Theory of Computing Systems manuscript No.
(will be inserted by the editor)

On the Cycle Augmentation Problem: Hardness and Approximation Algorithms

Waldo Gálvez · Fabrizio Grandoni · Afrouz Jabal Ameli · Krzysztof Sornat

Received: date / Accepted: date

Abstract In the k-Connectivity Augmentation Problem we are given a k-edge-connected graph and a set of additional edges called *links*. Our goal is to find a set of links of minimum cardinality whose addition to the graph makes it $(k+1)$-edge-connected. There is an approximation preserving reduction from the mentioned problem to the case $k = 1$ (a.k.a. the Tree Augmentation Problem or TAP) or $k = 2$ (a.k.a. the Cactus Augmentation Problem or CacAP). While several better-than-2 approximation algorithms are known for TAP, nothing better is known for CacAP (hence for k-Connectivity Augmentation in general).

As a first step towards better approximation algorithms for CacAP, we consider the special case where the input cactus consists of a single cycle, the *Cycle Augmentation* Problem (CycAP). This apparently simple special case retains part of the hardness of the general case. In particular, we are able to show that it is APX-hard.

In this paper we present a combinatorial $\left(\frac{3}{2}+\varepsilon\right)$-approximation for CycAP, for any constant $\varepsilon > 0$. We also present an LP formulation with a matching integrality gap: this might be useful to address the general case of the problem.

Keywords Approximation algorithms · Connectivity augmentation · Cactus augmentation · Cycle augmentation.

1 Introduction

The basic goal of *Survivable Network Design* is to construct low cost networks that provide connectivity guarantees between pre-specified sets of nodes even after the failure of a few edges/nodes (in the following we will focus on the edge failure case). This has many applications, e.g., in transportation and telecommunication networks.

A relevant subclass of these problems is given by *Network Augmentation* problems. Here the goal is to *augment* a given graph $G = (V, E)$ by adding extra edges taken from a given set L (*links*), so as to satisfy given (edge-)connectivity requirements. Several such problems are NP-hard, and in most cases the best known approximation factor is 2 due to Jain [18].

In this paper we focus on the following k-Connectivity Augmentation Problem (k-CAP). Given a k-(edge)-connected[1] undirected graph $G = (V, E)$ and a collection L of extra edges (*links*), the goal is to find a minimum cardinality subset $A \subseteq L$ such that $G' = (V, E \cup A)$ is $(k+1)$-connected. Dinitz et al. [9] presented an approximation preserving reduction from this problem to the case $k = 1$ for odd k, and $k = 2$ for even k. This motivates a deeper understanding of the latter two special cases.

Partially supported by the SNSF Grant 200021_159697/1, the SNSF Excellence Grant 200020B_182865/1, the National Science Centre, Poland, grant numbers 2015/17/N/ST6/03684, 2015/18/E/ST6/00456 and 2018/28/T/ST6/00366. K. Sornat was also supported by the Foundation for Polish Science (FNP) within the START programme.

W. Gálvez, F. Grandoni, A. Jabal Ameli
IDSIA, Lugano, Switzerland.
E-mail: {waldo,fabrizio,afrouz}@idsia.ch

K. Sornat
University of Wrocław, Wrocław, Poland
E-mail: krzysztof.sornat@cs.uni.wroc.pl

[1] We recall that $G = (V, E)$ is k-connected if for every set of edges $F \subseteq E$, $|F| \leq k-1$, the graph $G' = (V, E \setminus F)$ is connected.

The case $k = 1$ is also known as the Tree Augmentation Problem (TAP). The reason for this name is that any 2-connected component of the input graph G can be contracted, hence leading to a tree. For this problem several better-than-2 approximation algorithms are known [1,6,10,11,16,23,27]. The case $k = 2$ is also known as the Cactus Augmentation Problem (CacAP) since, similarly to the previous case, the input graph can be assumed to be a cactus[2] [9]. However, here the best-known approximation factor is still 2 [18] (implying the same for k-CAP in general).

For all the mentioned problems it makes sense to consider the weighted version, where links have non-negative integral weights, and the goal is to find a minimum weight (rather than minimum cardinality) subset of links A with the desired properties. In particular we will speak about Weighted TAP (WTAP) and Weighted CacAP (WCacAP). Here the best-known approximation factor is 2 in both cases [18]. Moreover, improving on that approximation factor for WTAP is considered as a major open problem in the area. We also notice that we can turn a WTAP instance into an equivalent WCacAP instance by replacing each edge with two parallel edges. Hence, approximating WCacAP is not any easier than approximating WTAP (and the same holds for the corresponding unweighted versions).

1.1 Our Results

As mentioned before, CacAP contains TAP as a special case when the cactus consists of several short cycles. Hence, in order to make progress on CacAP, it makes sense to consider the somehow complementary case where the input cactus consists of a single cycle of n nodes. We call the corresponding subproblem the Cycle Augmentation Problem (CycAP), and its weighted version Weighted CycAP (WCycAP). To the best of our knowledge, these special cases were not studied before. However, as we will see, they still retain part of the difficulties of the general cactus case. In more detail, we achieve the following main results:

Hardness of Approximation. We are able to show that WCycAP is as hard to approximate as WCacAP. Therefore, improving on a 2-approximation for WCycAP would imply a major breakthrough in the area (in particular, it would imply the same for WTAP). This also justifies a more careful investigation of CycAP. In our opinion it is a priori not so obvious that CycAP is even NP-hard. Indeed, the special case of TAP (and even of WTAP) where the input graph is a path can be solved exactly in polynomial time. The case of an input cycle might closely remind the path case. Here we show that this intuition is not correct: we prove that CycAP is NP-hard and even APX-hard via a simple but non-trivial adaptation of the proofs in [14,22]. In particular, we need one extra step in the reduction where we turn an intermediate CacAP instance into a CycAP one while maintaining certain properties of the optimal solution.

Approximation algorithms. As discussed, the best we can hope for CycAP is some constant $c > 1$ approximation. We present better-than-2 approximation algorithms for this problem. In particular, we present a simple $\frac{5}{3}$-approximation, and a slightly more complex $(3/2+\varepsilon)$-approximation for any constant $\varepsilon > 0$. Notice that the latter approximation factor is not far from the best known approximation factor for TAP which is equal to 1.458 [16]. Our algorithms are purely combinatorial, and they consist of two main phases. In the first phase, we *greedily* add some links to the solution under construction and *contract* them. At the end of this phase we achieve an instance of CacAP that can be solved exactly in polynomial time. In particular, for the $\frac{5}{3}$-approximation this reduces to computing a spanning tree, while for the $(3/2+\varepsilon)$-approximation we use an FPT algorithm parameterized by a proper notion of maximum *length* of a link.

LP gaps. The recent literature on TAP approximation [1,11,16] shows that finding strong LP relaxations for the problem can be very helpful to design improved approximation algorithms. In the same spirit, we tried to address the problem of finding LP relaxations for CycAP with small integrality gap. For both TAP and CacAP (hence CycAP) one can define a natural and simple standard cut LP (more details later). While for TAP it was recently shown that the standard cut LP has integrality gap smaller than 2 [28], interestingly for CycAP (hence for CacAP) the standard cut LP has integrality gap 2. Here we present a stronger LP that, for any $\varepsilon > 0$, has integrality gap at most $\frac{3}{2} + \varepsilon$ (hence matching the approximation ratio of our algorithm). In our opinion this could be useful for future work on CacAP approximation.

[2] We recall that a *cactus* G is a connected undirected graph in which every edge belongs to exactly one cycle. For technical reasons it is convenient to allow cycles of length 2 consisting of parallel edges.

1.2 Related work

As mentioned before, the best known result in terms of polynomial time approximation algorithms for k-CAP is a 2-approximation proposed by Jain [18]. However, if the set of links is equal to $V \times V$ it is possible to solve this problem optimally [32]. More recently, this problem has been studied in the framework of Fixed-Parameter Tractability: Végh and Marx [26] proved that this problem is in FPT when parameterized by the size of the optimal solution, and later the running time of their algorithm was further improved [2].

Tree Augmentation has been extensively studied over the past few decades. It was first shown that WTAP is NP-hard by Frederickson and Jájá [14], then that TAP is NP-hard by Cheriyan et al. [5], and later that TAP is APX-hard by Kortsarz et al. [22]. For WTAP, the best-known approximation guarantee is 2 and was first established by Frederickson and Jájá [14]. Their algorithm was later simplified by Khuller and Thurimella [20]. A 2-approximation can also be achieved by various other techniques developed later on, including a primal-dual approach [15] and iterative rounding [18]. Improvements on the factor 2 have only been obtained for restricted cases, including bounded diameter trees [7] and bounded weights [1,11, 16,28].

Regarding TAP, the first algorithm beating the approximation guarantee of 2 is due to Nagamochi [27], achieving an approximation factor of $1.815 + \varepsilon$. This factor was subsequently improved to 1.8 [10] and to 1.5 [6,23]. These results are combinatorial in nature, but LP-based results have been achieved as well. As an example, recently Nutov [28] showed that the standard cut LP for TAP has an integrality gap of at most 28/15 while a lower bound of 3/2 was known [6]. An LP-based $\left(\frac{5}{3} + \varepsilon\right)$-approximation was given by Adjiashvili [1] and then refined by Fiorini et al. [11] to obtain a $\left(\frac{3}{2} + \varepsilon\right)$-approximation (see also [3,4,25]). Both results are obtained by adding a proper family of extra constraints to the standard cut LP. Recently, Grandoni et al. [16] achieved a 1.458 approximation for TAP, which is smaller than the integrality gap of the standard cut LP.

