^{\alpha}_v$ is $2k$-edge-connected in V and $(H, c)^{\alpha}_v - x$ is k-edge-connected in $V - x$ for all $x \in V$. We first compute the maximum integer α' such that $(H, c)^{\alpha'}_v$ is $2k$-edge-connected in V. Using Lemma 5, this can be done in $O(n(m+n \log n))$ time. Next observe that for any nonnegative integer β, $(H, c)^{\beta}_v - v = (H, c) - v$ is always k-edge-connected in $V - v$ by assumption. Now consider $x \in V - v$ and observe that for any nonnegative integer β, we have $(H, c)^{\beta}_v - x = (H - x, c)^{\beta}_v$. It follows from Lemma 5 that we can compute the maximum integer α_x such that $(H, c)^{\alpha_x}_v - x$ is k-edge-connected in $V - x$ in $O(n(m+n \log n))$ time. We now can compute $\alpha = \min\{\alpha', \min_{x \in V - v} \alpha_x\}$. The overall running time is $O(n^2(m + n \log n))$. $\square$

Lemma 8 *Given a capacitated graph $(H = (V+s, E), c)$ that is $(2, k)$-connected in V and vertices $u, v \in N_H(s)$, we can compute $(H', c') = (H, c)^{max}_{u,v}$ in $O(n^2(m + n \log n))$ time. Further, if $c'(su), c'(sv) \neq 0$, we can compute a biset blocking (su, sv) in (H', c') in $O(n^2(m + n \log n))$ time.*

Proof By definition, $(H', c') = (H, c)^{\alpha}_{u,v}$ where α is the maximum integer such that $(H, c)^{\alpha}_{u,v}$ is $2k$-edge-connected in V and $(H, c)^{\alpha}_{u,v} - x$ is k-edge-connected in $V - x$ for all $x \in V$. We first compute the maximum integer α' such that $(H, c)^{\alpha'}_{u,v}$ is $2k$-edge-connected in V. Using Lemma 6, this can be done in $O(n(m + n \log n))$ time. For $x \in V - \{u, v\}$, we can compute, by Lemma 6, the maximum integer α_x such that $(H, c)^{\alpha_x}_{u,v} - x = (H - x, c)^{\alpha_x}_{u,v}$ is k-edge-connected in $V - x$ in $O(n(m + n \log n))$ time. We can compute, by Lemma 5, the maximum integer α_u such that $(H, c)^{\alpha_u}_{u,v} - u = (H - u, c)^{\alpha_u}_v$ is k-edge-connected in $V - u$ in $O(n(m + n \log n))$ time. We similarly compute α_v. We now can compute $\alpha = \min\{\alpha', \min_{x \in V} \alpha_x\}$. The overall running time is $O(n^2(m + n \log n))$.

Now suppose that $c'(su), c'(sv) \neq 0$. If $\alpha = \alpha'$, that is $(H', c') = (H, c)^{\alpha'}_{u,v}$, then, by Lemma 6, a set $S \subsetneq V$ with $u, v \in S$ and $c'(\delta_{H'}(S)) \leq 2k+1$ can be found in $O(n(m + n \ \log \ n))$ time. We obtain that (S, S) is a biset blocking (su, sv) in (H', c'). If $\alpha = \alpha_x$ for some $x \in V - \{u, v\}$, that is $(H', c') - x = (H - x, c)^{\alpha_x}_{u,v}$, then, by Lemma 6, a set $S \subsetneq V - x$ with $u, v \in S$ and $c'(\delta_{H'-x}(S)) \leq k+1$ can be found in $O(n(m+n \ \log \ n))$ time. We obtain that $(S \cup \{x\}, S)$ is a biset blocking (su, sv) in (H', c'). Finally, if $\alpha = \alpha_u$ or α_v, say α_u, that is $(H', c') - u = (H - u, c)^{\alpha_u}_{v}$, then, by Lemma 5, a set $S \subsetneq V - u$ with $v \in S$ and $c'(\delta_{H'}(S)) = k$ can be found in $O(n(m + n \ \log \ n))$ time. We obtain that $(S \cup \{u\}, S)$ is a biset blocking (su, sv) in (H', c'). □

3 Minimal even extensions for $(2, k)$-connectivity

A minimal even extension for $(2,k)$-connectivity of a capacitated graph $(G = (V, E_0), c_0)$ is obtained by adding a new vertex and edges incident to this vertex so that the obtained capacitated graph becomes $(2,k)$-connected in V and so that the total capacity of the new edges is even and minimal. The importance of minimal even extensions for $(2,k)$-connectivity is due to a theorem from [9] that shows that minimum augmentations for $(2,k)$-connectivity can be computed from minimal even extensions for $(2,k)$-connectivity by a complete admissible splitting off. We first give a simple algorithm to compute minimal even extensions for $(2,k)$-connectivity and give some basic properties. We further show a property of minimal even extensions for $(2,k)$-connectivity which is essential to our splitting off algorithms. This then allows to give a naive algorithm for the $(2,k)$-connectivity augmentation problem which is slower than the algorithm which is the main result of this article and is given later.

