


# Approximating Long Cycle Above Dirac's Guarantee

Fedor V. Fomin 

Department of Informatics, University of Bergen, Norway

Petr A. Golovach 

Department of Informatics, University of Bergen, Norway

Danil Sagunov 

St. Petersburg Department of V.A. Steklov Institute of Mathematics, Russia

Kirill Simonov 

Hasso Plattner Institute, Universität Potsdam, Germany

---

## Abstract

---

Parameterization above (or below) a guarantee is a successful concept in parameterized algorithms. The idea is that many computational problems admit “natural” guarantees bringing to algorithmic questions whether a better solution (above the guarantee) could be obtained efficiently. For example, for every boolean CNF formula on  $m$  clauses, there is an assignment that satisfies at least  $m/2$  clauses. How difficult is it to decide whether there is an assignment satisfying more than  $m/2 + k$  clauses? Or, if an  $n$ -vertex graph has a perfect matching, then its vertex cover is at least  $n/2$ . Is there a vertex cover of size at least  $n/2 + k$  for some  $k \geq 1$  and how difficult is it to find such a vertex cover?

The above guarantee paradigm has led to several exciting discoveries in the areas of parameterized algorithms and kernelization. We argue that this paradigm could bring forth fresh perspectives on well-studied problems in approximation algorithms. Our example is the longest cycle problem. One of the oldest results in extremal combinatorics is the celebrated Dirac's theorem from 1952. Dirac's theorem provides the following guarantee on the length of the longest cycle: for every 2-connected  $n$ -vertex graph  $G$  with minimum degree  $\delta(G) \leq n/2$ , the length of the longest cycle  $L$  is at least  $2\delta(G)$ . Thus the “essential” part of finding the longest cycle is in approximating the “offset”  $k = L - 2\delta(G)$ . The main result of this paper is the above-guarantee approximation theorem for  $k$ . Informally, the theorem says that approximating the offset  $k$  is not harder than approximating the total length  $L$  of a cycle. In other words, for any (reasonably well-behaved) function  $f$ , a polynomial time algorithm constructing a cycle of length  $f(L)$  in an undirected graph with a cycle of length  $L$ , yields a polynomial time algorithm constructing a cycle of length  $2\delta(G) + \Omega(f(k))$ .

**2012 ACM Subject Classification** Mathematics of computing  $\rightarrow$  Graph algorithms; Theory of computation  $\rightarrow$  Approximation algorithms analysis; Theory of computation  $\rightarrow$  Parameterized complexity and exact algorithms

**Keywords and phrases** Longest path, longest cycle, approximation algorithms, above guarantee parameterization, minimum degree, Dirac theorem

**Digital Object Identifier** 10.4230/LIPIcs.ICALP.2023.60

**Category** Track A: Algorithms, Complexity and Games

**Related Version** *Full Version*: <http://arxiv.org/abs/2305.02011> [26]

**Funding** Supported by the Research Council of Norway via the project BWCA (grant no. 314528) and DFG Research Group ADYN via grant DFG 411362735.



© Fedor V. Fomin, Petr A. Golovach, Danil Sagunov, and Kirill Simonov;  
licensed under Creative Commons License CC-BY 4.0

50th International Colloquium on Automata, Languages, and Programming (ICALP 2023).

Editors: Kousha Etessami, Uriel Feige, and Gabriele Puppis; Article No. 60; pp. 60:1–60:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



## 1 Introduction

One of the concepts that had a strong impact on the development of parameterized algorithms and kernelization is the idea of the above guarantee parameterization. Above guarantee parameterization grounds on the following observation: *the natural parameterization of a maximization/minimization problem by the solution size is not satisfactory if there is a lower bound for the solution size that is sufficiently large* [23]. To make this discussion concrete, consider the example of the classical NP-complete problem MAX CUT. Observe that in any graph with  $m$  edges there is always a cut containing at least  $m/2$  edges. (Actually, slightly better bounds are known in the literature [18, 10].) Thus MAX CUT is trivially fixed-parameter tractable (FPT) parameterized by the size of the max-cut. Indeed, the following simple algorithm shows that the problem is FPT: If  $k \leq m/2$ , then return **yes**; else  $m \leq 2k$  and any brute-force algorithm will do the job. However, the question about MAX CUT becomes much more meaningful and interesting, when one seeks a cut above the “guaranteed” lower bound  $m/2$ .

The above guarantee approach was introduced by Mahajan and Raman [46] and it was successfully applied in the study of several fundamental problems in parameterized complexity and kernelization. For illustrative examples, we refer to [1, 4, 14, 23, 25, 33, 34, 35, 37, 38, 45], see also the recent survey of Gutin and Mnich [36]. Quite surprisingly, the theory of the above (or below) guarantee *approximation* remains unexplored. (Notable exceptions are the works of Mishra et al. [47] on approximating the minimum vertex cover beyond the size of a maximum matching and of Bollobás and Scott on approximating max-cut beyond the  $m/2 + \sqrt{m}/8$  bound [10].)

In this paper, we bring the philosophy of the above guarantee parameterization into the realm of approximation algorithms. In particular,

The goal of this paper is to study the approximability of the classical problems of finding a longest cycle and a longest  $(s, t)$ -path in a graph from the viewpoint of the above guarantee parameterization.

**Our results.** Approximating the length of a longest cycle in a graph enjoys a lengthy and rich history [6, 8, 21, 20, 29, 30, 49]. There are several fundamental results in extremal combinatorics providing lower bounds on the length of a longest cycle in a graph. The oldest of these bounds is given by Dirac's Theorem from 1952 [17]. Dirac's Theorem states that a 2-connected graph  $G$  with the minimum vertex degree  $\delta(G)$  contains a cycle of length  $L \geq \min\{2\delta(G), |V(G)|\}$ . Since every longest cycle in a graph  $G$  with  $\delta(G) < \frac{1}{2}|V(G)|$  (otherwise,  $G$  is Hamiltonian and a longest cycle can be found in polynomial time) always has a “complementary” part of length  $2\delta(G)$ , the essence of the problem is in computing the “offset”  $k = L - 2\delta(G)$ . Informally, the first main finding of our paper is that Dirac's theorem is well-compatible with approximation. We prove that approximating the offset  $k$  is essentially not more difficult than approximating the length  $L$ .

More precisely. Recall that  $f$  is subadditive if for all  $x, y$  it holds that  $f(x+y) \leq f(x) + f(y)$ . Our main result is the following theorem.

► **Theorem 1.** *Let  $f : \mathbb{R}_+ \rightarrow \mathbb{R}_+$  be a non-decreasing subadditive function and suppose that we are given a polynomial-time algorithm finding a cycle of length at least  $f(L)$  in graphs with the longest cycle length  $L$ . Then there exists a polynomial time algorithm that finds a cycle of length at least  $2\delta(G) + \Omega(f(L - 2\delta(G)))$  in a 2-connected graph  $G$  with  $\delta(G) \leq \frac{1}{2}|V(G)|$  and the longest cycle length  $L$ .*

The 2-connectivity condition is important. As was noted in [24], deciding whether a connected graph  $G$  contains a cycle of length at least  $2\delta(G)$  is NP-complete. Theorem 1 trivially extends to approximating the longest path problem above  $2\delta(G)$ . For the longest path, the requirement on 2-connectivity of a graph can be relaxed to connectivity. This can be done by a standard reduction of adding an apex vertex  $v$  to the connected graph  $G$ , see e.g. [24]. The minimum vertex degree in the new graph  $G + v$ , which is 2-connected, is equal to  $\delta(G) + 1$ , and  $G$  has a path of length at least  $L$  if and only if  $G + v$  has a cycle of length at least  $L + 2$ . Thus approximation of the longest cycle (by making use of Theorem 1) in  $G + v$ , is also the approximation of the longest path in  $G$ .

**Related work.** The first approximation algorithms for longest paths and cycles followed the development of exact parameterized algorithms. Monien [48] and Bodlaender [9] gave parameterized algorithms computing a path of length  $L$  in times  $\mathcal{O}(L!2^L n)$  and  $\mathcal{O}(L!nm)$  respectively. These algorithms imply also approximation algorithms constructing in polynomial time a path of length  $\Omega(\log L / \log \log L)$ , where  $L$  is the longest path length in graph  $G$ . In their celebrated work on color coding, Alon, Yuster, and Zwick [2] obtained an algorithm that in time  $\mathcal{O}(5.44^L n)$  finds a path/cycle of length  $L$ . The algorithm of Alon et al. implies constructing in polynomial time a path of length  $\Omega(\log L)$ . A significant amount of the consecutive work targets to improve the base of the exponent  $c^L$  in the running times of the parameterized algorithms for longest paths and cycles [43, 50, 28, 5, 7]. The surveys [27, 44], and [15, Chapter 10] provide an overview of ideas and methods in this research direction. The exponential dependence in  $L$  in the running times of these algorithms is asymptotically optimal: An algorithm finding a path (or cycle) of length  $L$  in time  $2^{o(L)} n^{\mathcal{O}(1)}$  would fail the Exponential Time Hypothesis (ETH) of Impagliazzo, Paturi, and Zane [39]. Thus none of the further improvements in the running times of parameterized algorithms for longest cycle or path, would lead to a better than  $\Omega(\log L)$  approximation bound.