The rest of this paper is organized as follows. In Section 2 we give some preliminary definitions and results. The approximation algorithms, LP-gaps and hardness of approximation results are discussed in Sections 3, 4 and 5 respectively.

2 Preliminaries

For a set X and element y, we use the shortcut $X \setminus y$ for $X \setminus \{y\}$, and similarly for other set operations.

Given a graph $G = (V, E)$, we let $V(G) = V$ and $E(G) = E$. Recall that in WCacAP we are given a cactus $G = (V, E)$, a set of links $L \subseteq \binom{V}{2}$ and a non-negative weight function $c : L \to \mathbb{R}_{\geq 0}$. The task is to compute a subset of links $A \subseteq L$ such that the graph $(V, E \cup A)$ is 3-edge-connected while minimizing $c(A) := \sum_{\ell \in A} c(\ell)$. The special case where G is a cycle is called WCycAP, and the unweighted versions of the above problems are called CacAP and CycAP respectively. By n we will denote the number of nodes of the considered instance of the problem.

Notice that, given an instance (G, L) of CacAP, we can check in polynomial time if the graph $(V(G), E(G) \cup L)$ is 3-edge-connected by exhaustively checking if the removal of any pair of elements from $E(G) \cup L$ disconnects the graph. Hence we will assume along this work that the instance always admits a feasible solution.

Remark 1 The 2-edge cuts of a cactus G are identified by pairs $S = \{e, e'\}$ of distinct edges belonging to the same cycle, and consist of the node sets (V', V'') of the two connected components obtained by removing S from G. A necessary and sufficient condition for a subset of links A to be a feasible solution for WCacAP is that, for any such cut S, there is at least one $\ell \in A$ crossing the cut (in which case ℓ **satisfies** the $\{e, e'\}$-cut).

Note that in the case of CycAP, Remark 1 implies that any feasible solution must be an edge cover as 2-edge cuts defined by neighboring edges of the cycle must be satisfied. Given a 2-edge cut $S = \{e, e'\}$, let L_S be the subset of links satisfying S. The standard cut LP for CycAP is as follows:

$$\begin{aligned} \min \sum_{\ell \in L} x_\ell & \qquad \text{(standard cut LP)} \\ s.t. \sum_{\ell \in L_S} x_\ell \geq 1 & \qquad \forall S : S \text{ is a 2-edge cut} \\ 0 \leq x_\ell \leq 1 & \qquad \forall \ell \in L \end{aligned}$$

Now we proceed to define a standard building block for our algorithms, the *contraction* of a link.

Definition 1 Contracting a subset of nodes W consists of the following operations: (i) remove the nodes in W and all edges/links incident to them; (ii) add a new node w and, for each original edge/link of type (y, x), $x \in W, y \notin W$, add the edge/link (y, w) (of the same weight for the case of links). Note that we do not create loops this way but may introduce parallel links. We say that (y, w) is the image of (y, x) and (y, x) is the preimage of (y, w).

We will sometimes slightly abuse notation and use the same label to denote a link and its image: the meaning will be clear from the context.

For a link $\ell = (u, v)$, we define a sequence $w_0, \ldots, w_q$ of boundary nodes $B(\ell)$ as follows. Consider a simple path from u to v in the cactus, and let $C_1, C_2, \ldots, C_q$ be the ordered sequence of cycles visited[3] by this path (possibly $q = 1$). We define w_i, $i = 1, \ldots, q-1$ as the unique common node between C_i and C_{i+1}, and set $w_0 = u$ and $w_q = v$.

Definition 2 Contracting a link ℓ is the operation of contracting its boundary nodes $B(\ell)$. We denote by $G|\ell$ the graph obtained by this operation. Contracting a set of links A is the operation of contracting any $\ell \in A$, and then continue recursively on $G|\ell$ and on the image of $A \setminus \ell$ until A becomes empty.

Note that contracting a link in a cactus yields again a cactus. We will extensively use the following standard fact.

Lemma 1 *Let (G, L) be a CacAP instance, $A \subseteq L$, and $\ell \in A$. Then A is a feasible solution for (G, L) iff the image of $A \setminus \ell$ is a feasible solution for $(G|\ell, L \setminus \ell)$.*

We require some further notation before proving the lemma. The *internal projections* $S(\ell)$ of ℓ are the links (w_i, w_{i+1}), $i = 0, \ldots, q-1$. In terms of feasibility, ℓ and $S(\ell)$ are equivalent as the following proposition states.

Proposition 1 *Let (G, L) be a CacAP instance and $\ell \in L$. Then ℓ satisfies precisely the same 2-edge cuts as $S(\ell)$.*

Proof Let $B(\ell) = (w_0, \ldots, w_q)$ and $C_1, \ldots, C_q$ be the corresponding sequence of cycles visited by a simple path between the endpoints of ℓ. Notice that pairs (w_i, w_{i+1}), $i = 0, \ldots, q-1$, subdivide each C_i into two paths next denoted as C_i' and C_i''. Trivially ℓ satisfies only cuts belonging to the cycles $C_1, \ldots, C_q$, and the same holds for $S(\ell)$. Consider any pair (e_1, e_2) belonging to some C_i. Link ℓ satisfies the corresponding cut if and only if precisely one such edge e_j belongs to C_i'. The same holds for (w_i, w_{i+1}), hence for $S(\ell)$.

In order to prove Lemma 1, let us first consider the simpler case where G is a cycle.

Lemma 2 *Let $(G = (V, E), L)$ be a CycAP instance, $A \subseteq L$, and $\ell = (u, v) \in A$. Then A is a feasible solution for (G, L) iff the image of $A \setminus \ell$ is a feasible solution for the CacAP instance $(G|\ell, L \setminus \ell)$.*

Proof Let C_1 and C_2 be the two cycles in $G|\ell$, with common node w.

Suppose first that the image of $A \setminus \ell$ is a feasible solution for $(G|\ell, L \setminus \ell)$. Consider a pair of edges $\{e_1, e_2\}$ belonging to a common cycle C_i, and the corresponding cut (S', S'') in $G|\ell$ with $w \in S''$. There must be a link $\ell' \in A \setminus \ell$ satisfying this cut in $G|\ell$. The preimage of ℓ' has one endpoint in S' and the other in $V \setminus S' = (S'' \setminus \{w\}) \cup \{u, v\}$, hence it satisfies the $\{e_1, e_2\}$-cut in G. The remaining pairs of edges $\{e_1, e_2\}$ of G satisfy $e_1 \in C_1$ and $e_2 \in C_2$, modulo symmetries. Those cuts are satisfied by ℓ in G.

Suppose now that A is feasible for (G, L). Consider a pair of edges $\{e_1, e_2\}$ belonging to a common cycle C_i. Let (S', S'') be the corresponding cut in $G|\ell$ with $w \in S''$. Since ℓ does not satisfy that cut in G, this means that there is some other link $\ell' \in A \setminus \ell$ satisfying it. The image of ℓ' has one endpoint in S' and the other in S'', hence it satisfies the $\{e_1, e_2\}$-cut.

Now we can proceed with the proof of Lemma 1.

(Proof of Lemma 1) By Proposition 1, we obtain an equivalent statement of the lemma by replacing A with the set $S(A)$ of the internal projections of links in A and replacing ℓ with its internal projection $S(\ell)$.

Let $B(\ell) = (w_0, \ldots, w_q)$ and $C_1, \ldots, C_q$ be the corresponding sequence of cycles visited by a simple path between the endpoints of ℓ. Consider any cycle C not in the above list. Then trivially any pair of edges in C is covered by links in $S(A) \setminus S(\ell)$. Therefore it is sufficient to consider pairs of edges e_1, e_2 belonging to the same cycle C_i. Let $\ell_i = (w_i, w_{i+1})$ be the internal projection of ℓ with both endpoints in C_i, and define similarly $S_i(A)$ w.r.t. $S(A)$. Then it is sufficient to show that $S_i(A)$ is a feasible solution for the CycAP instance induced by C_i if and only if $S_i(A) \setminus \ell_i$ is a feasible solution for the CycAP instance induced by $C_i|\ell_i$, which follows from Lemma 2.

[3] A path visits a cycle iff it includes an edge from the cycle.

3 Approximation Algorithms for Cycle Augmentation

In this section we present improved approximation algorithms for CycAP. We start with a simple $\frac{5}{3}$-approximation to illustrate the main ideas, and then present a slightly more complex $\left(\frac{3}{2}+\varepsilon\right)$-approximation. The approach we will follow in both cases is as follows: in a first phase we iteratively add a properly chosen subset of a few links to the solution under construction, and then contract them. Notice that, after the first contraction, the cycle structure may be lost and we obtain a CacAP instance instead. These choices are designed so that, at the end of the first phase, the remaining CacAP instance can be solved efficiently, which is done in a second phase with an ad-hoc algorithm. We remark that the running times of the presented algorithms is not analyzed in detail, and indeed such a task may require to devise carefully crafted data structures.

3.1 A $\frac{5}{3}$-approximation

We next describe a simple greedy algorithm that provides a $\frac{5}{3}$-approximation for CycAP. We need the following definitions.

Definition 3 A link $\ell = (u, v)$ of a CacAP instance is **internal** if both its endpoints belong to a common cycle, and *external* otherwise.

Definition 4 Given a CacAP instance, a pair of internal links $\{(u_1, v_1), (u_2, v_2)\}$ of a cycle C is **crossing** if they are node disjoint and deleting u_2 and v_2 disconnects u_1 from v_1 in C.