We first introduce the algorithm for computing a minimal even extension for $(2,k)$-connectivity. In order to avoid a technical definition whose details are not essential to this article, Algorithm 1 will also serve as a definition for minimal even extensions for $(2,k)$-connectivity. Algorithm 1 takes a capacitated graph $(G = (V, E_0), c_0)$ as input and adds a new vertex s as well as edges of sufficiently high capacity between s and all other vertices to make the capacitated graph $(2,k)$-connected in V. It then reduces these capacities in a greedy way as much as possible while maintaining $(2,k)$-connectivity in V. Finally, if the degree of s is odd, it augments the capacity of a certain chosen edge by 1.

Given a capacitated graph (G, c_0), a capacitated graph (H, c) which is obtained by applying Algorithm 1 to (G, c_0) is called a *minimal even extension for $(2,k)$-connectivity* of (G, c_0).

Proposition 2 *Given a capacitated graph (G, c_0), a minimal even extension for $(2,k)$-connectivity of (G, c_0) can be computed in $O(n^3(m+n \ \log \ n))$ time.*

Proof By definition, a minimal even extension for $(2,k)$-connectivity can be computed by Algorithm 1. It follows from Lemma 7 that line 4 can be executed

Algorithm 1 Minimal even extensions for $(2, k)$-connectivity

```
Input: A capacitated graph $(G = (V, E_0), c_0)$, an integer $k \geq 2$.
Output: A minimal even extension for $(2, k)$-connectivity of $(G, c_0)$.
1 Create $(H, c)$ by adding a vertex $s$ to $V$ and adding an edge of capacity $2k$ between
    $s$ and every $v \in V$;
2 Let $(v_1, \ldots, v_n)$ be an arbitrary ordering of the vertices of $V$;
3 for $i = 1, \ldots, n$ do
4     $(H, c) = (H, c)^{max}_{v_i}$;
5 if $c(\delta_H(s))$ is odd then
6     choose the maximum $i^*$ such that $c(sv_{i^*})$ is odd;
7     $c(sv_{i^*}) = c(sv_{i^*}) + 1$;
8 Return $(H, c)$ ;
```

in $O(n^2(m + n \ \log \ n))$ time. As line 4 is executed n times and the rest of Algorithm 1 can be executed efficiently, Algorithm 1 runs in $O(n^3(m+n \ \log \ n))$ time. □

We next collect some basic properties of minimal even extensions for $(2, k)$-connectivity.

Proposition 3 *Let $(H = (V+s, E), c)$ be a minimal even extension for $(2, k)$-connectivity of a capacitated graph $(G = (V, E_0), c_0)$. Then the following hold:*

(a) (H, c) is $(2, k)$-connected in V,
(b) $c(\delta_H(s))$ is even,
(c) $c(sv)$ is even for all $u \in U_{(H,c)}$,
(d) $(H, c)^2_v$ is not $(2, k)$-connected in V for any $v \in V$.

Proof We obtain (a) as an immediate consequence of the construction of Algorithm 1.

If the if-condition in line 5 is not satisfied, $c(\delta_H(s))$ is even and remains unchanged in the rest of the algorithm. Otherwise, $c(\delta_H(s))$ is odd and is augmented by 1 in line 7. This yields (b).

We denote by (H_i, c_i) the capacitated graph defined in line 4 in iteration i. Since line 4 is executed, for all $i \in \{1, \ldots, n\}$ with $c_i(sv_i) \geq 1$, there exists a biset X^i such that $v_i \in X^i_I$ and $f_{(H_i,c_i)}(\mathsf{X}^i) = 2k$. If the if-condition in line 5 is not satisfied, then $f_{(H,c)}(\mathsf{X}^i) \leq f_{(H^i,c^i)}(\mathsf{X}^i) = 2k$ for all $i \in \{1, \ldots, n\}$ with $c_i(sv_i) \geq 1$ and so $U_{(H,c)} = \emptyset$. If the if-condition in line 5 is satisfied, then i^* is defined in line 6. For all $i \in \{1, \ldots, i^* - 1\}$ with $c_i(sv_i) \geq 1$, since $c_i(sv_i) \geq 1, c_i(sv_{i^*}) = 2k$ and $f_{(H_i,c_i)}(\mathsf{X}^i) = 2k$, we have $v_{i^*} \notin X^i_I$. This yields $f_{(H,c)}(\mathsf{X}^i) \leq f_{(H^i,c^i)}(\mathsf{X}^i) = 2k$. For all $i \in \{i^*, \ldots, n\}$, $c(sv_i)$ is even after the execution of line 7. This proves (c).