Björklund and Husfeldt [6] made the first step “beyond color-coding” in approximating the longest path. They gave a polynomial-time algorithm that finds a path of length  $\Omega(\log L / \log \log L)^2$  in a graph with the longest path length  $L$ . Gabow in [30] enhanced and extended this result to approximating the longest cycle. His algorithm computes a cycle of length  $2^{\Omega(\sqrt{\log L / \log \log L})}$  in a graph with a cycle of length  $L$ . Gabow and Nie [32] observed that a refinement of Gabow’s algorithm leads to a polynomial-time algorithm constructing cycles of length  $2^{\Omega(\sqrt{\log L})}$ . This is better than  $(\log(L))^{\mathcal{O}(1)}$  but worse than  $L^\epsilon$ . Pipelining the algorithm of Gabow and Nie with Theorem 1 yields a polynomial time algorithm constructing in a 2-connected graph  $G$  a cycle of length  $2\delta(G) + \Omega(c^{\sqrt{\log k}})$ . For graphs of bounded vertex degrees, better approximation algorithms are known [13, 21].

The gap between the upper and lower bounds for the longest path approximation is still big. Karger, Motwani, and Ramkumar [41] proved that the longest path problem does not belong to APX unless  $P = NP$ . They also show that for any  $\epsilon > 0$ , it cannot be approximated within  $2^{\log^{1-\epsilon} n}$  unless  $NP \subseteq \text{DTIME}(2^{\mathcal{O}(\log^{1/\epsilon} n)})$ . Bazgan, Santha, and Tuza [3] extended these lower bounds to cubic Hamiltonian graphs. For directed graphs the gap between the upper and lower bounds is narrower [8, 31].

Our approximation algorithms are inspired by the recent work Fomin, Golovach, Sagunov, and Simonov [24] on the parameterized complexity of the longest cycle beyond Dirac’s bound. Fomin et al. were interested in computing the “offset” beyond  $2\delta(G)$  exactly. Their parameterized algorithm decides whether  $G$  contains a cycle of length at least  $2\delta(G) + k$  in time  $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ , and thus in polynomial time computes a cycle of length  $2\delta(G) + \Omega(\log k)$ . However, the tools developed in [24] are not sufficient to go beyond  $\Omega(\log k)$ -bound on the

offset. The main combinatorial tools from [24] are Erdős-Gallai decomposition and Dirac decomposition of graphs. For the needs of approximation, we have to develop novel (“nested”) variants or prove additional structural properties of these decompositions.

Dirac’s theorem is one of the central pillars of Extremal Graph Theory. The excellent surveys [12] and [11] provide an introduction to this fundamental subarea of graph theory. Besides [24], the algorithmic applications of Dirac’s theorem from the perspective of parameterized complexity were studied by Jansen, Kozma, and Nederlof in [40].

**Paper structure.** Section 2 provides an overview of the techniques employed to achieve our results. Then, Section 3 introduces notations and lists auxiliary results. Section 4 guides through the proof of the approximation result for  $(s, t)$ -paths, which is the key ingredient required for Theorem 1. Finally, we conclude with a summary and some open questions in Section 5. Note that the proofs of technical statements are omitted from this extended abstract due to space constraints. Detailed proofs of all results can be found in the full version of the paper [26].

## 2 Overview of the proofs

In this section, we provide a high-level strategy of the proof of Theorem 1, as well as key technical ideas needed along the way. The central concept of our work is an approximation algorithm for the LONGEST CYCLE problem. Formally, such an algorithm should run in polynomial time for a given graph  $G$  and should output a cycle of length at least  $f(L)$ , where  $L$  is the length of the longest cycle in  $G$ . The function  $f$  here is the *approximation guarantee* of the algorithm. In our work, we allow it to be an arbitrary non-decreasing function  $f : \mathbb{R}_+ \rightarrow \mathbb{R}_+$  that is also subadditive (i.e.,  $f(x) + f(y) \geq f(x + y)$  for arbitrary  $x, y$ ). We also note that an  $f(L)$ -approximation algorithm for LONGEST CYCLE immediately gives a  $\frac{1}{2}f(2L)$ -approximation algorithm for LONG  $(s, t)$ -PATH in 2-connected graphs (by Menger’s theorem, see Lemma 4 for details).

Our two main contributions assume that we are given such an  $f$ -approximation algorithm as a black box. In fact, we only require to run this algorithm on an arbitrary graph as an oracle and receive its output. We do not need to modify or know the algorithm routine.

While the basis of our algorithm comes from the structural results of Fomin et al. [24], in the first part of this section we do not provide the details on how it is used.

The first of our contributions is a polynomial-time algorithm that finds a long  $(s, t)$ -path in a given 2-connected graph  $G$  with two vertices  $s, t \in V(G)$ . The longest  $(s, t)$ -path in  $G$  always has length  $\delta(G - \{s, t\}) + k$  for  $k \geq 0$  by Erdős-Gallai theorem, and the goal of the algorithm is to find an  $(s, t)$ -path of length at least  $\delta(G - \{s, t\}) + \Omega(f(k))$  in  $G$ . To find such a path, this algorithm first recursively decomposes the graph  $G$  in a specific technical way. As a result, it outputs several triples  $(H_i, s_i, t_i)$  in polynomial time, where  $H_i$  is a 2-connected minor of  $G$  and  $s_i, t_i \in V(H_i)$ . For each triple, the algorithm runs the black box to find a  $f$ -approximation of the longest  $(s_i, t_i)$ -path in  $H_i$ . In the second round, our algorithm cleverly uses constructed approximations to construct a path of length at least  $\delta(G - \{s, t\}) + \Omega(f(k))$  in the initial graph  $G$ . This is summarized as the following theorem.

► **Theorem 2.** *Let  $f : \mathbb{R}_+ \rightarrow \mathbb{R}_+$  be a non-decreasing subadditive function and suppose that we are given a polynomial-time algorithm computing an  $(s, t)$ -path of length at least  $f(L)$  in graphs with given two vertices  $s$  and  $t$  having the longest  $(s, t)$ -path of length  $L$ . Then there is a polynomial-time algorithm that outputs an  $(s, t)$ -path of length at least  $\delta(G - \{s, t\}) + \Omega(f(L - \delta(G - \{s, t\})))$  in a 2-connected graph  $G$  with two given vertices  $s$  and  $t$  having the longest  $(s, t)$ -path length  $L$ .*

The second (and main) contribution of this paper is the polynomial-time algorithm that approximates the longest cycle in a given 2-connected graph  $G$  such that  $2\delta(G) \leq |V(G)|$ . It employs the black-box  $f$ -approximation algorithm for LONGEST CYCLE to find a cycle of length  $2\delta(G) + \Omega(f(k))$ , where  $2\delta(G) + k$  is the length of the longest cycle in  $G$ . By Dirac's theorem applied to  $G$ ,  $k$  is always at least 0.

To achieve that, our algorithm first tries to decompose the graph  $G$ . However, in contrast to the first contributed algorithm, here the decomposition process is much simpler. In fact, the decomposition routine is never applied recursively, as the decomposition itself needs not to be used: its existence is sufficient to apply another, simple, procedure.

Similarly to the first contribution, the algorithm then outputs a series of triples  $(H_i, s_i, t_i)$ , where  $H_i$  is a 2-connected minor of  $G$  and  $s_i, t_i \in V(H_i)$ . The difference here is that for each triple the algorithm runs not the initial black-box  $f$ -approximation algorithm, but the algorithm of the first contribution, i.e. the algorithm of Theorem 2. Thus, the output of each run is an  $(s_i, t_i)$ -path of length  $\delta(H_i - \{s_i, t_i\}) + \Omega(f(k_i))$  in  $H_i$ , where  $\delta(H_i - \{s_i, t_i\}) + k_i$  is the length of the longest  $(s_i, t_i)$ -path in  $H_i$ .

Finally, from each approximation, our algorithm constructs a cycle of length at least  $2\delta(G) + \Omega(f(k_i))$ . It is guaranteed that  $k_i = \Omega(k)$  for at least one  $i$ , so the longest of all constructed cycles is of length at least  $2\delta(G) + \Omega(f(k))$ . The following theorem is in order.

► **Theorem 1.** *Let  $f : \mathbb{R}_+ \rightarrow \mathbb{R}_+$  be a non-decreasing subadditive function and suppose that we are given a polynomial-time algorithm finding a cycle of length at least  $f(L)$  in graphs with the longest cycle length  $L$ . Then there exists a polynomial time algorithm that finds a cycle of length at least  $2\delta(G) + \Omega(f(L - 2\delta(G)))$  in a 2-connected graph  $G$  with  $\delta(G) \leq \frac{1}{2}|V(G)|$  and the longest cycle length  $L$ .*

One may note that Theorem 2 actually follows from Theorem 1 (again, by Menger's theorem, see Lemma 4). However, as described above, the algorithm in Theorem 1 employs the algorithm of Theorem 2, so we have to prove the latter before the former.

In the remaining part of this section, we provide more detailed proof overviews of both theorems, in particular, we explain how the algorithms employ the structural results of [24]. In both proofs, we complement these results by showing useful properties of specific graph decompositions. For clarity, we start with Theorem 1, as its proof is less involved.