The kind of links that we want to add in the first stage of the algorithm are external links plus crossing pairs of links. More in detail, the algorithm has two main stages. The first stage consists of a set of rounds, where in each round we first check if there exists an external link ℓ, in which case we add it to our solution, contract it and proceed to the next round. Otherwise, if there exists a pair of (internal) crossing links ℓ' and ℓ'', we add them to our solution, contract them and proceed to the next round. If none of the two cases above applies, we are left with a CacAP instance without neither external links nor crossing pairs of links which we address in the second stage of the algorithm. We refer to this algorithm as CROSSING-FIRST. As the following lemma states, in the second stage we can efficiently compute the optimal solution.

Lemma 3 *Consider an instance $(G = (V, E), L)$ of CacAP. If there are no external links and no crossing pairs of links, then every minimal solution has size exactly $|V| - 1$ and induces a spanning tree over V.*

Proof We prove the first part of the claim by induction on $n = |V|$. The base case $n = 2$ is trivial since in this case the instance is just a cycle consisting of two parallel edges and any link must be incident to the two nodes of G (hence defining a feasible solution). For the inductive case, assume the claim is true up to instances having $n - 1$ nodes, and consider an instance of the problem defined by a cactus G having n nodes with optimal solution OPT. If G is not a cycle of length n, then it is defined by a set of cycles of length at most $n - 1$ where every link is internal, so we can apply the inductive hypothesis to each cycle independently. If G is a cycle of n nodes, then let $\ell = (u, v) \in \text{OPT}$. Contracting ℓ leads to a CacAP instance on two cycles C_1 and C_2 sharing a common node w, with $|V(C_1)| + |V(C_2)| = n$. Let OPT' be the optimal solution for the new instance. By Lemma 1, $|\text{OPT}| = |\text{OPT}'| + 1$. Observe that any remaining link ℓ' must have both endpoints in the same C_i (otherwise ℓ and ℓ' would be crossing). Thus by the inductive hypothesis the optimum solution for the problem induced by C_i has size $|V(C_i)| - 1$. It then follows that $|\text{OPT}'| = |V(C_1)| - 1 + |V(C_2)| - 1 = n - 2$. Hence $|\text{OPT}| = n - 1$ as desired.

For the second part of the claim, it is sufficient to show that a minimal solution does not induce a cycle. By contradiction, consider a minimal solution containing a simple cycle L', and consider now a solution where we remove precisely one arbitrary link $\ell = (u, v)$ from L'. Consider any pair of edges e_1, e_2 belonging to the same cycle such that ℓ satisfies the $\{e_1, e_2\}$-cut. Since $L' \setminus \ell$ induces a simple u-v path, then some $\ell' \in L' \setminus \ell$ must satisfy the cut. Thus $L' \setminus \ell$ is a feasible solution, contradicting the minimality of L'.

Now we proceed to prove the approximation guarantee of the algorithm.

Theorem 1 *The CROSSING-FIRST algorithm is a $\frac{5}{3}$-approximation for CycAP.*

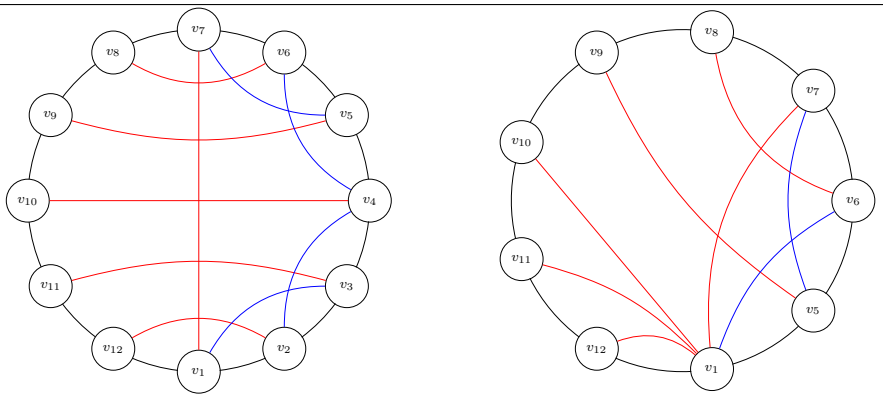

Fig. 1 Left: Instance (G_2, L_2) from the lower bound construction in Lemma 4. Red links define an optimal solution. **Right**: If the algorithm in the first phase picks and contracts the crossing links $\{(v_1, v_3), (v_2, v_4)\}$, this is the obtained CacAP instance.

Proof Let OPT be the optimal solution and APX the computed solution. Let also n'' be the number of nodes remaining at the end of the first stage, and APX$'$ (resp. APX$''$) be the set of links added to the solution during the first (resp. second) stage. Since contracting an external link decreases the number of nodes by at least 2 and contracting any pair of crossing links decreases the number of nodes by at least 3, we have that $|\text{APX}'| \leq \frac{2}{3}(n - n'')$.

By Lemma 3, $|\text{APX}''| = n'' - 1$, and hence $|\text{APX}| \leq \frac{2}{3}(n - n'') + n'' - 1 = \frac{2n+n''-3}{3}$. On the other hand, since any feasible solution must be an edge cover, we have that $|\text{OPT}| \geq n/2$. Observe also that $|\text{OPT}| \geq n'' - 1$ since by Lemma 1 contracting links cannot increase the cost of the optimum solution. Thus $|\text{OPT}| \geq \max\{n/2, n'' - 1\}$. We can conclude that $\frac{|\text{APX}|}{|\text{OPT}|} \leq \frac{(2n+n''-3)/3}{\max\{n/2, n''-1\}} \leq \frac{5}{3}$, being $n'' - 1 = n/2$ the worst case.

We complement this result with an asymptotically matching lower bound.

Lemma 4 *The approximation ratio of the* CROSSING-FIRST *algorithm is not better than* $\frac{5}{3}$.

Proof Consider the following construction: for each $k \geq 2$ consider an instance (G_k, L_k) of CycAP defined by a cycle of $n = 6k$ nodes (assume that the cycle is defined by the order of the nodes $v_1, v_2, \ldots, v_{6k}$) and the following set of links (see Figure 1 (Left)):

- $(v_1, v_{\frac{n}{2}+1}) \in L_k$;
- For each $i = 1, \ldots, \frac{n}{2} - 1$, $(v_{i+1}, v_{n+1-i}) \in L_k$;
- For each $i = 1, \ldots, \frac{n}{6}$, $(v_{3(i-1)+1}, v_{3(i-1)+3}) \in L_k$ and $(v_{3(i-1)+2}, v_{3(i-1)+4}) \in L_k$;

Notice that the first and second set of links define a feasible solution of size $\frac{n}{2}$, hence being optimal: if we remove any two edges of the cycle, then we are either satisfying the corresponding cut via $(v_1, v_{\frac{n}{2}+1})$, or one side of the partition is contained in either $\{v_2, \ldots, v_{\frac{n}{2}}\}$ or in $\{v_{\frac{n}{2}+2}, \ldots, v_n\}$ but the links selected form a matching between those sets.

We will now prove that there exists a sequence of choices performed by our algorithm that outputs a solution of size $\frac{5n}{6} - 1$, which implies that the approximation ratio is at least $\frac{5}{3} - \frac{2}{n}$ and this value approaches $\frac{5}{3}$ as k goes to infinity. Notice first that the pair of links $\{(v_1, v_3), (v_2, v_4)\} \subseteq L_k$ is crossing, and hence the algorithm can include them in the solution in the first round (and finish the round). Furthermore, after these links are contracted no link becomes external as the new cactus instance consists of a cycle of length $n - 3$, and also the links with endpoints v_n, v_{n-1} and v_{n-2} are not part of any pair of crossing links (see Figure 1 (Right)). If we now iteratively pick all the pairs of crossing links $\{(v_{3(i-1)+1}, v_{3(i-1)+3}), (v_{3(i-1)+2}, v_{3(i-1)+4})\} \subseteq L_k$, $i = 2, \ldots, \frac{n}{6}$, after $\frac{n}{6}$ rounds we end up with a cycle of length $\frac{n}{2}$ without crossing links, and the algorithm must now take the remaining $\frac{n}{2} - 1$ links to complete the solution. Thus, the size of the computed solution is $2 \cdot \frac{n}{6} + \frac{n}{2} - 1 = \frac{5}{6}n - 1$, proving the claim.

3.2 A $\left(\frac{3}{2} + \varepsilon\right)$-approximation

The family of instances from Lemma 4 suggests that "short" crossing pairs of links, although being locally profitable, may enforce the algorithm to take expensive decisions in the end. In this section we present a more involved $\left(\frac{3}{2} + \varepsilon\right)$-approximation for CycAP that tries to avoid this kind of situation. Like in the previous algorithm, there is a certain kind of links that we want to iteratively add to our solution in a

first phase, and in this case such links correspond to external links and *long* links, which are defined as follows.

Definition 5 The **length** of an internal link (u, v) is the length of the shortest path between u and v in the corresponding cycle. For a given parameter $0 < \varepsilon < 1$, an internal link is called **long** if its length is at least $\frac{1}{\varepsilon}$, and **short** otherwise.

Our algorithm consists of the following two main phases. In the first phase, we iteratively check if there exists a long (internal) link ℓ. Otherwise, we check if there exists an external link ℓ. In both cases, we add ℓ to the solution under construction and contract it. Observe that contracting links does not create new long links, hence we will first select a set L_{long} of long links, and then a set L_{ext} of external links. After exhausting the previous choices, we move to the second phase. Here we are left with an instance where all links are short and internal, so we can solve independently the sub-instance induced by each cycle. We refer to this algorithm as LONG-FIRST. This second stage can be solved efficiently, due to the lack of long links, by means of the following lemma[4].