For any $v_i \in V$ with $c(sv_i) \geq 1$, we have $f_{(H,c)}(\mathsf{X}^i) \leq f_{(H^i,c^i)}(\mathsf{X}^i) + 1 = 2k + 1$. This proves (d). □

We now give the following theorem which is the reason for us considering minimal even extensions for $(2, k)$-connectivity. Its proof can be found in [9]. It shows that minimum augmentations for $(2, k)$-connectivity can be computed from minimal even extensions for $(2, k)$-connectivity.

Theorem 3 *Let $(G = (V, E_0), c_0)$ be a capacitated graph, $(H = (V + s, E), c)$ a minimal even extension for $(2, k)$-connectivity of (G, c_0) and let (H', c') be obtained from (H, c) by a complete admissible splitting off. Then $(H', c') - s$ is a minimum $(2, k)$-connected augmentation of (G, c).*

We now prove an important result that shows that when finding a complete admissible splitting off, we do not need to worry about obstacles.

Lemma 9 *Let $(H = (V + s, E), c)$ be a minimal even extension for $(2, k)$-connectivity of a capacitated graph (G, c_0) and let (H_1, c_1) be obtained from (H, c) by a series of admissible splitting offs. Then (H_1, c_1) contains no obstacle.*

Proof Suppose that (H_1, c_1) contains an obstacle $\mathcal{B}$ with special vertex t. By (1), $c_1(st)$ is odd and $t \in U_{(H_1,c_1)} \subseteq U_{(H,c)}$. It follows by Proposition 3(c) that $c(st)$ is even, so st was split off with an edge sv. By Proposition 3(d), we have $v \neq t$. Let (H_2, c_2) be the capacitated graph such that $(H_1, c_1) = (H_2, c_2)^1_{t,v}$.

Claim 1 *(H_2, c_2) contains no obstacle.*

Proof Suppose otherwise, so (H_2, c_2) contains an obstacle $\mathcal{B}'$. By (1), the special vertex of $\mathcal{B}'$ cannot be t because $c_2(st) = c_1(st) + 1$ is even. It follows by (4) that $t \in B'_I$ for some $\mathsf{B}' \in \mathcal{B}'$. Now (2) yields $f_{(H_2,c_2)}(\mathsf{B}') = 2k$ which contradicts that by (1), $t \in U_{(H_1,c_1)} \subseteq U_{(H_2,c_2)}$. □

By Claim 1 and Theorem 2, (H_2, c_2) has a complete admissible splitting off. In particular, as $c_2(st) \geq 2$, there exist $x, y \in N_{H_2}(s)$ (possibly $x = y$) such that $(H_3, c_3) = ((H_2, c_2)^1_{t,x})^1_{t,y}$ is $(2, k)$-connected in V. By Proposition 3(d), $(H, c)^1_{t,t} = (H, c)^2_t$ is not $(2, k)$-connected in V and hence neither is $(H_2, c_2)^1_{t,t}$, so $x, y \neq t$. Obviously $x, y \neq v$, so, by (4), $x, y \in \bigcup_{\mathsf{B} \in \mathcal{B}} B_I$. Then, by (2), we have $2 + |\mathcal{B}|k \leq 2 + \sum_{\mathsf{B} \in \mathcal{B}} c_3(\delta_{H_3 - t}(B_I)) = \sum_{\mathsf{B} \in \mathcal{B}} c_2(\delta_{H_2 - t}(B_I)) = 1 + \sum_{\mathsf{B} \in \mathcal{B}} c_1(\delta_{H_1 - t}(B_I)) = 1 + |\mathcal{B}|k$, a contradiction. □

We are now ready to give a first naive algorithm for finding a complete admissible splitting off of minimal even extensions for $(2, k)$-connectivity.

Algorithm 2 Naive splitting off

```
Input: A minimal even extension for (2,k)-connectivity $(H = (V + s, E), c)$ of a
       capacitated graph $(G, c_0)$.
Output: A minimum (2,k)-connected augmentation of $(G, c_0)$.
1 for $u \neq v \in N_H(s)$ do
2     $(H, c) = (H, c)^{max}_{u,v}$;
3 Return $(H, c) - s$ ;
```

By Lemma 9, no obstacle is created during the execution of Algorithm 2 and so, by Theorems 2 and 3, the output of Algorithm 2 is a minimum $(2, k)$-connected augmentation of (G, c_0). As line 2 is executed at most n^2 times and by Lemma 8, Algorithm 2 runs in $O(n^4(m + n\log n))$ time. Together with Algorithm 1, this yields an $O(n^4(m + n\log n))$ time algorithm for the $(2, k)$-connectivity augmentation problem.