## 2.1 Approximating long cycles

The basis of our algorithm is the structural result due to Fomin et al. [24]. In that work, the authors show the following: There is an algorithm that, given a cycle in a 2-connected graph, either finds a longer cycle or finds that  $G$  is of a "particular structure". This algorithm can be applied to any cycle of length less than  $(2 + \sigma_1) \cdot \delta(G)$  (to be specific, we use  $\sigma_1 = \frac{1}{24}$ , see [26] for details).

To see how this structural result is important, recall that we aim to find a cycle of length at least  $2\delta(G) + \Omega(f(k))$  in a 2-connected graph  $G$  with the longest cycle length  $2\delta(G) + k$ . Our algorithm simply starts with some cycle in  $G$  and applies the result of [24] to enlarge it exhaustively. It stops when either a cycle is of length at least  $(2 + \sigma_1) \cdot \delta(G)$ , or the particular structure of  $G$  is found.

The crucial observation here is that if a long cycle is found, we can trivially find a good approximation. If  $\sigma_1 \cdot \delta(G)$  is, e.g., less than  $\sigma_1/10 \cdot f(k)$ , then  $10\delta(G) < f(k)$ . If we just apply the blackbox  $f$ -approximation algorithm for the LONGEST CYCLE problem, we get a cycle of length at least  $f(2\delta(G) + k) \geq f(k) \geq 2\delta(G) + 4/5 \cdot f(k)$ . Hence, by taking the longest of the cycles of length  $(2 + \sigma_1) \cdot \delta(G)$  and of length  $f(2\delta(G) + k)$  we always achieve a good approximation guarantee on  $k$ .

The most important part of the algorithm is employed when the “particular structure” outcome is received from the structural lemma applied on  $G$  and the current cycle  $C$ . Here we need to be specific about this structure, and the outcome can be of two types. The first outcome is a bounded vertex cover of the graph. This vertex cover is of size at most  $\delta(G) + 2(k' + 1)$ , where  $k' \geq 0$  is such that  $|V(C)| = 2\delta(G) + k'$ . Such vertex cover is a guarantee that  $C$  is not much shorter than the longest cycle in  $G$ : the length of the longest cycle is bounded by twice the vertex cover size, so  $k \leq 4(k' + 1)$ . Hence,  $k' = \Omega(k)$  and  $C$  is a sufficient approximation.

The second, and last, structural outcome is the Dirac decomposition, defined in [24]. Basically, this decomposition is obtained by finding a small separator of  $G$  (that consists of just two subpaths  $P_1, P_2$  of the cycle  $C$ ), and the parts of this decomposition are the connected components of  $G$  after the separation. The main result on Dirac decomposition proved in [24] is that there always exists a longest cycle that contains an edge in at least one of these parts.

While the definition and properties of Dirac decomposition may seem quite involved, our algorithm does not even require the Dirac decomposition of  $G$  to be found. In fact, we show a new nice property of Dirac decomposition. It guarantees that if a Dirac decomposition for  $G$  exists, then there also exists a 2-vertex separator  $\{u, v\}$  of  $G$  that also divides the longest cycle in  $G$  into almost even parts. Our contribution is formulated in the following lemma.

► **Lemma 3.** *Let  $G$  be a 2-connected graph and  $P_1, P_2$  induce a Dirac decomposition for a cycle  $C$  of length at most  $2\delta(G) + \kappa$  in  $G$  such that  $2\kappa \leq \delta(G)$ . If there exists a cycle of length at least  $2\delta(G) + k$  in  $G$ , then there exist  $u, v \in V(G)$  such that*

- $G - \{u, v\}$  is not connected, and
- there is an  $(u, v)$ -path of length at least  $\delta(G) + (k - 2)/4$  in  $G$ .

Our algorithm employs Lemma 3 in the following way. Since there are  $\mathcal{O}(|V(G)|^2)$  vertex pairs in  $G$ , our algorithm iterates over all vertex pairs. If a pair  $u, v$  separates the graph into at least two parts, then our algorithm finds a long  $(u, v)$ -path that contains vertices in only one of the parts. Formally, it iterates over all connected components in  $G - \{u, v\}$ . For a fixed connected component  $H$ , our algorithm applies the algorithm of Theorem 2 to the graph  $G[V(H) \cup \{u, v\}] + uv$  (the edge  $uv$  is added to ensure 2-connectivity), to find an approximation of the longest  $(u, v)$ -path. By Lemma 3, if  $u, v$  is the required separating pair, then for at least one  $H$  the length of the found  $(u, v)$ -path should be  $\delta(G) + \Omega(k)$ . And if such a path is found, a sufficiently long  $(u, v)$ -path outside  $H$  in  $G$  is guaranteed by Erdős-Gallai theorem. Together, these two paths form the required cycle of length  $2\delta(G) + \Omega(k)$ .

With that, the proof overview of Theorem 1 is finished. The formal proof is presented in [26].

## 2.2 Approximating long $(s, t)$ -paths

While the algorithm of Theorem 1 does not use the underlying Dirac decomposition explicitly, in the case of finding  $(s, t)$ -paths (and to prove Theorem 2), we require deeper usage of the obtained graph decomposition. While the Dirac decomposition of Fomin et al. was originally used in [24] to find long cycles above  $2\delta(G)$ , for finding  $(s, t)$ -paths above  $\delta(G - \{s, t\})$  the authors introduced the Erdős-Gallai decomposition.

In the formal proof of Theorem 2 in Section 4, we give a complete definition of Erdős-Gallai decomposition. In this overview, we aim to avoid the most technical details in order to provide an intuition of the structure of the decomposition and how our algorithm employs it.



Similarly to Dirac decomposition, the Erdős-Gallai decomposition is obtained through the routine that, given a graph  $G$  and an  $(s, t)$ -path inside it, either enlarges the path or reports that two subpaths  $P_1$  (that starts with  $s$ ) and  $P_2$  (that starts with  $t$ ) of the given path induce (when deleted) an Erdős-Gallai decomposition in  $G$ . This routine can be applied to an  $(s, t)$ -path until it reaches  $(1 + \sigma_2) \cdot \delta(G - \{s, t\})$  in length (specifically,  $\sigma_2 = \frac{1}{4}$ , see Lemma 13; in this overview, we also skip the case of a Hamiltonian  $(s, t)$ -path for brevity). Note that, in contrast to the cycle enlargement routine of the Dirac decomposition, here the bounded vertex cover outcome is not possible. Similarly to the algorithm of the previous subsection, the only non-trivial part of the algorithm is dealing with the Erdős-Gallai decomposition outcome. In the other case, a single run of the black-box  $f$ -approximation algorithm for LONGEST CYCLE provides the desired approximation immediately.

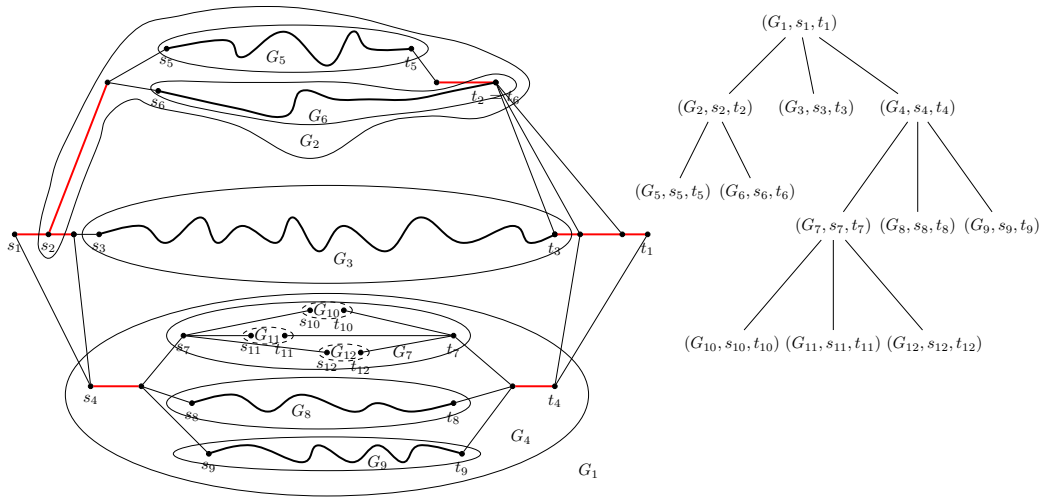
The main property of this decomposition due to [24] is as follows: If an  $(s, t)$ -path of length at least  $\delta(G - \{s, t\}) + k$  exists in  $G$ , then there necessarily exists the path of length at least  $\delta(G - \{s, t\}) + k$  that goes through one of the connected components in the decomposition. Moreover, for each of the connected components  $G_i$  there is exactly one pair of distinct entrypoints  $s_i, t_i$ : if an  $(s, t)$ -path in  $G$  goes through  $G_i$ , it should necessary enter  $G_i$  in  $s_i$  (or  $t_i$ ) once and leave  $G_i$  in  $t_i$  (or  $s_i$ ) exactly once as well.