Lemma 5 *Given a CycAP instance, there exists an algorithm that returns the optimal solution in time* $poly(n) \cdot 2^{O(h_{\max}^2)}$*, where* $h_{\max}$ *is the maximum length among the links.*

Let L_{short} be the collection of edges obtained in the second stage. The final solution is $L_{\text{long}} \cup L_{\text{ext}} \cup L_{\text{short}}$.

Theorem 2 *The* LONG-FIRST *algorithm is a* $(\frac{3}{2} + \varepsilon)$*-approximation algorithm for CycAP.*

Proof The running time of the algorithm is upper-bounded by $\text{poly}(n)2^{O(1/\varepsilon^2)}$. Consider next the approximation factor. Note first that $|L_{\text{long}}| \leq \varepsilon n$. Indeed, contracting a long link always increases the number of cycles in the cactus by one without decreasing the number of edges, and all these cycles always have size at least $1/\varepsilon$, so there are at most εn of them. Similarly to Theorem 1, we have that $|\text{OPT}| \geq |L_{\text{short}}|$ and $|\text{OPT}| \geq \frac{n}{2}$.

If $|L_{\text{long}}| + |L_{\text{ext}}| + |L_{\text{short}}| \leq \frac{(3+2\varepsilon)n}{4}$ then we already have a $\left(\frac{3}{2} + \varepsilon\right)$-approximation as $|\text{OPT}| \geq \frac{n}{2}$. Otherwise, since the contraction of each external link reduces the number of nodes by at least 2 and the contraction of any other link reduces the number of nodes by at least 1, we have that $|L_{\text{long}}| + 2|L_{\text{ext}}| + |L_{\text{short}}| \leq n$. So $|L_{\text{ext}}| \leq n - \frac{(3+2\varepsilon)n}{4} = \frac{(1-2\varepsilon)n}{4}$ and hence $|L_{\text{ext}}| + |L_{\text{long}}| \leq \frac{n+2\varepsilon n}{4} \leq \left(\frac{1}{2} + \varepsilon\right)|\text{OPT}|$. Since $|\text{OPT}| \geq |L_{\text{short}}|$, we have that in this case the size of the solution is also at most $(\frac{3}{2} + \varepsilon)|\text{OPT}|$, concluding the proof.

Remark 2 By replacing ε with $1/\sqrt{\log n}$ in the above construction, we can obtain a slightly improved approximation factor of $3/2 + o(1)$ which still runs in polynomial time.

It remains to prove Lemma 5. We need some more notation. Recall that a 2-cut $\{e, e'\}$ is satisfied iff there is some link crossing the cut. Given a link $\ell = (u, v)$, we say that the edges of the shortest path between u and v in the cycle are *covered* by ℓ (in case of multiple shortest paths we choose the one going from u to v in counter-clockwise order along the cycle). Given an edge e of the cycle, we define the *cut-neighborhood* of e, namely $\mathcal{N}(e)$, as the $2h_{\max} - 1$ edges that are closest to e, e included. We also define $\mathcal{N}_L(e)$ as the set of links in L covering at least one edge from $\mathcal{N}(e)$.

Notice that in any feasible solution to a CycAP instance, at most one edge of the cycle is not covered: if it is not the case, then the cut defined by two uncovered edges is not satisfied as any link satisfying the cut would cover one of these two edges. We can use this observation to characterize the feasibility of a solution in terms of the cut-neighborhoods.

Lemma 6 *Consider a CycAP instance and let* A *be a set of links such that every edge of the cycle is covered by some link in* A*.* A *is feasible iff for each edge* e*, all the* $\{e, e'\}$*-cuts, where* $e' \in \mathcal{N}(e)$*, are satisfied.*

Proof If A is feasible then the required properties are clearly satisfied since every cut is satisfied. On the other hand, suppose that A satisfies that every edge is covered by some link in A and the $\{e, e'\}$-cuts are satisfied for every edge e and $e' \in \mathcal{N}(e)$. Consider a pair of edges $\{e, e'\}$ such that $e' \notin \mathcal{N}(e)$. By definition of $\mathcal{N}(e)$ there is no link in A covering both edges at the same time, and as e is covered by some link, this link satisfies the $\{e, e'\}$-cut. This implies that A is feasible as every cut is satisfied.

This lemma is useful as it implies that, given an edge e and a set of links S, we can optimally complete S in order to satisfy every $\{e, e'\}$-cut in time $2^{O(h_{\max}^2)}$ just by guessing the subset of links from $\mathcal{N}_L(e)$ that must be added, which are $O(h_{\max}^2)$ only. Now we proceed to present the proof.

(Proof of Lemma 5) Let us assume that we deal with instances of CycAP such that there exists an

[4] This lemma implies that CycAP is FPT with parameter $h_{\max}$.

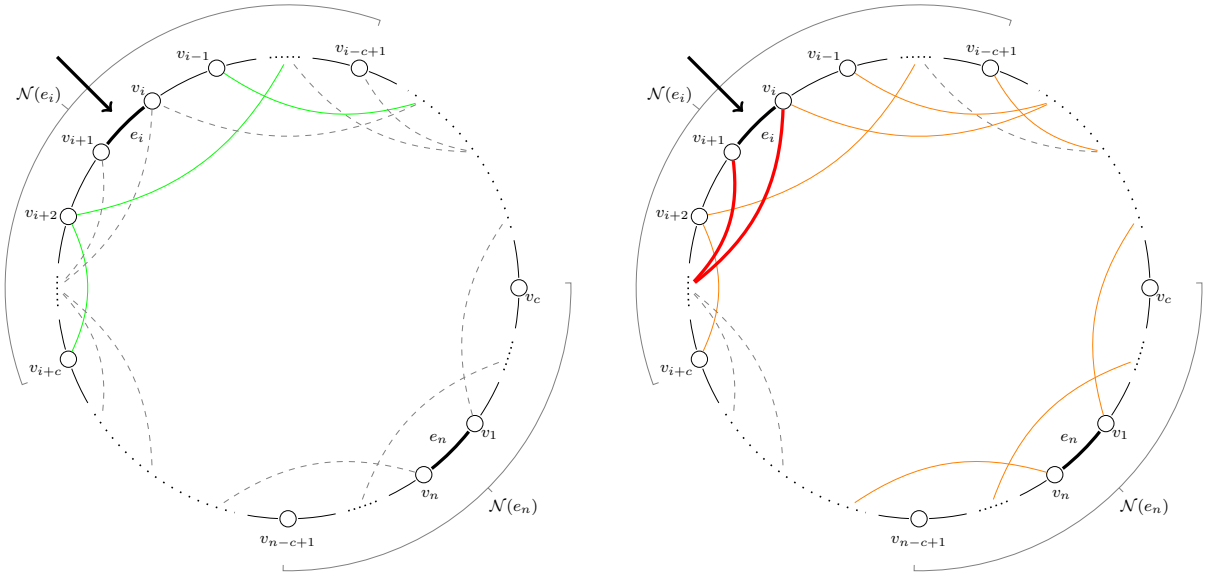

Fig. 2 Depiction of an iteration of the DP from Lemma 5, where we are currently at edge e_i. **Left**: Green links correspond to S and at this point we must decide which extra links to add to S in order to satisfy the edges $e_1, \ldots, e_i$. **Right**: This computation is done by looking at a proper previous cell in the table (orange links) which contains S and satisfies $e_1, \ldots, e_{i-1}$, and then add the extra required links A^* (red links) in order to satisfy e_i too.

optimal solution where every edge is covered by some link. If it is not the case, as there may be only one uncovered edge, we can guess this edge and contract it; this leads to an equivalent instance of the problem where we can require that the optimum solution covers all the edges. We say that an edge e is *satisfied* by a set of links A if it is covered by some link in A and furthermore every $\{e, e'\}$-cut is satisfied by A. In particular A is a feasible solution for the problem iff it satisfies all the edges.

We next design a dynamic programming algorithm to compute a minimum cardinality feasible solution. Let us name the nodes $v_1, v_2, ..., v_n$ in counter-clockwise order starting from some arbitrary node v_1, and let the edges be $e_i = (v_i, v_{i+1})$ for each $i = 1, \ldots, n$ (assuming $v_{n+1} = v_1$).

For each edge e_i and $S \subseteq \mathcal{N}_L(e_i)$, we define a cell $T[i][S]$ which will correspond to a set S' of links of smallest cardinality such that for each $j \in \{1, \ldots, i\}$, e_j is satisfied by S', subject to $S \subseteq S'$. It is then sufficient to return $T[n][\emptyset]$.

We initialize the table by computing $T[1][S]$ for each set $S \subseteq \mathcal{N}_L(e_1)$, which can be done by guessing how to complete S in order to satisfy e_1 with links from $\mathcal{N}_L(e_1)$. Then, for each $i \geq 2$ and $S \subseteq \mathcal{N}_L(e_i)$, in order to fill the cell $T[i][S]$, we consider all the possible subsets $A \subseteq \mathcal{N}_L(e_i)$ such that $S(A) := T[i-1][(S \cup A) \cap \mathcal{N}_L(e_{i-1})] \cup (S \cup A)$ satisfies e_i. Among them we select a set A^* that minimizes $|S(A)|$, and we set $T[i][S] = S(A^*)$ (see Figure 2 for a sketch).