4 A fast splitting off algorithm

This section is dedicated to refining Algorithm 2 in order to improve its running time from $O(n^4(m+n\log n))$ time to $O(n^3(m+n\log n))$ time. Together with Algorithm 1 and Theorem 3, this yields an $O(n^3(m+n\log n))$ time algorithm for the $(2, k)$-connectivity augmentation problem.

While Algorithm 2 executes maximal splitting offs for all pairs of edges incident to s and therefore executes maximal splitting offs for a number of pairs which is quadratic in n before terminating, the refined version, Algorithm 3, carefully chooses the pairs to be split off. This allows to terminate after a number of maximal splitting offs which is linear in n.

In order to achieve this, Algorithm 3 maintains not only a capacitated graph that is obtained from the minimal even extension for $(2, k)$-connectivity that is its input by a series of splitting offs. It also stores the information obtained from the fact that certain pairs are not admissible in the form of a biset X. If two edges incident to s both have their second endvertex in the inner set of X, their splitting off is not admissible. The maintenance of X therefore allows to avoid attempts of splitting offs of pairs which are known to be nonadmissible.

During each iteration of the algorithm, we execute one or two maximal splitting offs. If none of these maximal splitting offs delete an edge incident to s, we modify X. The number of neighbors of s which is not covered by X_O never increases. Further, after a small constant number of iterations of our algorithm, either an edge incident to s is deleted or the number of neighbors of s not covered by X_O decreases. This allows to obtain the desired running time.

In the first part of this section, we show a key lemma that is needed to modify X in a favorable way. After, we describe the algorithm in the form a pseudocode. Finally, we prove the correctness of the algorithm and analyze its running time.

4.1 Key lemma

This part gives a result that allows to modify the biset X.

Lemma 10 *Let $(H = (V + s, E), c)$ be a capacitated graph that is $(2, k)$-connected in V for some $k \geq 2$ and has a complete admissible splitting off. Let X be a horrifying biset, $u \in X_I \cap N_H(s), v \in N_H(s) - X_O$, Y a biset blocking (su, sv), $z \in (X_I - Y_I) \cap N_H(s)$ and suppose that $\mathsf{X} \cup \mathsf{Y}$ is not horrifying. Let Z be a biset blocking (sv, sz). Then $\mathsf{X} \cup \mathsf{Y} \cup \mathsf{Z}$ is horrifying.*

Proof By Lemma 3, $w(\mathsf{X}) = w(\mathsf{Y}) = \{p\}$ for some vertex $p \in V$. Hence $u, v \in Y_I$. Let $\boldsymbol{(H', c')} = (H, c) - p$ and $\boldsymbol{d'}(S) := c'(\delta_{H'}(S))$ for $S \subseteq V - p$. Observe that (H', c') is k-edge-connected in V since (H, c) is $(2,k)$-connected in V. This yields that $d'(S) \geq k$ for any $\emptyset \neq S \subset V - p$.

We distinguish two cases depending on where the wall of Z is located.

Case 1 $w(\mathsf{Z}) = \{p\}$.

We show that in this case $\mathsf{X} \cup \mathsf{Y} \subseteq \mathsf{Z}$ and hence $\mathsf{X} \cup \mathsf{Y} \cup \mathsf{Z} = \mathsf{Z}$ is horrifying. For the sake of a contradiction, suppose that $(X_I \cup Y_I) - Z_I \neq \emptyset$. In order to use some symmetry arguments, let $\boldsymbol{A^1} = X_I$, $\boldsymbol{A^2} = Y_I$ and $\boldsymbol{A^3} = Z_I$. Then, by $z \in (X_I \cup Z_I) - Y_I$, $v \in (Y_I \cup Z_I) - X_I$, and the assumption that $(X_I \cup Y_I) - Z_I \neq \emptyset$, we obtain $(A^i \cup A^j) - A^\ell \neq \emptyset$ whenever $\{i, j, \ell\} = \{1, 2, 3\}$. Since X, Y and Z are horrifying bisets whose wall is $\{p\}$, we have $d'(A^i) \leq (2k+1) - k = k + 1$ for $i \in \{1, 2, 3\}$.

Claim 2 $A^\ell \subseteq A^i \cup A^j$ *whenever* $\{i, j, \ell\} = \{1, 2, 3\}$.