Additionally to that, we have that the degree of each  $G_i$  is not much different from  $G$ :  $\delta(G_i - \{s_i, t_i\}) \geq \delta(G - \{s, t\}) - 2$  holds true. And this constant difference is always compensated by paths from  $s$  and  $t$  to  $s_i$  and  $t_i$ : if we succeed to find an  $(s_i, t_i)$ -path of length at least  $\delta(G_i - \{s_i, t_i\}) + k_i$  inside  $G_i$ , we can always complete it with *any* pair of disjoint paths from  $\{s, t\}$  to  $\{s_i, t_i\}$  into an  $(s, t)$ -path of length  $\delta(G - \{s, t\}) + k_i$  in  $G$ . Should this pair be longer than the trivial lower bound of 2, it grants the additional length above  $\delta(G - \{s, t\}) + k_i$ .

The previous paragraph suggests the following approach for our approximation algorithm: for each  $G_i, s_i, t_i$ , our algorithm applies itself recursively to find an  $(s_i, t_i)$ -path of length  $\delta(G_i - \{s_i, t_i\}) + \Omega(f(k_i))$ , where  $k_i$  comes from the longest  $(s_i, t_i)$ -path length in  $G_i$ . Since the other part of the additional length comes from two disjoint paths between  $\{s, t\}$ , and  $\{s_i, t_i\}$ , we would like to employ the black-box  $f$ -approximation algorithm to find the  $f$ -approximation of this pair of paths.

Unfortunately, finding such pair of paths reduces only to finding a long cycle through a given pair of vertices (it is enough to glue  $s$  with  $t$  and  $s_i$  with  $t_i$  in  $G$ , and ask to find the long cycle through the resulting pair of vertices). In their work, Fomin et al. have shown that the problem of finding such a cycle of length at least  $k$  can be done in  $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$  time. However, this is of little use to us, as  $k$  is only bounded by  $\mathcal{O}(\delta(G))$ , but we require polynomial time. Simultaneously, we do not know of any way to force the black-box algorithm to find an  $f$ -approximation for a cycle through the given pair of vertices.

These arguments bring us away from the idea of a recursive approximation algorithm. Instead, our approximation algorithm will apply the black-box algorithm to a single “complete-picture” graph that is obtained according to the structure brought by the Erdős-Gallai decomposition. However, the recursion here remains in the sense that we apply the path-enlarging routine to each component of the decomposition. This brings us to the idea of the recursive decomposition, which we define as the *nested Erdős-Gallai decomposition* in Section 4. This decomposition can be seen as a tree, where the root is the initial triple  $(G, s, t)$ , the children of a node represent the triples  $(G_i, s_i, t_i)$  given by the Erdős-Gallai decomposition, and the leaves of this decomposition are the graphs  $G_i$  where sufficient approximations of long  $(s_i, t_i)$ -paths are found (by taking the longest of  $(1 + \sigma_2) \cdot \delta(G - \{s_i, t_i\})$ -long path from the enlarging routine and the approximation obtained from the blackbox algorithm). A schematic picture of this novel decomposition is present in Figure 1.



**Figure 1** A schematic example of a nested Erdős-Gallai decomposition (left) and the corresponding recursion tree (right). Red straight paths inside  $G_i$  denote the pair of paths inducing an Erdős-Gallai decomposition in  $G_i$ . Bold  $(s_i, t_i)$ -paths are sufficient approximations of the longest  $(s_i, t_i)$ -paths in  $G_i$ . Dashed contours correspond to  $G_i$  with constant  $\delta(G_i - \{s_i, t_i\})$ , which is one of a few technical cases in the proof.

In Section 4, we show that a long path found inside a leaf  $(G_i, s_i, t_i)$  of the decomposition can be contracted into a single edge  $s_i t_i$ . Moreover, if  $(G_j, s_j, t_j)$  is a child of a  $(G_i, s_i, t_i)$  in the decomposition, and the longest pair of paths from  $\{s_i, t_i\}$  to  $\{s_j, t_j\}$  is just a pair of edges (so it does not grant any additional length as described before), we contract these edges. The crucial in our proof is the claim that after such a contraction, if an  $(s, t)$ -path of length  $\delta(G - \{s, t\}) + k$  exists in the initial graph, an  $(s, t)$ -path of length at least  $\Omega(k)$  exists in the graph obtained with described contractions. After doing all the contractions, the algorithm applies the black-box algorithm to the transformed graph and finds an  $(s, t)$ -path of length  $f(\Omega(k))$  (which is  $\Omega(f(k))$  by subadditivity) inside it.

The final part of our algorithm (and the proof of Theorem 2) is the routine that *transforms* this  $(s, t)$ -path inside the *contracted* graph  $G$  into a path of length  $\delta(G - \{s, t\}) + \Omega(f(k))$  in the *initial* graph  $G$ . In this part, we prove that it is always possible to transform an  $(s, t)$ -path of length  $r$  in the contracted graph into a path of length  $\Omega(r)$  that goes through at least one edge corresponding to a leaf of the nested Erdős-Gallai decomposition (hence, to a good approximation of  $(s_i, t_i)$ -path inside  $G_i$ ). Finally, we observe that reversing the contractions in  $G$  transforms this path into the required approximation.

This finishes the overview of the proof of Theorem 2. Section 4 outlines the proof in detail, providing the sequence of intermediate technical results leading to the proof of the theorem.

### 3 Preliminaries

In this section, we define the notation used throughout the paper and provide some auxiliary results. We use  $[n]$  to denote the set of positive integers  $\{1, \dots, n\}$ . We remind that a function  $f: D \rightarrow \mathbb{R}$  is *subadditive* if  $f(x + y) \leq f(x) + f(y)$  for all  $x, y \in D \subseteq \mathbb{R}$ . We denote the set of all nonnegative real numbers by  $\mathbb{R}_+$ .



Recall that our main theorems are stated for arbitrary nondecreasing subadditive functions  $f : \mathbb{R}_+ \rightarrow \mathbb{R}$ , such that an algorithm achieving the respective approximation exists. Throughout the proofs we will, additionally assume that  $f(x) \leq x$  for every  $x \in \mathbb{R}_+$ . For any integer  $x \geq 3$ , this is already implied by the statement, since a consistent approximation algorithm cannot output an  $(s, t)$ -path (respectively, cycle) of length greater than  $x$  in a graph where the longest  $(s, t)$ -path (respectively, cycle) has length  $x$ . However, for a general function  $f(\cdot)$  this does not necessarily hold on the whole  $\mathbb{R}_+$ . If this is the case, for clarity of the proofs we redefine  $f(x) := \min\{x, f(x)\}$  for every  $x \in \mathbb{R}_+$ . Clearly,  $f$  remains subadditive and non-decreasing, while also imposing exactly the same guarantee on the approximation algorithm.

**Graphs.** We consider only finite simple undirected graphs and use the standard notation (see, e.g., the book of Diestel [16]). The following useful observation follows immediately from Menger's theorem (see, e.g., [16, 42]).

► **Lemma 4.** *For any 2-connected graph  $G$  with a cycle of length  $L$ , there is a path of length at least  $L/2$  between any pair of vertices in  $G$ . Moreover, given a cycle  $C$  and two distinct vertices  $s$  and  $t$ , an  $(s, t)$ -path of length at least  $|V(C)|/2$  can be constructed in polynomial time.*

We observe that given an approximation algorithm for a longest cycle, we can use it as a black box to approximate a longest path between any two vertices.

► **Lemma 5.** *Let  $\mathcal{A}$  be a polynomial-time algorithm that finds a cycle of length at least  $f(L)$  in a graph with the longest cycle length  $L$ . Then there is a polynomial-time algorithm using  $\mathcal{A}$  as a subroutine that, given a graph  $G$  and two distinct vertices  $s$  and  $t$ , finds an  $(s, t)$ -path of length at least  $\frac{1}{2}f(2L)$ , where  $L$  is the length of a longest  $(s, t)$ -path in  $G$ .*

We will use as a subroutine an algorithm finding two disjoint paths between two pairs of vertices of total length at least  $k$ , where  $k$  is the given parameter. For us, constant values of  $k$  suffice, though in fact there exists an FPT algorithm for this problem parameterized by the total length. It follows as an easy corollary from the following result of [24] about LONG  $(s, t)$ -CYCLE, the problem of finding a cycle of length at least  $k$  through the given two vertices  $s$  and  $t$ .

► **Theorem 6** (Theorem 4 in [24]). *There exists an FPT algorithm for LONG  $(s, t)$ -CYCLE parameterized by  $k$ .*

For completeness, we state the corollary next.

► **Corollary 7.** *There is an FPT algorithm that, given a graph  $G$  with two pairs of vertices  $\{s, t\}$  and  $\{s', t'\}$ , and a parameter  $k$ , finds two disjoint paths between  $\{s, t\}$  and  $\{s', t'\}$  in  $G$  of total length at least  $k$ , or correctly determines that such paths do not exist.*

Finally, it is convenient to use the following corollary, which generalizes the theorem of Erdős and Gallai [19, Theorem 1.16].