The correctness of the computation follows by a simple induction on i. The table can be filled in total time $\text{poly}(n) \cdot 2^{O(h_{\max}^2)}$, plus an extra factor n from the initial guessing of an uncovered edge (that is contracted).

We complement Theorem 2 with an asymptotically matching lower bound.

Lemma 7 *The approximation ratio of the* LONG-FIRST *algorithm is at least* $\frac{3}{2}$.

Proof Consider the following construction: for each $k > \frac{1}{2\varepsilon}$ consider an instance (G_k, L_k) of CycAP defined by a cycle of $n = 4k$ nodes (assume that the cycle is defined by the order of the nodes $v_1, v_2, \ldots, v_{4k}$) and the following set of links (see Figure 3 (Left)):

- For each $i = 1, \ldots, \frac{n}{2} - 1$, $(v_{i+1}, v_{n+1-i}) \in L_k$;
- $(v_1, v_{\frac{n}{2}+1}) \in L_k$;
- For each $i = 1, \ldots, \frac{n}{4} - 1$, $(v_{i+1}, v_{\frac{n}{2}+1-i}) \in L_k$.

As argued in Lemma 4, the first and second set of links define an optimal solution of size $\frac{n}{2}$. We will now prove that there exists a sequence of choices performed by our algorithm that outputs a solution of size $\frac{3n}{4} - 1$, which implies that the approximation ratio is at least $\frac{3}{2} - \frac{2}{n}$ and this value approaches $\frac{3}{2}$ as k goes to infinity. Notice first that the link $(v_{\frac{n}{4}+1}, v_{\frac{3n}{4}+1}) \in L_k$ has length $2k > \frac{1}{\varepsilon}$ and hence it is long

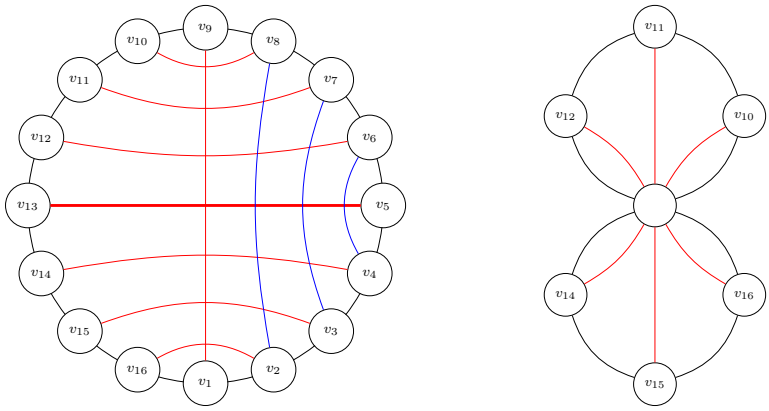

Fig. 3 **Left**: Instance (G_4, L_4) from the lower bound construction in Lemma 7. An optimal solution is defined by red links. **Right**: If the algorithm picks first the thick red link (which is long) and then the links which become external (blue links and (v_1, v_9)) we obtain this subinstance without crossing pairs of links.

so the first stage of the algorithm can include it in the solution. After doing that, the second and third set of links become external and thus the algorithm will include them in the solution. Once all these links are included and contracted, we get a cactus consisting of two cycles of $\frac{n}{4}$ nodes each and without crossing links (see Figure 3 (Right)). Hence, the algorithm must pick all the remaining links to complete the solution. The size then of this solution is $\frac{n}{4} + 1 + 2\left(\frac{n}{4} - 1\right) = \frac{3n}{4} - 1$.

4 LP Relaxations for CycAP

We start by lower-bounding the integrality gap of the standard cut LP for CycAP.

Lemma 8 *The standard cut LP for CycAP has integrality gap at least* 2.

Proof Consider a cycle of size k and, for each edge, a parallel link. The optimum integral solution has size $k - 1$, while setting each variable to $\frac{1}{2}$ gives a feasible fractional solution of cost $\frac{k}{2}$.

This shows that the standard cut LP is not strong enough even for instances without crossing nor long links, cases that we can handle optimally via combinatorial algorithms. We next present a stronger LP that exploits a more general set of constraints.

Let $(G = (V, E), L)$ be a CycAP instance and $S \subseteq E$. We define the *S-reduced* instance (G_S, L_S) as follows: We contract the edges of $E \setminus S$, obtaining a cycle with $|S|$ edges which defines G_S, and the set of links L_S will correspond to the images of L. Notice that there is a one-to-one relation between L_S and the links in L which satisfy some cut defined by a pair of edges from S. We denote by OPT_S the optimal solution for the instance (G_S, L_S)[5]. The following lemma characterizes the feasibility of a solution.

Lemma 9 *Given an instance (G, L) of CycAP, a solution $A \subseteq L$ is feasible iff for every $S \subseteq E$ it holds that $|A \cap L_S| \geq |\mathrm{OPT}_S|$.*

Proof Suppose that there exists $S \subseteq E$ such that $|A \cap L_S| < |\mathrm{OPT}_S|$. This means that $A \cap L_S$ is not a feasible solution for (G_S, L_S) and hence there exist two edges $e_i, e_j \in S$ such that no link in $A \cap L_S$ satisfies the $\{e_i, e_j\}$-cut. As the remaining links in $A \setminus L_S$ also do not satisfy the cut by definition, this cut remains unsatisfied in the original instance, implying that A is not feasible.

On the other hand, suppose that A satisfies the claimed property for every set S. If we consider just sets S consisting of two edges this is exactly the characterization of feasibility shown in Remark 1, implying that A is feasible.

This implies that we can add the constraint $\sum_{\ell \in L_S} x_\ell \geq |\mathrm{OPT}_S|$ for $S \subseteq E$. Unfortunately there is an exponential number of such constraints and most of them require to compute $|\mathrm{OPT}_S|$ for large instances. However, if we restrict ourselves to sets of edges having constant size, we get an LP formulation with polynomially many constraints that can be written in polynomial time. We call this LP the k-edge-cut LP for a given constant $k \in \mathbb{N}$, which is similar in spirit to the *bundle-LP* for TAP introduced by Adjiashvili [1].

[5] For $|S| \leq 1$, we simply set $\mathrm{OPT}_S = \emptyset$.

$$\begin{aligned} \min \sum_{\ell \in L} x_\ell & \qquad (k\text{-edge-cut LP}) \\ s.t. \sum_{\ell \in L_S} x_\ell \geq |\mathrm{OPT}_S| & \qquad \forall S \subseteq E, |S| \leq k \\ 0 \leq x_\ell \leq 1 & \qquad \forall \ell \in L \end{aligned}$$

Notice that for $k = 2$ this is exactly the standard cut LP. Now we will prove some properties of this relaxation and bound its integrality gap.

Lemma 10 *Given $\varepsilon > 0$, for $k = \frac{1}{\varepsilon^2}$ the k-edge-cut LP restricted to instances with links of length at most $\frac{1}{\varepsilon}$ has integrality gap at most $(1 + 2\varepsilon)$.*

Proof We will assume w.l.o.g. that the set of links L contains every possible link of length 1. If it is not the case, let us include them obtaining a new set of links $L' \supseteq L$. The optimal LP value can only decrease while the size of the optimal solution cannot decrease, implying that the integrality gap can only increase due to this operation. To see this last fact, assume by contradiction that there exists a solution OPT' for the new instance having strictly smaller size than OPT. Consider now a solution S consisting of $\mathrm{OPT}' \cap L$ plus a minimal set of links from L that makes S feasible (this is possible since the instance admits a feasible solution). If we in parallel iteratively contract the common links in S and OPT' we arrive to the same CacAP instance, but now the remaining links from OPT' have length 1 and the contraction of each of them reduces the number of nodes in the instance by exactly one node while the contraction of the remaining links in S reduces the number of nodes by at least 1. Thus $|S| \leq |\mathrm{OPT}'|$ which is not possible since $S \subseteq L$.

Let $X = (x_\ell)_{\ell \in L}$ be an optimal solution for the k-edge-cut LP. We will construct an integral feasible solution of size at most $(1 + \varepsilon) \sum_{\ell \in L} x_\ell$. To do so, we will partition the cycle into disjoint intervals as follows: We will first define an interval of size k (which we will call a *long* interval) and then an interval of size $\frac{1}{\varepsilon}$ (which we will call a *short* interval), and then continue with this procedure until it is not possible to continue. If in the end there are at most $\frac{1}{\varepsilon}$ edges we define a last short interval consisting of these remaining edges, otherwise we define a short interval consisting of the last $\frac{1}{\varepsilon}$ edges and a long interval consisting of the remaining edges (which will have size at most k). The number of short intervals is upper bounded by $1 + \left\lfloor \frac{n}{1/\varepsilon^2 + 1/\varepsilon} \right\rfloor \leq 1 + \frac{\varepsilon^2 n}{1+\varepsilon} \leq \varepsilon^2 n$ assuming w.l.o.g. that n is lower bounded by a large enough constant.

Notice that $\sum_{\ell \in L} x_\ell \geq n/2$ by a simple averaging argument over the n constraints corresponding to all the pairs of consecutive edges: every link appears in exactly two such constraints and the right-hand side of each constraint is 1. Since the total number of links of length 1 having both endpoints in a short interval is at most $\varepsilon^2 n \cdot \frac{1}{\varepsilon} = \varepsilon n \leq 2\varepsilon \sum_{\ell \in L} x_\ell$, we can add them to our solution at a negligible cost.