Proof Suppose that $A^\ell - (A^i \cup A^j) \neq \emptyset$ for some $\{i, j, \ell\} = \{1, 2, 3\}$. By $z \in X_I \cap Z_I \cap N_H(s), v \in Y_I \cap Z_I \cap N_H(s)$ and $u \in Y_I \cap X_I \cap N_H(s)$, we have $A^i \cap A^j \neq \emptyset$ and $\overline{d'}(A^i \cup A^j, A^\ell) \geq 2$. As $A^i \cap A^j \neq \emptyset$ and $(A^i \cup A^j) - A^\ell \neq \emptyset$, it follows that $d'(A^i \cap A^j), d'((A^i \cup A^j) - A^\ell) \geq k$. Then, as $d'(A^i), d'(A^j), d'(A^\ell) \leq k + 1$, Proposition 1($a$) and ($b$) yield

$$\begin{aligned}
3(k+1) - k &\geq d'(A^i) + d'(A^j) - d'(A^i \cap A^j) + d'(A^\ell) \\
&\geq d'(A^i \cup A^j) + d'(A^\ell) \\
&= d'((A^i \cup A^j) - A^\ell) + d'(A^\ell - (A^i \cup A^j)) + 2\overline{d'}(A^i \cup A^j, A^\ell) \\
&\geq 2k + 4,
\end{aligned}$$

a contradiction. □

Claim 3 *For any* $i \neq j \in \{1, 2, 3\}$*, we have* $\overline{d'}(A^i, A^j) \leq 1$.

Proof Let ℓ be the remaining element in $\{1, 2, 3\} - \{i, j\}$. By assumption, $(A^i \cup A^\ell) - A^j \neq \emptyset$. By Claim 2, $A^\ell - (A^i \cup A^j) = \emptyset$. This yields $A^i - A^j = ((A^i \cup A^\ell) - A^j) - (A^\ell - (A^i \cup A^j)) \neq \emptyset$. Similarly, $A^j - A^i \neq \emptyset$. It follows that $d'(A^i - A^j), d'(A^j - A^i) \geq k$. As $d'(A_i), d'(A_j) \leq k+1$, Proposition 1($b$) yields $2\overline{d'}(A^i, A^j) = d'(A^i) + d'(A^j) - d'(A^i - A^j) - d'(A^j - A^i) \leq 2(k+1) - 2k = 2$. □

Claim 4 $d'(X_I \cup Y_I) \geq k + 2$.

Proof As (H, c) has a complete admissible splitting off, there exist $x, y \in N_H(s)$ such that $(H'', c'') := ((H, c)^1_{z,x})^1_{v,y}$ is $(2,k)$-connected in V. Since $z \in X_I \cap Z_I, v \in Y_I \cap Z_I$ and $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$ are horrifying, we obtain $x \notin X_I \cup Z_I$ and $y \notin Y_I \cup Z_I$. Then, by Claim 2, $x, y \notin X_I \cup Y_I$. If $x = p = y$, then $f_{(H'',c'')}(\mathsf{Z}) = f(\mathsf{Z}) - 2$, so, since (H'', c'') is $(2,k)$-connected in V and Z is horrifying, we have $2k \leq f_{(H'',c'')}(Z) = f(\mathsf{Z}) - 2 \leq (2k+1) - 2 = 2k - 1$, a contradiction. So one of x and y belongs to $V - (\mathsf{X} \cup \mathsf{Y})_O$. Then, since $\mathsf{X} \cup \mathsf{Y}$ is not horrifying, we have $d'(X_I \cup Y_I) = f(\mathsf{X} \cup \mathsf{Y}) - |w(\mathsf{X} \cup \mathsf{Y})|k \geq (2k+2) - k = k + 2$. □

By Claim 2, we have $X_I \cup Y_I = X_I \cup Y_I \cup Z_I$ and every edge that contributes to $d'(X_I \cup Y_I \cup Z_I)$ also contributes to $d'(A^i, A^j)$ for some $i \neq j \in \{1,2,3\}$. By $k \geq 2$, and Claims 4 and 3, we obtain $4 \leq k+2 \leq d'(X_I \cup Y_I) = d'(X_I \cup Y_I \cup Z_I) \leq \sum_{i \neq j} \overline{d'}(A^i, A^j) \leq 3$, a contradiction. This finishes the case. □

Case 2 $w(\mathsf{Z}) \neq \{p\}$.