► **Corollary 8** (Corollary 3 in [24]). *Let  $G$  be a 2-connected graph and let  $s, t$  be a pair of distinct vertices in  $G$ . For any  $B \subseteq V(G)$  there exists a path of length at least  $\delta(G - B)$  between  $s$  and  $t$  in  $G$ . Moreover, there is a polynomial time algorithm constructing a path of such length.*

## 4 Approximating $(s, t)$ -path

In this section, we outline the proof of Theorem 2, stating that any guarantee for approximating the longest cycle in a 2-connected graph can be transferred to approximating the longest  $(s, t)$ -path above minimum degree. For the convenience of the reader, we recall the precise statement next.

► **Theorem 2.** *Let  $f : \mathbb{R}_+ \rightarrow \mathbb{R}_+$  be a non-decreasing subadditive function and suppose that we are given a polynomial-time algorithm computing an  $(s, t)$ -path of length at least  $f(L)$  in graphs with given two vertices  $s$  and  $t$  having the longest  $(s, t)$ -path of length  $L$ . Then there is a polynomial-time algorithm that outputs an  $(s, t)$ -path of length at least  $\delta(G - \{s, t\}) + \Omega(f(L - \delta(G - \{s, t\})))$  in a 2-connected graph  $G$  with two given vertices  $s$  and  $t$  having the longest  $(s, t)$ -path length  $L$ .*

In order to obtain this result, we first recall the concept of Erdős-Gallai decomposition introduced in [24] together with a few of its helpful properties established there. Then we introduce the recursive generalization of this concept, called nested Erdős-Gallai decomposition, and show how to obtain with its help the compression of the graph such that a long  $(s, t)$ -path in the compressed graph can be lifted to an  $(s, t)$ -path in the original graph with a large offset.

### 4.1 Erdős-Gallai decomposition

This subsection encompasses the properties of an Erdős-Gallai decomposition, defined next. The definition itself and most of the technical results presented here are due to [24]. Some of the results from [24] need to be modified in order to be used for our purposes, we provide the proofs of such results in [26]. Note that the statements in [24] hold in the more general case where there is also a low-degree vertex subset in the graph, here while recalling the results we automatically simplify the statements. Next, we recall the definition of an Erdős-Gallai decomposition.

► **Definition 9** (Erdős-Gallai decomposition and Erdős-Gallai component, Definition 2 in [24]). *Let  $P$  be a path in a 2-connected graph  $G$ . We say that two disjoint paths  $P_1$  and  $P_2$  in  $G$  induce an Erdős-Gallai decomposition for  $P$  in  $G$  if*

- *Path  $P$  is of the form  $P = P_1 P' P_2$ , where the inner path  $P'$  has at least  $\delta(G - \{s, t\})$  edges.*
- *There are at least two connected components in  $G - V(P_1 \cup P_2)$ , and for every connected component  $H$ , it holds that  $|V(H)| \geq 3$  and one of the following is fulfilled:*
  1.  *$H$  is 2-connected and the maximum size of a matching in  $G$  between  $V(H)$  and  $V(P_1)$  is one, and between  $V(H)$  and  $V(P_2)$  is also one;*
  2.  *$H$  is not 2-connected, exactly one vertex of  $P_1$  has neighbors in  $H$ , that is  $|N_G(V(H)) \cap V(P_1)| = 1$ , and no inner vertex from a leaf-block of  $H$  has a neighbor in  $P_2$ ;*
  3. *The same as 2, but with  $P_1$  and  $P_2$  interchanged. That is,  $H$  is not 2-connected,  $|N_G(V(H)) \cap V(P_2)| = 1$ , and no inner vertex from a leaf-block of  $H$  has a neighbor in  $P_1$ .*

*The set of Erdős-Gallai components for an Erdős-Gallai decomposition is defined as follows. First, for each component  $H$  of type 1,  $H$  is an Erdős-Gallai component of the Erdős-Gallai decomposition. Second, for each  $H$  of type 2, or of type 3, all its leaf-blocks are also Erdős-Gallai components of the Erdős-Gallai decomposition.*

As long as an Erdős-Gallai decomposition is available, Erdős-Gallai components allow us to bound the structure of optimal solutions in a number of ways. First, Fomin et al. [24] observe that the longest  $(s, t)$ -path necessarily visits an Erdős-Gallai component.

► **Lemma 10** (Lemma 7 in [24]). *Let  $G$  be a graph and  $P_1, P_2$  induce an Erdős-Gallai decomposition for an  $(s, t)$ -path  $P$  in  $G$ . Then there is a longest  $(s, t)$ -path in  $G$  that enters an Erdős-Gallai component.*

Next, since an Erdős-Gallai component has a very restrictive connection to the rest of the graph, it follows that any  $(s, t)$ -path has only one chance of entering the component.

► **Lemma 11** (Lemma 5 in [24]). *Let  $G$  be a 2-connected graph and  $P$  be an  $(s, t)$ -path in  $G$ . Let paths  $P_1, P_2$  induce an Erdős-Gallai decomposition for  $P$  in  $G$ . Let  $M$  be an Erdős-Gallai component. Then for every  $(s, t)$ -path  $P'$  in  $G$ , if  $P'$  enters  $M$ , then all vertices of  $V(M) \cap V(P')$  appear consecutively in  $P'$ .*

For the purposes of recursion, it is convenient to enclose an Erdős-Gallai component together with some of its immediate connections, so that this slightly larger subgraph behaves exactly like an  $(s, t)$ -path instance. The subgraph  $K$  in the next lemma plays this role.

► **Lemma 12** (Lemma 8 in [24]). *Let paths  $P_1, P_2$  induce an Erdős-Gallai decomposition for an  $(s, t)$ -path  $P$  in graph  $G$ . Let  $M$  be an Erdős-Gallai component in  $G$ . Then there is a polynomial time algorithm that outputs a 2-connected subgraph  $K$  of  $G$  and two vertices  $s', t' \in V(K)$ , such for that every  $(s, t)$ -path  $P'$  in  $G$  that enters  $M$ , the following hold:*

1.  $V(K) = (V(M) \cup \{s', t'\})$ ;
2.  $P'[V(K)]$  is an  $(s', t')$ -subpath of  $P'$  and an  $(s', t')$ -path in  $K$ ;
3.  $\delta(K - \{s', t'\}) \geq \delta(G - \{s, t, s', t'\})$ .

Most importantly, Erdős-Gallai decompositions capture extremal situations, where the current  $(s, t)$ -path cannot be made longer in a “simple” way. The next lemma formalizes that intuition, stating that in polynomial time we can find either a long  $(s, t)$ -path or an Erdős-Gallai decomposition. The lemma is largely an analog of Lemma 4 in [24], however our statement here is slightly modified. Next, we recall the statement from Section 2.

► **Lemma 13.** *Let  $G$  be a 2-connected graph such that  $\delta(G - \{s, t\}) \geq 16$ . There is a polynomial time algorithm that*

- *either outputs an  $(s, t)$ -path  $P$  of length at least  $\min\{\frac{5}{4}\delta(G - \{s, t\}) - 3, |V(G)| - 1\}$ ,*
- *or outputs an  $(s, t)$ -path  $P$  with paths  $P_1, P_2$  that induce an Erdős-Gallai decomposition for  $P$  in  $G$ . Additionally, there is no  $(s, t)$ -path in  $G$  that enters at least two Erdős-Gallai components of this Erdős-Gallai decomposition.*

Finally, to deal with  $(s, t)$ -paths that do not enter any Erdős-Gallai component, one can observe the following. Intuitively, such a path should be far from optimal, as going through an Erdős-Gallai component would immediately give at least  $\delta(G - \{s, t\}) - \mathcal{O}(1)$  additional edges of the path. The final lemma of this subsection establishes how precisely the length of a path avoiding Erdős-Gallai components can be “boosted” in this fashion. To obtain this result, we first need a technical lemma from [24] that yields long paths inside separable components.

► **Lemma 14** (Lemma 6 in [24]). *Let  $H$  be a connected graph with at least one cut-vertex. Let  $I$  be the set of inner vertices of all leaf-blocks of  $H$ . Let  $S \subseteq V(H) \setminus I$  separate at least one vertex in  $V(H) \setminus I$  from  $I$  in  $H$ . For any vertex  $v$  that is not an inner vertex of a leaf-block of  $H$ , there is a cut-vertex  $c$  of a leaf-block of  $H$  and a  $(c, v)$ -path of length at least  $\frac{1}{2}(\delta(H) - |S|)$  in  $H$ . This path can be constructed in polynomial time.*

Now we move to  $(s, t)$ -paths that avoid Erdős-Gallai components. The following Lemma 15 has been already stated in Section 2, here we recall the statement.

► **Lemma 15.** *Let  $P$  be an  $(s, t)$ -path of length at most  $\delta(G - \{s, t\}) + k$  and let two paths  $P_1, P_2$  induce a Erdős-Gallai decomposition for  $P$  in  $G$ . There is a polynomial time algorithm that, given an  $(s, t)$ -path of length at least  $4k + 5$  in  $G$  that does not enter any Erdős-Gallai component, outputs a path of length at least  $\min\{\delta(G - \{s, t\}) + k - 1, \frac{3}{2}\delta(G - \{s, t\}) - \frac{5}{2}k - 1\}$  in  $G$ .*

## 4.2 Proof of Theorem 2

To deal with the recursive structure of the solution, we introduce the following *nested* generalization of an Erdős-Gallai decomposition. Intuitively, it captures how the structural observations of the previous subsection allow us to recursively construct Erdős-Gallai decompositions with the aim of finding a long  $(s, t)$ -path. For an illustration of a nested Erdős-Gallai decomposition, see Figure 1. We recall the formal definition from Section 2.