Consider now the set of long intervals $S_1, S_2, \ldots, S_T$. Notice that no link has endpoints in different long intervals, and hence the LP constraints associated to such intervals do not share common variables. This implies that $\sum_{\ell \in L} x_\ell \geq \sum_{i=1}^{T} |\mathrm{OPT}_{S_i}|$. Our feasible solution will consist of all the links of length 1 with both endpoints in a short interval plus the optimal solutions OPT_{S_i} for each long interval S_i. As argued before, the size of this solution is at most $(1 + 2\varepsilon) \sum_{\ell \in L} x_\ell$ and the feasibility of the solution follows since every $\{e, e'\}$-cut where e is in a short interval is satisfied by a link of length 1, while the remaining cuts are satisfied by the links computed optimally.

Lemma 11 *Given $\varepsilon > 0$, for $k = \frac{1}{\varepsilon^2}$ the k-edge-cut formulation has integrality gap at most $(1 + 4\varepsilon)$ restricted to instances without crossing pairs of links.*

Proof Let $X = (x_\ell)_{\ell \in L}$ be an optimal solution for the k-edge-cut LP. Suppose that the instance does not contain links of length at least $\frac{1}{\varepsilon}$, then we can conclude the claim thanks to Lemma 10. Otherwise, we will pick any link of length at least $\frac{1}{\varepsilon}$ and contract it, obtaining a CacAP instance consisting of two cycles without external links (as there are no crossing links), both of size at least $\frac{1}{\varepsilon}$. If any cycle still contains some long link, we iterate this procedure. Let L_{long} be the set of long links we picked during this procedure and $C_1, C_2, \ldots, C_T$ be the set of cycles at the end. By the same argument as in Theorem 2, we have that $|L_{\text{long}}| \leq \varepsilon n \leq 2\varepsilon \sum_{\ell \in L} x_\ell$.

Applying Lemma 10 to each cycle, we obtain a feasible solution of size at most $(1+2\varepsilon) \sum_{i=1}^{T} \mathrm{OPT}_{\mathrm{LP}_i} + |L_{\text{long}}|$, where LP_i is the k-edge-cut LP defined by each cycle C_i and its internal links. As there are no

external links, the sum of the previous LP solutions is the optimal solution for the following LP:

$$\begin{aligned} \min \quad & \sum_{\ell \in L \setminus L_{\text{long}}} x_\ell \\ s.t. \quad & \sum_{\ell \in L_S} x_\ell \geq |\text{OPT}_S| \ \forall i \in \{1, \ldots, T\}, \forall S \subseteq E(C_i), |S| \leq \tfrac{1}{\varepsilon^2} \\ & 0 \leq x_\ell \leq 1 \qquad \forall \ell \in L \setminus L_{\text{long}} \end{aligned}$$

The set of constraints of this LP is a subset of the constraints of the original LP as links in L_{long} do not appear in these constraints and the set of variables is a subset of the original one. Thus we have $\sum_{i=1}^{T} \text{OPT}_{\text{LP}_i} \leq \sum_{\ell \in L} x_\ell$, and then we can conclude that the constructed solution has size at most $(1 + 4\varepsilon) \sum_{\ell \in L} x_\ell$.

Following the proof of Theorem 2 plus the previous results we can get the following bound on the integrality gap for general instances of CycAP.

Corollary 1 *For any $\varepsilon > 0$, the integrality gap of the k-edge-cut LP for $k = \frac{1}{\varepsilon^2}$ is at most $\frac{3}{2} + O(\varepsilon)$.*

Proof Let $X = (x_\ell)_{\ell \in L}$ be an optimal solution for the k-edge cut LP and consider the output of the $\left(\frac{3}{2} + \varepsilon\right)$-approximation from Section 3.2 decomposed into L_{long}, L_{ext} and L_{short} as in the proof of Theorem 2. As argued before, we know that $\sum_{\ell \in L} x_\ell \geq \frac{n}{2}$ and analogously to the proof of Lemma 11 we have that $|L_{\text{short}}| \leq (1 + 2\varepsilon) \sum_{\ell \in L} x_\ell$. Hence essentially the same analysis as in Theorem 2 provides the same bound of $3/2 + O(\varepsilon)$ up to an extra $(1 + \varepsilon)$ factor.

5 Hardness of Approximation

In the following two sections we discuss the hardness of approximation for WCycAP and CycAP, respectively.

5.1 Hardness of Approximation for WCycAP

We now provide an approximation preserving reduction from WCacAP to WCycAP. Note that finding a better-than-2-approximation for WCacAP is at least as hard as finding such an approximation for WTAP, a big open problem in the area. Therefore our reduction shows that achieving a similar result for WCycAP is a very hard task as well.

Theorem 3 *Given an instance A of WCacAP, it is possible to construct in polynomial time an instance B of WCycAP (whose only possibly new weight value is 0) such that any feasible solution to A can be mapped in polynomial time into a feasible solution to B of the same cost and vice versa.*

To prove Theorem 3 we make use of the "inverse" of the contraction of a link, which we call an *expansion*: Consider a WCacAP instance with a node v with degree greater than 2. An expansion of v will consist of taking two cycles containing v and replacing them by the Eulerian tour that traverses them starting from v. Every node appears exactly once except for v which appears twice, for which we create two copies: v_1 the starting node and v_2 the intermediate one. The links originally incident to v are replaced by links of the same cost incident to v_1, and we also add a link of cost zero between v_1 and v_2 (see Figure 4 for an example). The two main properties of this procedure are that: (1) the contraction of a link created by an expansion brings back the graph to the original state and (2) v is replaced by v_1 and v_2, which have degree $\deg(v) - 2$ and 2, respectively.

(Proof of Theorem 3) At high level our proof works as follows. We will build in polynomial time a chain of WCacAP instances $(G_1, L_1), \ldots, (G_k, L_k)$, with the following properties: (i) (G_1, L_1) is the input instance and G_k is a cycle; (ii) (G_{i+1}, L_{i+1}), $i = 1, \ldots, k-1$, is obtained from (G_i, L_i) via precisely one expansion (so G_{i+1} contains precisely one cycle less than G_i, and precisely one new link ℓ_{i+1} of cost zero); (iii) a feasible solution to (G_i, L_i), $i = 1, \ldots, k-1$, can be turned in polynomial time into a feasible solution to (G_{i+1}, L_{i+1}) of the same cost and vice versa. The above properties together trivially imply the claim.

Given (G_i, L_i), we proceed as follows. Consider any node $v \in G_i$ of degree at least 4, and let C_1 and C_2 be any two cycles incident to v (that must exist). We apply an expansion to node v w.r.t. C_1 and C_2,

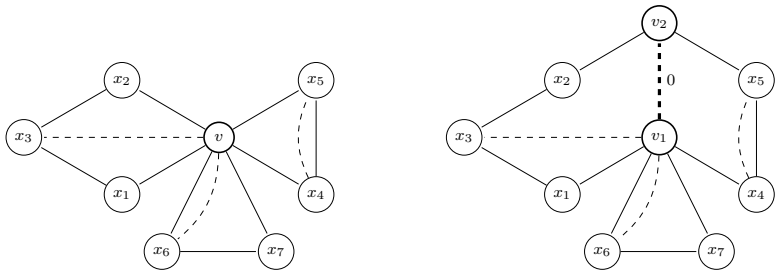

Fig. 4 Depiction of an expansion applied to node v in the left graph considering the cycles to the left and right of v. Dashed edges correspond to links and the highlighted link in the right graph corresponds to the extra link of cost zero added by the expansion.

hence creating a new link ℓ_{i+1} of cost zero. Properties (i) and (ii) follow immediately by construction. Observe that (G_i, L_i) can be obtained from (G_{i+1}, L_{i+1}) by contracting ℓ_{i+1}. Hence property (iii) follows directly from Lemma 1. In more detail, given a feasible solution A_{i+1} to (G_i, L_i), we first add ℓ_{i+1} to A_{i+1} (that keeps the solution feasible, and does not change its cost). By Lemma 1, $A_i := A_{i+1} \setminus \ell_{i+1}$ is a feasible solution to (G_i, L_i) of the same cost. Vice versa, given a feasible solution A_i to (G_i, L_i), $A_{i+1} := A_i \cup \ell_{i+1}$ is a feasible solution to (G_{i+1}, L_{i+1}) of the same cost.

5.2 Hardness of Approximation for CycAP

In this section we prove that CycAP is APX-hard via a reduction from a restricted case of 3-Dimensional Matching (3DM). In the general version of 3DM we are given three disjoint sets W, X and Y having equal cardinality p and a set of m hyperedges $H \subseteq W \times X \times Y$. A (3D) matching is a subset $M \subseteq H$ such that each element of $W \cup X \cup Y$ belongs to at most one hyperedge in M, and this matching is *perfect* if $|M| = p$. Notice that in a perfect matching M each element of $W \cup X \cup Y$ belongs to precisely one hyperedge. The goal is to determine whether a perfect matching exists. We will consider the special case 3DM-K, $K \in \mathbb{N}$, where we add the constraint that each element from $W \cup X \cup Y$ appears in at most K hyperedges. The following result will help us to conclude our final claim.