By symmetry, we may suppose that $v \in Z_I$. As $z \in X_I, v \in N_H(s) - X_I$, Z blocks (sz, sv) and $w(\mathsf{Z}) \neq \{p\}$, we may apply Lemma 3 to obtain that $\mathsf{X} \cup \mathsf{Z}$ is horrifying. Since $z, v \in (\mathsf{X} \cup \mathsf{Z})_I$, $\mathsf{X} \cup \mathsf{Z}$ blocks (sz, sv). As $v \in Y_I$, $z \in N_H(s) - Y_I$, we may apply Lemma 3 once more and obtain that either $\mathsf{X} \cup \mathsf{Y} \cup \mathsf{Z}$ is horrifying or $w(\mathsf{X} \cup \mathsf{Z}) = w(\mathsf{Y})$. In the first case we are done, so we may suppose that $w(\mathsf{X} \cup \mathsf{Z}) = w(\mathsf{Y}) = \{p\}$. If $Y_I - (X_I \cup Z_I) = \emptyset$, we obtain that $\mathsf{X} \cup \mathsf{Y} \cup \mathsf{Z} = \mathsf{X} \cup \mathsf{Z}$ is horrifying.

We may therefore suppose that $Y_I - (X_I \cup Z_I) \neq \emptyset$. As $z \in (X_I \cup Z_I) - Y_I$, we obtain that $d'(Y_I - (X_I \cup Z_I)) \geq k$ and $d'((X_I \cup Z_I) - Y_I) \geq k$. Also, as $\mathsf{X} \cup \mathsf{Z}$ and Y are horrifying bisets whose wall is $\{p\}$, we obtain $d'(X_I \cup Z_I) \leq k+1$ and $d'(Y_I) \leq k+1$. As $u, v \in (X_I \cup Z_I) \cap Y_I \cap N_H(s)$, Proposition 1 (b) yields $2(k+1) \geq d'(X_I \cup Z_I) + d'(Y_I) = d'((X_I \cup Z_I) - Y_I) + d'(Y_I - (X_I \cup Z_I)) + 2\overline{d'}(X_I \cup Z_I, Y_I) \geq 2k+4$, a contradiction. □

4.2 Decription of the algorithm

We are now ready to describe the algorithm in the form of a pseudocode. It first is initialized with the input capacitated graph and an empty biset X. The main part of the algorithm consists of a while-loop in which maximal splitting offs are executed and X is modified. In order to apply the structure found in Lemma 10, we need X to be horrifying. Therefore, the first part of the while-loop in line 3 to 7 is concerned with reinitializing X with a horrifying biset if X is not horrifying before the iteration. The main part from line 9 to 24 deals with the case that X is horrifying. Algorithm 3 then performs up to two maximal splitting offs of pairs of edges incident to s whose choice depends on X. If none of these two maximal splitting offs leads to the deletion of an edge incident to s, Algorithm 3 augments X in a beneficial way. After the last iteration of the while-loop, Algorithm 3 outputs the obtained capacitated graph after deleting s.

Algorithm 3 Complete admissible splitting off

```
Input: A minimal even extension for $(2,k)$-connectivity $(H = (V+s, E), c)$ of a
       capacitated graph $(G, c_0)$.
Output: A minimum $(2,k)$-connected augmentation of $(G, c_0)$.
$\mathsf{X} := (\emptyset, \emptyset)$;
while $|N_H(s)| \geq 2$ do
    if $\mathsf{X}$ is not horrifying then
        let $u \neq v \in N_H(s)$;
        $(H, c) = (H, c)^{max}_{u,v}$;
        if $c(su), c(sv) > 0$ then
            let $\mathsf{X}$ be a biset blocking $(su, sv)$;
    else
        let $u \in X_I \cap N_H(s)$;
        let $v \in N_H(s) - X_O$;
        $(H, c) = (H, c)^{max}_{u,v}$;
        if $c(su), c(sv) > 0$ then
            let $\mathsf{Y}$ be a biset blocking $(su, sv)$;
            if $\mathsf{X} \cup \mathsf{Y}$ is horrifying then
                $\mathsf{X} = \mathsf{X} \cup \mathsf{Y}$;
            else
                if $X_I \cap N_H(s) \subseteq Y_I$ then
                    $\mathsf{X} = \mathsf{Y}$;
                else
                    let $z \in (X_I - Y_I) \cap N_H(s)$;
                    $(H, c) = (H, c)^{max}_{v,z}$;
                    if $c(sv), c(sz) > 0$ then
                        let $\mathsf{Z}$ be a biset blocking $(sv, sz)$;
                        $\mathsf{X} = \mathsf{X} \cup \mathsf{Y} \cup \mathsf{Z}$;
return $(H, c) - s$;
```

4.3 Analysis of the algorithm

This last section is dedicated to the analysis of Algorithm 3. We first give a collection of properties of the capacitated graphs and bisets obtained at intermediate steps of Algorithm 3. We then conclude the correctness and the running time of Algorithm 3.

Proposition 4 *The following hold for every iteration i of the while loop starting in line 2:*

(a) All steps in iteration i are well-defined.
(b) (H, c) is $(2, k)$-connected in V and has a complete admissible splitting off after iteration i.
(c) $f(\mathsf{X}) \leq 2k + 1, |w(\mathsf{X})| \leq 1$ and $X_O \neq V$ after iteration i.