► **Definition 16** (Nested Erdős-Gallai decomposition). *A sequence of triples  $(G_1, s_1, t_1), (G_2, s_2, t_2), \dots, (G_\ell, s_\ell, t_\ell)$  is called a nested Erdős-Gallai decomposition for  $G$  and two vertices  $s, t \in V(G)$  if*

- $(G_1, s_1, t_1) = (G, s, t)$ ;
- for each  $i \in [\ell]$ , either
  - $\delta(G_i - \{s_i, t_i\}) < 16$ , or
  - Lemma 13 applied to  $G_i, s_i, t_i$  gives a path  $P_i$  of length at least  $\min\{\frac{5}{4}\delta(G_i - \{s_i, t_i\}) - 3, |V(G_i)| - 1\}$  in  $G_i$ , or
  - Lemma 13 applied to  $G_i, s_i, t_i$  gives a path  $P_i$  and two paths  $P_{i,1}, P_{i,2}$  that induce an Erdős-Gallai decomposition for  $P_i$  in  $G_i$ , and for each Erdős-Gallai component  $M$  of this decomposition there is  $j > i$  such that  $(G_j, s_j, t_j)$  is the result of Lemma 12 applied to  $M$  in  $G_i$ . In this case, we say that  $G_i$  is decomposed.
- for each  $i \in \{2, \dots, \ell\}$ , there is  $e(i) < i$  such that  $(G_i, s_i, t_i)$  is a result of Lemma 12 applied to some Erdős-Gallai component of the Erdős-Gallai decomposition of  $G_{e(i)}$  for  $P_{e(i)}$ .

The proof of Theorem 2 is performed in two steps: first, we show how to obtain a nested Erdős-Gallai decomposition for a given graph  $G$ , and then we use the nested Erdős-Gallai decomposition to recursively construct a good approximation to the longest  $(s, t)$ -path. The first part is achieved simply by applying Lemma 13 recursively on each Erdős-Gallai component until components are no longer decomposable. The main hurdle is the second part, on which we focus for the rest of the section. For completeness, first we show that a nested Erdős-Gallai decomposition can always be constructed in polynomial time.

► **Lemma 17.** *There is a polynomial time algorithm that, given a 2-connected graph  $G$  and its two vertices  $s$  and  $t$ , outputs a nested Erdős-Gallai decomposition for  $G, s, t$ .*

Clearly, it follows that the size of a nested Erdős-Gallai decomposition returned by Lemma 17 is also polynomial. Observe also that the construction algorithm invokes Lemma 13 for all sufficiently large  $G_i$ , thus in what follows we assume that the corresponding paths  $P_i$  are already computed.

Now we focus on using a constructed nested Erdős-Gallai decomposition for approximating the longest  $(s, t)$ -path. First of all, we present the algorithm `long_nested_st_path` that, given a nested Erdős-Gallai decomposition of  $G$ , computes a long  $(s, t)$ -path by going over

the decomposition. The pseudocode of `long_nested_st_path` is present in Algorithm 3. Intuitively, first the algorithm computes a compression  $H$  of the graph  $G$  that respects the nested Erdős-Gallai decomposition: components that are not decomposed are replaced by single edges, and edges that are “unavoidable” to visit a component are contracted. The computation of this compression is encapsulated in the `nested_compress` function presented in Algorithm 1. As a subroutine, this function uses the `two_long_disjoint_paths` algorithm given by Corollary 7, that finds two disjoint paths of at least the given length between the given pairs of vertices.

Next, the blackbox approximation algorithm `long_st_path_approx` is used to compute an  $(s, t)$ -path  $Q$  in  $H$ . The function `nested_decompress` reconstructs then this path in the original graph  $G$ , see Algorithm 2 for the pseudocode. Later we argue (Lemma 19) that any  $(s, t)$ -path in  $H$  of length  $r$  yields in this way an  $(s, t)$ -path in  $G$  of length at least  $\delta(G - \{s, t\}) + r/8 - 3$ . Finally, either the length of  $Q$  in  $H$  was large enough and the reconstructed path provides the desired approximation or a long path can be found inside one of the components in a “simple” way, and then connected arbitrarily to  $\{s, t\}$ . Specifically, in this component, it suffices to either take an approximation of the longest path computed by `long_st_path_approx`, or a long Erdős-Gallai path returned by the algorithm from Corollary 8, `long_eg_st_path`. Thus, in the final few lines `long_nested_st_path` checks whether any of these paths is longer than the reconstructed path  $Q$ . The path from inside the component is extended to an  $\{s, t\}$ -path in  $G$  by using the algorithm `two_long_disjoint_paths`, given by Corollary 7, with the parameter 0.

■ **Algorithm 1** The algorithm compressing a given graph  $G$  with a given nested Erdős-Gallai decomposition.

---

```

nested_compress(( $G_1, s_1, t_1$ ), ( $G_2, s_2, t_2$ ), ..., ( $G_\ell, s_\ell, t_\ell$ ))
  Input: a nested Erdős-Gallai decomposition for  $G$ ,  $s$  and  $t$ .
  Output: the compressed graph  $H$ .
1.1  $H \leftarrow G$ ;
1.2 foreach  $i \in \{2, \dots, \ell\}$  do
1.3    $j \leftarrow e(i)$ ;
1.4    $d_i \leftarrow |\{s_j, t_j\} \setminus \{s_i, t_i\}|$ ;
1.5   if two_long_disjoint_paths ( $G_i, \{s_j, t_j\}, \{s_i, t_i\}, d_i + 1$ ) is NO then
1.6     contract all edges of a maximum matching between  $\{s_j, t_j\}$  and  $\{s_i, t_i\}$  in
      $H$ ;
1.7   end
1.8   if  $G_i$  is not decomposed then
1.9     remove all vertices in  $V(G_i) \setminus \{s_i, t_i\}$  from  $H$ ;
1.10    add edge  $s_i t_i$  to  $H$  and mark it with  $G_i$ ;
1.11   end
1.12 end
1.13 return  $H$ ;

```

---

Now, our goal is to show that the path that the `long_nested_st_path` algorithm constructs serves indeed as the desired approximation of the longest  $(s, t)$ -path in  $G$ . For the rest of this section, let  $G_1, \dots, G_\ell$  be the given nested Erdős-Gallai decomposition for  $G$ ,  $s, t$ . An important piece of intuition about nested Erdős-Gallai decomposition is that, as we go deeper into the nested Erdős-Gallai components, the minimum degree of the component  $\delta(G_i \setminus \{s_i, t_i\})$  decreases, but we gain more and more edges that we collect while going

■ **Algorithm 2** The algorithm decompressing a path in  $H$  into a long path in  $G$ .

---

```

nested_decompress(( $G_1, s_1, t_1$ ), ( $G_2, s_2, t_2$ ), ..., ( $G_\ell, s_\ell, t_\ell$ ),  $H, Q$ )
  Input: a nested Erdős-Gallai decomposition for  $G, s$  and  $t$ , the compressed graph
            $H$  and an  $(s, t)$ -path  $Q$  in  $H$  of length  $r$ .
  Output: an  $(s, t)$ -path of length at least  $\delta(G - \{s, t\}) + r/8 - 3$  in  $G$ .
2.1 foreach  $i \in \{2, \dots, \ell\}$  such that  $d_i > 0$  and  $Q$  enters  $G_i$  do
2.2    $j \leftarrow e(i)$ ;
2.3   if an edge between  $\{s_j, t_j\}$  and  $\{s_i, t_i\}$  was contracted in  $H$  then
2.4     replace  $s_i$  and/or  $t_i$  in  $Q$  with the respective contracted edges;
2.5   else
2.6      $S_1, S_2 \leftarrow \text{two\_long\_disjoint\_paths}(G, \{s_j, t_j\}, \{s_i, t_i\}, d_i + 1)$ ;
2.7     replace the two subpaths of  $Q$  going from  $\{s_j, t_j\}$  to  $\{s_i, t_i\}$  with  $S_1$  and  $S_2$ 
           if the length of  $Q$  increases;
2.8   end
2.9 end
2.10  $h \leftarrow$  largest  $h \in [\ell]$  such that  $Q$  enters  $G_h$ ;
2.11 if  $G_h$  is not decomposed then
2.12   replace  $s_h t_h$  in  $Q$  with  $P_h$ ;
2.13 else
2.14    $k' \leftarrow \lfloor (|E(Q) \cap E(G_h)| - 5)/8 \rfloor$ ;
2.15   if  $|E(P_h)| \geq \delta(G_h - \{s_h, t_h\}) + k'$  then
2.16      $R \leftarrow P_h$ ;
2.17   else
2.18      $R \leftarrow$  result of Lemma 15 applied to  $G_h, P_h$  and the  $(s_h, t_h)$ -subpath of  $Q$ ;
2.19   end
2.20   if  $(s_h, t_h)$ -subpath of  $Q$  is shorter than  $R$  then
2.21     replace the  $(s_h, t_h)$ -subpath of  $Q$  with  $R$ ;
2.22   end
2.23 end
2.24 return  $Q$ ;

```

---

from  $\{s, t\}$  to  $\{s_i, t_i\}$ . We introduce values that help us measure this difference between the nested components: for each  $i \in [\ell]$ , denote  $d_i = |\{s_{e(i)}, t_{e(i)}\} \setminus \{s_i, t_i\}|$ . In particular, by Lemma 12 we know that for any  $i \in [\ell]$ ,  $\delta(G_i) \geq \delta(G_{e(i)}) - d_i$ . On the other hand, any pair of disjoint paths that connects  $\{s_{e(i)}, t_{e(i)}\}$  to  $\{s_i, t_i\}$  contains at least  $d_i$  edges. This leads to the following simple observation about extending an  $(s_j, t_j)$ -path in a component  $G_j$  to an  $(s, t)$ -path in  $G$ .