Theorem 4 (Petrank [29]) *For some fixed $\varepsilon_0 > 0$, it is NP-hard to distinguish whether an instance of 3DM-5 with $|W| = |X| = |Y| = q$ has a perfect matching (of size q) or every matching has size at most $(1 - \varepsilon_0)q$.*

The proof of the following theorem is similar in spirit to the proof of NP-hardness for WTAP due to Frederickson and JáJá [14] and the extension presented by Kortsarz et al. [22]. In the first reduction the authors start from an instance A of 3DM with $3p$ nodes and m hyperedges, and build a WTAP instance B such that: A has a feasible solution (with p hyperedges) iff B has a feasible solution with $p + m$ links. By duplicating the edges in B, one obtains a CacAP instance C with exactly the same property over some cactus G. Our main idea is to turn C into an instance D of CycAP by constructing an Euler tour G' out of G and shortcutting some nodes. However, we need to carefully choose the ordering in the Euler tour in order to preserve a mapping between the feasible solutions of C and D. By following the refined approach from the second reduction, we will show that it is hard to distinguish solutions with a gap depending on the maximum degree in the instance and then use Theorem 4 to conclude the following result.

Theorem 5 *For some fixed $\varepsilon > 0$, it is NP-hard to approximate CycAP within a factor $1 + \varepsilon$.*

Construction of the instance: Let $H \subseteq W \times X \times Y$ be an instance of 3DM with $|H| = m$, $W = \{w_1, \ldots, w_p\}, X = \{x_1, \ldots, x_p\}$ and $Y = \{y_1, \ldots, y_p\}$. We will define an instance $(G = (V, E), L)$ of CycAP where nodes are placed on the cycle in the order as they appear below in counterclockwise direction (see Figure 5 for a depiction of the instance):

- For each node $x_i \in X$ we define a node x_i;
- For each node $y_i \in Y$ we define a node y_i;
- Let $H(w_i)$ denote the hyperedges in H containing $w_i \in W$. For each hyperedge $h \in H(w_i)$ we define two nodes, namely h_X and h_Y (*hyperedge nodes*). These nodes are added to the cycle in the following order. For each $i \in \{1, \ldots, p\}$, we add first nodes h_X corresponding to hyperedges in $H(w_i)$ (in some arbitrary order) and then the corresponding nodes h_Y respecting the *same order used before*. We will denote the first set of nodes by $H_X(w_i)$, and the second set by $H_Y(w_i)$.

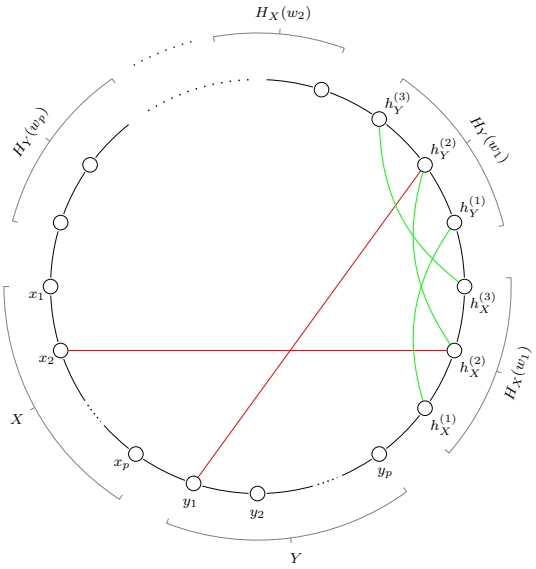

Fig. 5 Example of the construction in Theorem 5. Red links correspond to hyperedge $h^{(2)} = (w_1, x_2, y_1)$ and green links join the copies of the hyperedges.

The set of links L is defined as follows:

- For each hyperedge $h \in H$ we add the link (h_X, h_Y);
- For each hyperedge $h \in H$ and a node $x \in X$, we add the link (h_X, x) iff $x \in h$;
- For each hyperedge $h \in H$ and a node $y \in Y$, we add the link (h_Y, y) iff $y \in h$.

Lemma 12 *If the 3DM instance H contains a 3D matching M with p hyperedges then the CycAP instance (G, L) constructed as above admits a solution A of size $p + m$.*

Proof Suppose that H contains a 3D matching M of size p. We build a solution A to (G, L) as follows: For each hyperedge $h = (w, x, y) \in M$ we add to A the links (h_X, x) and (h_Y, y). Also, for each hyperedge $h \in H \backslash M$ we add the link (h_X, h_Y) to A. Observe that the total number of links in A is $2p+(m-p) = p+m$.

Let us show that A is a feasible solution. By Remark 1, it is sufficient to consider any pair of edges $\{e_1, e_2\}$, and show that there exists some link $\ell \in A$ satisfying the corresponding $\{e_1, e_2\}$-cut. Let us denote by S' and S'' the sets of nodes induced by the cut. Let H_X (resp., H_Y) be the collection of nodes of type h_X (resp., h_Y). We make the following case distinction: Suppose first that e_1 is incident to two nodes in X or $e_1 = (x_p, y_1)$ (the case e_1 being incident to two nodes in Y is symmetric). We distinguish the following 3 subcases depending on e_2:

1. Suppose e_2 is incident to at least one node in $X \cup Y$. Then one of the sets in the cut, say S'', contains all the hyperedge nodes while S' contains at least one node in $z \in X \cup Y$. By construction each node in X (resp. Y) is adjacent to some node in H_X (resp., in H_Y). Thus this cut is satisfied.
2. Suppose e_2 is not incident to any node in $H_Y(w_p)$. Then one of the sets in the cut, say S', contains completely Y, while S'' contains $H_Y(w_p)$. By construction, for $h = (w_p, x, y) \in M$, $\ell = (h_Y, y) \in A$, hence this cut is satisfied.
3. Suppose e_2 is incident to some node in $H_Y(w_p)$. Then one of the sets in the cut, say S'', contains H_X while the other set contains at least one node x from X. Again by construction, for $h = (w, x, y) \in M$, $\ell = (h_X, x) \in A$. Hence this cut is satisfied.

Suppose on the other hand that e_1 and e_2 are incident to at least one hyperedge node. Notice that one of the sets in the cut, say S', contains $X \cup Y$. We distinguish the following 2 subcases:

1. If S'' contains entirely $H_X(w)$ or $H_Y(w)$ for some $w \in W$, then for $h = (w, x, y) \in M$, (h_X, x) or (h_Y, y) is contained in A and the cut is satisfied.
2. In the remaining case we prove that the following claim holds: There exists an hyperedge h such that $h_X \in S'$ and $h_Y \in S''$.
   Suppose by contradiction that for every hyperedge h both h_X and h_Y belong to the same side of the considered cut. Let w_i be such that either $H_X(w_i)$ or $H_Y(w_i)$ has non-empty intersection with both sides of the cut. Note that such w_i must exist, otherwise there would exist w_j such that S'' contains

either $H_X(w_j)$ or $H_Y(w_j)$ completely which was already covered by the previous case. Assume w.l.o.g. that $H_X(w_i) = \{h_X^1, \ldots, h_X^q\}$ is the considered set with elements sorted in counterclockwise direction. Since h_X^1 and h_Y^1 are on the same side of the partition and $H_X(w_i)$ is not fully contained in any side of the partition, it must hold that one set of the partition is properly contained in $H_X(w_i)$. Then any node inside that set has its copy on the other side of the partition. This is in contradiction with the assumption.
Let h be an hyperedge as in the previous claim. We are either adding to the solution the link that joins both copies of h (i.e. the case when $h \notin M$) and the proof is finished, or we are adding the two links joining the two copies of h to elements in X and Y (i.e. the case when $h \in M$). Since $X \cup Y$ is contained in S' and both copies of h are in different sides of the partition, one of the links satisfies the cut.

For any $z \in W \cup X \cup Y$ in a 3DM instance, let $\deg(z)$ be the number of hyperedges in H containing z. Let also Δ denote the maximum degree of the instance, i.e., $\Delta = \max_{z \in W \cup X \cup Y} \deg(z)$. By following an analogous approach to the one from Kortsarz et al. [22], we can prove that even instances with a gap can be mapped.

Lemma 13 *If the CycAP instance (G, L) constructed as above admits a solution A with $|A| \leq (1+\varepsilon)(p+m)$, then the 3DM instance H contains a 3D matching M with $|M| \geq p - (2 + 10\Delta)(p+m)\varepsilon$.*

Proof Let A be a feasible solution to (G, L) with $|A| \leq (1+\varepsilon)(p+m)$. Note that G contains $2(p+m)$ nodes and the links must form an edge cover (otherwise the resulting graph would not be 3-edge-connected). Call a node *permissible* if it is adjacent to exactly one link in A and *impermissible* otherwise. Let V_{perm} and V_{imperm} be the set of permissible and impermissible nodes respectively. We will first prove that the number of impermissible nodes is upper bounded by $2\varepsilon(p+m)$. In fact, if $\deg_A(v)$ denotes the number of links in A incident to v, we have that

$$2|A| = \sum_{v \in V} |\deg_A(v)| = \sum_{v \in V_{\text{perm}}} \deg_A(v) + \sum_{v \in V_{\text{imperm}}} \deg_A(v) \geq |V_{\text{perm}}| + 2|V_{\text{imperm}}|$$

where the last inequality comes from the fact that impermissible nodes are adjacent to at least two links. Since $|A| \leq (1+\varepsilon)(p+m)$, and $|V_{\text{perm}}| + |V_{\text{imperm}}| = 2(p+m)$, we can conclude the claim.