Proof By Lemma 9, (b) holds before iteration 1 and trivially (c) also holds. Inductively, we may suppose that (a), (b) and (c) hold for all iterations $1, \ldots, i-1$. We show that they also hold for iteration i:

(a): The choice of u and v in line 4 is justified by the fact that the while-condition in line 2 was satisfied. If X is horrifying, the choice of u in line 9 is justified and Lemma 4 justifies the choice of v in line 10. The choice of z in line 20 is justified by the fact that the if-condition in line 17 was not satisfied. The horrifying bisets in lines 7,13 and 23 exist by Lemma 2.

(b): It follows immediately from the construction that $c(\delta_H(s))$ always remains even and (H, c) always remains $(2, k)$-connected in V. By Lemma 9, no obstacle in (H, c) can ever be created. Now Theorem 2 yields that (H, c) has a complete admissible splitting off after iteration i.

(c): As splitting offs do not increase f, it suffices to prove that X either remains unchanged or is horrifying after iteration i. First suppose that the if-condition in line 3 is satisfied. If the if-condition in line 6 is not satisfied, X remains unchanged. Otherwise, X is replaced by a horrifying biset in line 7.

Now suppose that the else-case starting in line 8 is executed. If the if-condition in line 12 is not satisfied, X remains unchanged, so suppose otherwise. If the if-condition in line 14 is satisfied, X is replaced by a horrifying biset in line 15, so suppose otherwise. If the if-condition in line 17 is satisfied, X is replaced by a horrifying biset in line 18. So suppose that the else-case starting in line 19 is executed. If the if-condition in line 22 is not satisfied, X remains unchanged. Otherwise, $u \in X_I \cap N_H(s), v \in N_H(s) - X_O, \mathsf{Y}$ blocks $(su, sv), z \in (X_I - Y_I) \cap N_H(s), \mathsf{Z}$ blocks (sv, sz) and $\mathsf{X} \cup \mathsf{Y}$ is not horrifying. Together with (b), Lemma 10 yields that $\mathsf{X} \cup \mathsf{Y} \cup \mathsf{Z}$ is horrifying, so X is replaced by a horrifying biset in line 24. $\square$

We now obtain the correctness of our algorithm as a simple corollary:

Theorem 4 *If Algorithm 3 terminates, it outputs a minimum $(2, k)$-connected augmentation of (G, c_0).*

Proof By Theorem 2, it is sufficient to prove that Algorithm 3 executes a complete admissible splitting off of the input capacitated graph. Let (H, c) be the current capacitated graph ffter the last iteration of the while-loop. By construction, $|N_H(s)| \leq 1$. If $N_H(s)$ contains a single vertex u, by Proposition 3 (d), (su, su) is not admisible in the input capacitated graph. As (H, c) has been obtained by admissible splitting offs, (su, su) is neither admissible in (H, c). It follows that (H, c) does not have a complete admissible splitting, a contradiction to Proposition 4 (b). Hence s is an isolated vertex in (H, c) and so, by Proposition 4 (b), Algorithm 3 executed a complete admissible splitting off. $\square$

The remaining part is concerned with the running time analysis of Algorithm 3.

Theorem 5 *Algorithm 3 runs in $O(n^3(m + n \log n))$ time.*

Proof Obviously, the initialization of Algorithm 3 and the final output can be executed efficiently. Also, it follows from Lemma 8 that every iteration of the while-loop starting in line 2 can be executed in $O(n^2(m + n \log n))$ time. It remains to show that the while-loop runs a linear number of times. In order to do this, we define the parameter $\boldsymbol{M} = |N_H(s)| + |N_H(s) - X_O|$. The decrease of M measures the progress of our algorithm. We next prove two claims that show that M decreases regularly.

Claim 5 *If in an iteration i the else-case starting in line 8 is executed, then M decreases in iteration i.*

Proof If the if-condition in line 12 is not satisfied, $|N_H(s)|$ decreases in iteration i and X remains unchanged, so suppose otherwise. If the if-condition in line 14 is satisfied, then in line 15 X is replaced by a biset containing X and v leaves $N_H(s) - X_O$, so $|N_H(s) - X_O|$ decreases and $N_H(s)$ remains unchanged. Otherwise, by Lemma 3, we have $w(\mathsf{X}) = w(\mathsf{Y})$. Therefore, if the if-condition in line 17 is satisfied, X is replaced by Y in line 18 and Y satisfies $X_O \cup \{v\} \subseteq Y_O$, so $|N_H(s) - X_O|$ decreases and $N_H(s)$ remains unchanged. So suppose that the else-case starting in line 19 is executed. If the if-condition in line 22 is not satisfied, $|N_H(s)|$ decreases in iteration i and X remains unchanged, so suppose otherwise. Then in line 23 X is replaced by a biset containing X and v leaves $N_H(s) - X_O$, so $|N_H(s) - X_O|$ decreases and $N_H(s)$ remains unchanged. □