▷ **Claim 18.** For each  $j \in [\ell]$ , let  $G_{j_1}, \dots, G_{j_c}$  be such that  $j_c = j$  and  $j_1 = 1$  and  $e(j_{i+1}) = j_i$  for each  $i \in [c-1]$ . Let  $P$  be an  $(s_j, t_j)$ -path in  $G_j$ . Then  $P$  combined with any pair of disjoint paths connecting  $\{s, t\}$  to  $\{s_j, t_j\}$  yields an  $(s, t)$ -path in  $G$  of length at least  $|E(P)| + \sum_{i \in [c-1]} d_{j_{i+1}}$ .

However, there might also exist longer paths connecting nested components  $G_{e(i)}$  and  $G_i$ . When we construct the compressed graph  $H$  in Algorithm 1, we distinguish between two cases. Either any pair of such paths have the total length  $d_i$ , meaning that the only option is to use the edges of a matching between  $\{s_{e(i)}, t_{e(i)}\}$  and  $\{s_i, t_i\}$ . In that case we simply contract these edges as we know that there is no choice on how to reach  $G_i$  from  $G_{e(i)}$ . Or, there is a pair of disjoint paths of total length at least  $d_i + 1$ . This situation is



■ **Algorithm 3** The algorithm finding a long  $(s, t)$ -path in a 2-connected graph with a given nested Erdős-Gallai decomposition.

---

```

long_nested_st_path( $(G_1, s_1, t_1), (G_2, s_2, t_2), \dots, (G_\ell, s_\ell, t_\ell)$ )
  Input: a nested Erdős-Gallai decomposition for  $G, s$  and  $t$ .
  Output: an  $(s, t)$ -path of length at least  $\delta(G - \{s, t\}) + f(k)/32 - 3$  in  $G$  where
            $k = L - \delta(G - \{s, t\})$  for the longest  $(s, t)$ -path length  $L$  in  $G$ .
3.1  $H \leftarrow$  nested_compress( $(G_1, s_1, t_1), (G_2, s_2, t_2), \dots, (G_\ell, s_\ell, t_\ell)$ );
3.2  $Q \leftarrow$  long_st_path_approx( $H, s, t$ );
3.3  $Q \leftarrow$  nested_decompress( $(G_1, s_1, t_1), (G_2, s_2, t_2), \dots, (G_\ell, s_\ell, t_\ell), H, Q$ );
3.4 foreach  $i \in [\ell]$  do
3.5    $P_i \leftarrow$  the longest of
       {long_st_path_approx( $G_i, s_i, t_i$ ), long_eg_st_path( $G_i, s_i, t_i$ )};
3.6    $Q \leftarrow$  the longest of  $\{Q, \text{two\_long\_disjoint\_paths}(G, \{s, t\}, \{s_i, t_i\}, 0) \cup P_i\}$ ;
3.7 end
3.8 return  $Q$ ;
```

---

beneficial to us in a different way: since we can find such a pair of paths in polynomial time, we can traverse at least  $d_i + 1$  edges going from  $G_{e(i)}$  to  $G_i$ , while we only lose at most  $d_i$  in the minimum degree. This dichotomy on the structure of the “slice” between two nested components is the main leverage that allows us to lift the length of an  $(s, t)$ -path in  $H$  to an offset above the minimum degree in  $G$ . We formally show this crucial property of the compressed graph  $H$  and the `nested_decompress` routine in the next lemma.

► **Lemma 19.** *The `nested_decompress` routine transforms an  $(s, t)$ -path  $Q$  in  $H$  of length  $r$  into an  $(s, t)$ -path in  $G$  of length at least  $\delta(G - \{s, t\}) + r/8 - 3$ .*

It will also be helpful to observe that in the “slice” between a decomposed component and the nested components, at most two edges of any path can be contracted. Note that this does not follow immediately, as a pair of edges to *each* of the nested components is potentially contracted.

▷ **Claim 20.** Let  $Q$  be an  $(s_j, t_j)$ -path inside a decomposed graph  $G_j$ . Then all edges  $E(Q) \cap E(G_j) \setminus \bigcup_{e(i)=j} E(G_i)$  are unchanged in  $H$  except for, possibly, contraction of the first and the last edge of  $Q$ .

Now we are ready to prove the main lemma that bounds the length of the  $(s, t)$ -path returned by Algorithm 3.

► **Lemma 21.** *long\_nested\_st\_path outputs an  $(s, t)$ -path in  $G$  of length at least  $\delta(G - \{s, t\}) + f(k)/32 - 3$ , where  $k = L - \delta(G - \{s, t\})$  and  $L$  is the length of the longest  $(s, t)$ -path in  $G$ .*

Finally, observe that the running time of Algorithm 3 is polynomial in the size of the given nested Erdős-Gallai decomposition. By Lemma 17, its size is polynomial in the size of the input graph  $G$ . This concludes the proof of Theorem 2.

## 5 Conclusion

In this article, we have shown a general theorem that allows us to leverage all the algorithmic machinery for approximating the length of the longest cycle to approximate the “offset” of the longest cycle provided by the classical Dirac’s theorem. As far as one can compute

a cycle of length  $f(L)$  in a 2-connected graph  $G$  with the longest cycle length  $L$ , we can also construct a cycle of length  $2\delta(G) + \Omega(f(L - 2\delta(G)))$ . In particular, we can use the state-of-the-art approximation algorithm for Longest Cycle due to Gabow and Nie [31]. They achieve an algorithm finding a cycle of length  $f(L) = c\sqrt{\log L}$  for some constant  $c > 1$  in a graph with the longest cycle length  $L$ . Note that  $f$  is non-decreasing and subadditive (as  $f$  is concave on  $[1, +\infty]$ , and any concave function is subadditive; we also can formally set  $f(x) = \min\{x, c\sqrt{\log x}\}$  for  $x \geq 1$  and  $f(x) = x$  for  $x < 1$  to fit the statement of Theorem 1). By substituting this to Theorem 1, we achieve a polynomial-time algorithm that outputs a cycle of length  $2\delta(G) + 2^{\Omega(\sqrt{\log(L - 2\delta(G))})}$  in a 2-connected graph  $G$  with the longest cycle length  $L > 2\delta(G)$ .

In the field of parameterized algorithms, there are many results on computing longest cycles or paths above some guarantees. It is a natural question, whether approximation results similar to ours hold for other types of “offsets”. To give a few concrete questions, recall that the *degeneracy*  $\text{dg}(G)$  of a graph  $G$  is the maximum  $d$  such that  $G$  has an induced subgraph of minimum degree  $d$ . By Erdős and Gallai [19], a graph of degeneracy  $d \geq 2$  contains a cycle of length at least  $d + 1$ . It was shown by Fomin et al. in [22] that a cycle of length at least  $L = \text{dg}(G) + k$  in a 2-connected graph can be found in  $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$  time. This immediately yields a polynomial-time algorithm for computing a cycle of length at least  $\text{dg}(G) + \Omega(\log(L - \text{dg}(G)))$ . Is there a better approximation of the longest cycle above the degeneracy?

Another concrete question. Bezáková et al. [4] gave an FPT algorithm that for  $s, t \in V(G)$  finds a detour in an undirected graph  $G$ . In other words, they gave an algorithm that finds an  $(s, t)$ -path of length at least  $L = \text{dist}_G(s, t) + k$  in  $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$  time. Here  $\text{dist}_G(s, t)$  is the distance between  $s$  and  $t$ . Therefore, in undirected graph we can find an  $(s, t)$ -path of length  $\text{dist}_G(s, t) + \Omega(\log(L - \text{dist}_G(s, t)))$  in polynomial time. The existence of any better bound is open. For directed graphs, the question of whether finding a long detour is FPT is widely open [4]. Nothing is known about the (in)approximability of long detours in directed graphs.