We will now compute a set M' which is almost a matching. We initialize $M' = \emptyset$ and then, iteratively for $j = 1, \ldots, p$, we try to add an hyperedge to M' as follows: if x_j is permissible, then it is adjacent to one node $h_x^{(j)} \in H_X$ (let us assume $h_x^{(j)} \in H_X(w_i)$); if both $h_x^{(j)}$ and its copy $h_y^{(j)} \in H_Y(w_i)$ are permissible, then $h_y^{(j)}$ is adjacent to one node y_k. If y_k is permissible, then we add (w_i, x_j, y_k) to M'. Notice that hyperedges added by this procedure are indeed in H by construction. Our claim is that $|M'| \geq p - 2\Delta(p+m)\varepsilon$. Actually, if x_j, $h_x^{(j)}$ or $h_y^{(j)}$ are impermissible, then only one iteration fails (the one indexed by j). If y_k is impermissible then it can cause at most Δ iterations to fail, since it can be connected to at most Δ nodes in H_Y. If we denote by n_y the number of impermissible nodes y_k involved in the procedure, then the number of iterations that fail is at most $(2\varepsilon(p+m) - n_y) + n_y\Delta$. Since $n_y \leq 2\varepsilon(p+m)$ (the total number of impermissible nodes), the number of iterations that fail is at most $2\Delta(p+m)\varepsilon$, proving the claim.

By construction, hyperedges in M' have different elements from X and Y but elements from W might be repeated. Thus, for every w_i belonging to more than one hyperedge in M', we remove from M' all but one of such hyperedges, obtaining M'' which is now a matching. Let $\mu = p - |M''|$ be the number of vertices w_i not appearing in any hyperedge of M' (equivalently of M''). Since $|M'| - |M''| \leq p - |M''| = \mu$, we can find a lower bound on the size of M'' by bounding above μ. We indeed claim that $\mu \leq (2 + 8\Delta)(p+m)\varepsilon$.

Let L' be the links in L of the form $(x_j, h_X^{(j)})$ and $(y_k, h_Y^{(j)})$ where $h_X^{(j)}$ corresponds to a hyperedge $(w_i, x_j, y_k) \in M'$ and $h_Y^{(j)}$ corresponds to its copy. We have that $|L'| = 2|M'| \geq 2p - 4\Delta(p+m)\varepsilon$, hence

$$|A \setminus L'| \leq (1+\varepsilon)(p+m) - 2p + 4\Delta(p+m)\varepsilon = m - p + (1 + 4\Delta)(p+m)\varepsilon.$$

Consider on the other hand the μ nodes w_i which are not intersected by hyperedges in M'. Since A is a feasible solution, for each such w_i there must be a link in A connecting $H_X(w_i) \cup H_Y(w_i)$ and $X \cup Y$, because otherwise we could disconnect $H_X(w_i) \cup H_Y(w_i)$ from the rest of the graph by removing the two edges in the boundary of $H_X(w_i) \cup H_Y(w_i)$, contradicting the feasibility of A. Notice that these μ links are part of $A \setminus L'$. Furthermore, since A is an edge cover, the remaining $2m - 2p - \mu$ nodes in $H_X \cup H_Y$ untouched by L' plus the μ aforementioned links must be incident to some link in A, implying that

$$|A \setminus L'| \geq \mu + \frac{2m - 2p - \mu}{2} = m - p + \frac{\mu}{2}.$$

Combining both inequalities we get that $\mu \leq (2+8\Delta)(p+m)\varepsilon$, and hence we conclude that the size of M'' is at least

$$|M'| - \mu \geq p - 2\Delta(p+m)\varepsilon - (2+8\Delta)(p+m)\varepsilon = p - (2+10\Delta)(p+m)\varepsilon,$$

completing the proof.

We can now use Lemmas 12 and 13 together with Theorem 4 to conclude the proof of Theorem 5. Notice that in 3DM-5, since $\Delta = 5$, we have that $m = |H| \leq 5|W| = 5p$.

(Proof of Theorem 5) We will show that our reduction presented above is gap-preserving. Specifically, we will show that if H is an instance of 3DM-5 and (G, L) is the corresponding CycAP instance, then

1. If H admits a matching of size p, then (G, L) admits a feasible solution of size $p + m$;
2. If H does not admit a matching of size at least $p(1-\varepsilon_0)$, then (G, L) does not admit a feasible solution of size at most $(p+m)(1+\frac{\varepsilon_0}{312})$.

The first statement follows directly from Lemma 12, while the second is the contrapositive of Lemma 13 when setting $\varepsilon = \frac{\varepsilon_0}{312}$, as in this case we have that $p - (2+10\Delta)(5p+p)\varepsilon = p(1-312\varepsilon) = p(1-\varepsilon_0)$.

References

1. Adjiashvili, D.: Beating Approximation Factor Two for Weighted Tree Augmentation with Bounded Costs. SODA 2017, 2384–2399 (2017)
2. Basavaraju, M., Fomin, F.V., Golovach, P., Misra, P., Ramanujan, M.S., Saurabh, S.: Parameterized Algorithms to Preserve Connectivity. ICALP 2014, 800–811 (2014)
3. Cheriyan, J., Gao, Z.: Approximating (Unweighted) Tree Augmentation via Lift-and-Project, Part I: Stemless TAP. Algorithmica 80, 530–559 (2018)
4. Cheriyan, J., Gao, Z.: Approximating (Unweighted) Tree Augmentation via Lift-and-Project, Part II. Algorithmica 80, 608–651 (2018)
5. Cheriyan, J., Jordán, T., Ravi, R.: On 2-Coverings and 2-Packings of Laminar Families. ESA 1999, 510–520 (1999)
6. Cheriyan, J., Karloff, H., Khandekar, R., Könemann, J.: On the Integrality Ratio for Tree Augmentation. Oper. Res. Lett. 36, 399–401 (2008)
7. Cohen, N., Nutov, Z.: A (1 + ln2)-Approximation Algorithm for Minimum-Cost 2-Edge-Connectivity Augmentation of Trees with Constant Radius. APPROX/RANDOM 2011, 147–157 (2011)
8. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, Third Edition. (2009)
9. Dinitz, E., Karzanov, A., Lomonosov, M.: On the Structure of the System of Minimum Edge Cuts of a Graph. Studies in Discrete Optimization, 290–306 (1976)
10. Even, G., Feldman, J., Kortsarz, G. Nutov, Z.: A 1.8 Approximation Algorithm for Augmenting Edge-Connectivity of a Graph from 1 to 2. ACM Trans. Algorithms 5, 21:1–21:17 (2009)
11. Fiorini, S., Groß, M., Könemann, J., Sanità, L.: Approximating Weighted Tree Augmentation via Chvátal-Gomory Cuts. SODA 2018, 817–831 (2018)
12. Frank, A.: Augmenting Graphs to Meet Edge-Connectivity Requirements. SIAM J. Discrete Math. 5, 25–53 (1992)
13. Frank, A.: Connections in Combinatorial Optimization. Number 38 in Oxford lecture series in mathematics and its applications (2011)
14. Frederickson, G.N., JáJá, J.: Approximation Algorithms for Several Graph Augmentation Problems. SIAM J. Comput. 10, 270–283 (1981)
15. Goemans, M.X., Goldberg, A.V., Plotkin, S., Shmoys, D.B., Tardos, É., Williamson, D.P.: Improved Approximation Algorithms for Network Design Problems. SODA 1994, 223–232 (1994)
16. Grandoni, F., Kalaitzis, C., Zenklusen, R.: Improved Approximation for Tree Augmentation: Saving by Rewiring. STOC 2018, 632–645 (2018)
17. Hsu, T.: On Four-Connecting a Triconnected Graph. J. Algorithms 35, 202–234 (2000)
18. Jain, K.: A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem. Combinatorica 21, 39–60 (2001)
19. Jordan, T.: On the Optimal Vertex-Connectivity Augmentation. J. Combin. Theory 35, 202–234 (1995)
20. Khuller, S., Thurimella, R.: Approximation Algorithms for Graph Augmentation. J. Algorithms 14, 214–225 (1993)
21. Knuth, D.E.: Postscript About NP-hard Problems. SIGACT News 6, 15–16 (1974)
22. Kortsarz, G., Krauthgamer R., Lee, J.R.: Hardness of Approximation for Vertex-Connectivity Network Design Problems. SIAM J. Comput. 33, 704–720 (2004)
23. Kortsarz, G., Nutov, Z.: A Simplified 1.5-Approximation Algorithm for Augmenting Edge-Connectivity of a Graph from 1 to 2. ACM Trans. Algorithms 12, 23:1–23:20 (2015)
24. Kortsarz, G., Nutov, Z.: Approximating Minimum Cost Connectivity Problems. Handbook on Approximation Algorithms and Metaheuristics 35, 202–234 (2007)
25. Kortsarz, G., Nutov, Z.: LP-Relaxations for Tree Augmentation. APPROX/RANDOM 2016, 13:1–13:16 (2016)
26. Marx, D., Végh, L.: Fixed-Parameter Algorithms for Minimum-Cost Edge-Connectivity Augmentation. ACM Trans. Algorithms 11(4), 27:1–27:24 (2015)
27. Nagamochi, H.: An Approximation for Finding a Smallest 2-Edge-Connected Subgraph Containing a Specified Spanning Tree. Discrete Appl. Math. 126, 83–113 (2003)
28. Nutov, Z.: On the Tree Augmentation Problem. ESA 2017, 61:1–61:14 (2017)
29. Petrank, E.: The Hardness of Approximation: Gap Location. Computational Complexity 4, 133–157 (1994)
30. Shmoys, D.B.: The Design of Approximation Algorithms. Cambridge (2011)
31. Végh, L.: Augmenting Undirected Node-Connectivity by One. STOC 2010, 563–572 (2010)
32. Watanabe, T., Nakamura, A.: Edge-Connectivity Augmentation Problems. J. Comput. Syst. Sci. 35, 96-144 (1987)
33. Williamson, D.P., Shmoys, D.B.: Algorithms and Complexity. Amsterdam: Elsevier (2011)