Claim 6 *If in an iteration i the if-condition in line 3 is satisfied, then either M decreases in iteration i or M remains unchanged in iteration i and decreases in iteration $i + 1$.*

Proof If the if-condition in line 6 is not satisfied, $|N_H(s)|$ decreases in iteration i and X remains unchanged, so suppose otherwise. By Proposition 4(c), $|X_O \cap N_H(s)| \leq 2$ before iteration i. As X is replaced by a horrifying biset in line 7, we have $|X_O \cap N_H(s)| \geq 2$ after iteration i. As $N_H(s)$ remains unchanged, M does not increase in iteration i. As X is horrifying after iteration i, it follows that in iteration $i+1$ the else-case starting in line 8 is executed, so M decreases in iteration $i + 1$ by Claim 5. □

Claims 5 and 6 show that M never increases and decreases in at least one of two consecutive iterations. Further observe that M is always an integer satisfying $0 \leq M \leq 2n$. It follows that the while-loop runs at most $4n$ times. This finishes the proof. □

5 Acknowledgement

We thank the anonymous referees to kindly ask us to simplify our algorithm. This allowed us to find a new approach and to significantly simplify the algorithm and the proofs.

References

1. J. Bang-Jensen, A. Frank, B. Jackson, Preserving and increasing local edge-connectivity in mixed graphs, *SIAM J. Discrete Math.* Vol. 8 No. 2 (1995) 155-178.
2. J. Bang-Jensen, H. Gabow, T. Jordán, Z. Szigeti, Edge-connectivity augmentation with partition constraints, *SIAM J. Discrete Math.* Vol. 12 No. 2 (1999) 160-207.
3. J. Bang-Jensen, B. Jackson, Augmenting hypergraphs by edges of size two, *Math. Program.* Vol. 84 No. 3 (1999) 467-481.

4. J. Bang-Jensen, T. Jordán, Edge-connectivity augmentation preserving simplicity, *SIAM J. Discrete Math.* Vol. 11 No. 4 (1998) 603-623.
5. A. Benczúr, A. Frank, Covering symmetric supermodular functions by graphs, *Math. Program.* Vol. 84 No. 3 (1999) 483-503.
6. A. Bernáth, R. Grappe, Z. Szigeti, Augmenting the edge-connectivity of a hypergraph by adding a multipartite graph, *Journal of Graph Theory*, 72/3 (2013) 291-312.
7. A. Bernáth, R. Grappe, Z. Szigeti, Partition constrained covering of a symmetric crossing supermodular function by a graph, *SIAM J. on Discrete Math.* 31/1 (2017) 335-382.
8. G. R. Cai, Y. G. Sun, The minimum augmentation of any graph to k-edge-connected graphs, *Networks*, 19 (1989) 151-172.
9. O. Durand de Gevigney, Z. Szigeti, On $(2, k)$-connected graphs, *Journal of Graph Theory*, 91/4 (2019) 305-325.
10. K. P. Eswaran, P. E. Tarjan, Augmentation problems, *SIAM J. Comput.*, 5(4): (1976) 653-665.
11. A. Frank, Augmenting graphs to meet edge-connectivity requirements, *SIAM J. Discrete Math.* Vol. 5 No. 1 (1992) 22-53.
12. A. Frank, Connections in Combinatorial Optimization, Oxford University Press, 2011.
13. A. Frank, T. Jordán, Minimal Edge-Coverings of Pairs of Sets. *J. Comb. Theory*, Ser. B 65(1): (1995) 73-110.
14. B. Jackson, T. Jordán, Independence free graphs and vertex connectivity augmentation. *J. Comb. Theory*, Ser. B 94(1): (2005) 31-77.
15. A. Kaneko, K. Ota. On minimally (n, λ)-connected graphs. *J. Comb. Theory*, Ser. B, 80(1): (2000) 156-171.
16. H. Nagamochi, T. Ibaraki, Graph connectivity and its augmentation: applications of MA ordering, *Disrete Applied Mathematics* 123 (2002) 447-472.
17. Z. Szigeti, On edge-connectivity augmentation of graphs and hypergraphs, William Cook, László Lovász, Jens Vygen (Editors): Research Trends in Combinatorial Optimization. Springer, Berlin 2009, 483-521.
18. T. Watanabe, A. Nakamura, Edge-connectivity augmentation problems, *Comp. System Sci.* 35 (1987) 96-144.