---

## References

- 1 Noga Alon, Gregory Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. Solving MAX- $r$ -SAT above a tight lower bound. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 511–517. SIAM, 2010.
- 2 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.
- 3 Cristina Bazgan, Miklos Santha, and Zsolt Tuza. On the approximation of finding a(nother) hamiltonian cycle in cubic hamiltonian graphs. *J. Algorithms*, 31(1):249–268, 1999. doi:10.1006/jagm.1998.0998.
- 4 Ivona Bezáková, Radu Curticapean, Holger Dell, and Fedor V. Fomin. Finding detours is fixed-parameter tractable. *SIAM J. Discrete Math.*, 33(4):2326–2345, 2019. doi:10.1137/17M1148566.
- 5 Andreas Björklund. Determinant sums for undirected hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014. doi:10.1137/110839229.
- 6 Andreas Björklund and Thore Husfeldt. Finding a path of superlogarithmic length. *SIAM J. Comput.*, 32(6):1395–1402, 2003. doi:10.1137/S0097539702416761.
- 7 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *CoRR*, abs/1007.1161, 2010. arXiv:1007.1161.
- 8 Andreas Björklund, Thore Husfeldt, and Sanjeev Khanna. Approximating longest directed paths and cycles. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3142 of *Lecture Notes in Comput. Sci.*, pages 222–233. Springer, 2004. doi:10.1007/978-3-540-27836-8\_21.

- 9 Hans L. Bodlaender. On linear time minor tests with depth-first search. *J. Algorithms*, 14(1):1–23, 1993. doi:10.1006/jagm.1993.1001.
- 10 B. Bollobás and A. D. Scott. Better bounds for Max Cut. In *Contemporary combinatorics*, volume 10 of *Bolyai Soc. Math. Stud.*, pages 185–246. János Bolyai Math. Soc., Budapest, 2002.
- 11 Béla Bollobás. Extremal graph theory. In *Handbook of combinatorics, Vol. 1, 2*, pages 1231–1292. Elsevier Sci. B. V., Amsterdam, 1995.
- 12 J. A. Bondy. Basic graph theory: paths and circuits. In *Handbook of combinatorics, Vol. 1, 2*, pages 3–110. Elsevier Sci. B. V., Amsterdam, 1995.
- 13 Guantao Chen, Zhicheng Gao, Xingxing Yu, and Wenan Zang. Approximating longest cycles in graphs with bounded degrees. *SIAM Journal on Computing*, 36(3):635–656, 2006.
- 14 Robert Crowston, Mark Jones, Gabriele Muciaccia, Geevarghese Philip, Ashutosh Rai, and Saket Saurabh. Polynomial kernels for lambda-extendible properties parameterized above the Poljak-Turzik bound. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 24 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43–54, Dagstuhl, Germany, 2013. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- 15 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
- 16 Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 5th edition, 2017.
- 17 G. A. Dirac. Some theorems on abstract graphs. *Proc. London Math. Soc. (3)*, 2:69–81, 1952.
- 18 C. S. Edwards. Some extremal properties of bipartite subgraphs. *Canad. J. Math.*, 3:475–485, 1973.
- 19 P. Erdős and T. Gallai. On maximal paths and circuits of graphs. *Acta Math. Acad. Sci. Hungar.*, 10:337–356, 1959.
- 20 Tomás Feder and Rajeev Motwani. Finding large cycles in Hamiltonian graphs. *Discrete Appl. Math.*, 158(8):882–893, 2010. doi:10.1016/j.dam.2009.12.006.
- 21 Tomás Feder, Rajeev Motwani, and Carlos Subi. Approximating the longest cycle problem in sparse graphs. *SIAM J. Comput.*, 31(5):1596–1607, 2002. doi:10.1137/S0097539701395486.
- 22 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Going far from degeneracy. *SIAM J. Discrete Math.*, 34(3):1587–1601, 2020. doi:10.1137/19M1290577.
- 23 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Multiplicative parameterization above a guarantee. *ACM Trans. Comput. Theory*, 13(3):18:1–18:16, 2021. doi:10.1145/3460956.
- 24 Fedor V. Fomin, Petr A. Golovach, Danil Sagunov, and Kirill Simonov. Algorithmic extensions of Dirac’s theorem. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 406–416, 2022. doi:10.1137/1.9781611977073.20.
- 25 Fedor V. Fomin, Petr A. Golovach, Danil Sagunov, and Kirill Simonov. Longest cycle above erdős-gallai bound. In *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, pages 55:1–55:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.55.
- 26 Fedor V. Fomin, Petr A. Golovach, Danil Sagunov, and Kirill Simonov. Approximating long cycle above dirac’s guarantee. *CoRR*, abs/2305.02011, 2023. arXiv:2305.02011.
- 27 Fedor V. Fomin and Petteri Kaski. Exact exponential algorithms. *Commun. ACM*, 56(3):80–88, 2013. doi:10.1145/2428556.2428575.
- 28 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.
- 29 Martin Fürer and Balaji Raghavachari. Approximating the minimum-degree steiner tree to within one of optimal. *J. Algorithms*, 17(3):409–423, 1994. doi:10.1006/jagm.1994.1042.
- 30 Harold N. Gabow. Finding paths and cycles of superpolylogarithmic length. *SIAM J. Comput.*, 36(6):1648–1671, 2007. doi:10.1137/S0097539704445366.

- 31 Harold N. Gabow and Shuxin Nie. Finding a long directed cycle. *ACM Transactions on Algorithms*, 4(1), 2008. doi:10.1145/1328911.1328918.
- 32 Harold N. Gabow and Shuxin Nie. Finding long paths, cycles and circuits. In *Proceedings of the 19th International Symposium on Algorithms and Computation (ISAAC)*, volume 5369 of *Lecture Notes in Comput. Sci.*, pages 752–763. Springer, 2008. doi:10.1007/978-3-540-92182-0\_66.
- 33 Shivam Garg and Geevarghese Philip. Raising the bar for vertex cover: Fixed-parameter tractability above a higher guarantee. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1152–1166. SIAM, 2016. doi:10.1137/1.9781611974331.ch80.
- 34 Gregory Gutin, Eun Jung Kim, Michael Lampis, and Valia Mitsou. Vertex cover problem parameterized above and below tight bounds. *Theory of Computing Systems*, 48(2):402–410, 2011. doi:10.1007/s00224-010-9262-y.
- 35 Gregory Gutin, Leo van Iersel, Matthias Mnich, and Anders Yeo. Every ternary permutation constraint satisfaction problem parameterized above average has a kernel with a quadratic number of variables. *J. Computer and System Sciences*, 78(1):151–163, 2012. doi:10.1016/j.jcss.2011.01.004.
- 36 Gregory Z. Gutin and Matthias Mnich. A survey on graph problems parameterized above and below guaranteed values. *CoRR*, abs/2207.12278, 2022. doi:10.48550/arXiv.2207.12278.
- 37 Gregory Z. Gutin and Viresh Patel. Parameterized traveling salesman problem: Beating the average. *SIAM J. Discrete Math.*, 30(1):220–238, 2016.
- 38 Gregory Z. Gutin, Arash Rafiey, Stefan Szeider, and Anders Yeo. The linear arrangement problem parameterized above guaranteed value. *Theory Comput. Syst.*, 41(3):521–538, 2007. doi:10.1007/s00224-007-1330-6.
- 39 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity. *J. Computer and System Sciences*, 63(4):512–530, 2001.
- 40 Bart M. P. Jansen, László Kozma, and Jesper Nederlof. Hamiltonicity below Dirac's condition. In *Proceedings of the 45th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 11789 of *Lecture Notes in Computer Science*, pages 27–39. Springer, 2019.
- 41 David R. Karger, Rajeev Motwani, and G. D. S. Ramkumar. On approximating the longest path in a graph. *Algorithmica*, 18(1):82–98, 1997. doi:10.1007/BF02523689.
- 42 Jon M. Kleinberg and Éva Tardos. *Algorithm design*. Addison-Wesley, 2006.
- 43 Ioannis Koutis. Faster algebraic algorithms for path and packing problems. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5125 of *Lecture Notes in Comput. Sci.*, pages 575–586. Springer, 2008.
- 44 Ioannis Koutis and Ryan Williams. Algebraic fingerprints for faster algorithms. *Commun. ACM*, 59(1):98–105, 2016. doi:10.1145/2742544.
- 45 Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Trans. Algorithms*, 11(2):15:1–15:31, 2014. doi:10.1145/2566616.
- 46 Meena Mahajan, Venkatesh Raman, and Somnath Sikdar. Parameterizing above or below guaranteed values. *J. Computer and System Sciences*, 75(2):137–153, 2009. doi:10.1016/j.jcss.2008.08.004.
- 47 Sounaka Mishra, Venkatesh Raman, Saket Saurabh, Somnath Sikdar, and C. R. Subramanian. The complexity of König subgraph problems and above-guarantee vertex cover. *Algorithmica*, 61(4):857–881, 2011. doi:10.1007/s00453-010-9412-2.
- 48 B. Monien. How to find long paths efficiently. In *Analysis and design of algorithms for combinatorial problems (Udine, 1982)*, volume 109 of *North-Holland Math. Stud.*, pages 239–254. North-Holland, Amsterdam, 1985. doi:10.1016/S0304-0208(08)73110-4.
- 49 Sundar Vishwanathan. An approximation algorithm for finding long paths in hamiltonian graphs. *J. Algorithms*, 50(2):246–256, 2004. doi:10.1016/S0196-6774(03)00093-2.
- 50 Ryan Williams. Finding paths of length  $k$  in  $O^*(2^k)$  time. *Inf. Process. Lett.*, 109(6):315–318, 2009.