Algorithmica (2024) 86:2676–2713
https://doi.org/10.1007/s00453-024-01240-5

# Approximating Long Cycle Above Dirac's Guarantee

**Fedor V. Fomin[1] · Petr A. Golovach[1] · Danil Sagunov[2] · Kirill Simonov[3]**

Received: 19 August 2023 / Accepted: 14 May 2024 / Published online: 30 May 2024

## 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

This work was originally presented at the ICALP 2023 conference. A preliminary version of this article appeared in the Proceedings of the 50th EATCS International Colloquium on Automata, Languages and Programming (ICALP 2023), volume 261 of LIPICS, pages 60:1–60:18. This full version contains all missing proofs and claims with detailed analysis.

✉ Petr A. Golovach
petr.golovach@ii.uib.no

Fedor V. Fomin
fomin@ii.uib.no

Danil Sagunov
danilka.pro@gmail.com

Kirill Simonov
kirillsimonov@gmail.com

[1] Department of Informatics, University of Bergen, Bergen, Norway

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

[3] Hasso Plattner Institute, University of Potsdam, Potsdam, Germany

of a longest cycle $L$ is at least $2\delta(G)$. Thus the "essential" part in 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 $2\delta(G) + \Omega(f(k))$.

**Keywords** Longest path · Longest cycle · Approximation algorithms · Above guarantee parameterization · Minimum degree · Dirac theorem

## 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* [25]. 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 [11, 18].) 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 [45] and it was successfully applied in the study of several fundamental problems in parameterized complexity and kernelization. For illustrative examples, we refer to [2, 4, 14, 25, 27, 32–34, 36, 37, 44], see also the recent survey of Gutin and Mnich [35]. Quite surprisingly, the theory of the above (or below) guarantee *approximation* remains unexplored. (Notable exceptions are the works of Mishra et al. [46] 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 [11].)

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, 7, 20, 21, 28, 29, 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 $f : \mathbb{R}_+ \to \mathbb{R}_+$ 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 [26], 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. [26]. 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 [47] 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 [1] 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 [5, 8, 23, 42, 50]. The surveys [22, 43], 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 [38]. 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 [29] 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 [31] 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 $L^{\varepsilon}$. 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 [40] proved that the longest path problem does not belong to APX unless P = NP. They also show that for any $\varepsilon > 0$, it cannot be approximated within $2^{\log^{1-\varepsilon} n}$ unless NP $\subseteq$ DTIME($2^{O(\log^{1/\varepsilon} n)}$). 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 [7, 30].

Our approximation algorithms are inspired by the recent work Fomin, Golovach, Sagunov, and Simonov [26] 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 parameterizes 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 [26] are not sufficient to go beyond $\Omega(\log k)$-bound on the offset. The main combinatorial tools from [26] 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 [10] provide an introduction to this fundamental subarea of graph theory. Besides [26], the algorithmic applications of Dirac's theorem from the perspective of parameterized complexity were studied by Jansen, Kozma, and Nederlof in [39].

*Paper structure* Sect. 2 provides an overview of the techniques employed to achieve our results. Then, Sect. 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. Section 5 is dedicated to the proof of Theorem 1 itself. Finally, we conclude with a summary and some open questions in Sect. 6.

## 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 arbitrary non-decreasing function $f : \mathbb{R}_+ \to \mathbb{R}_+$ 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 2 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. [26], 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, 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 $f : \mathbb{R}_+ \to \mathbb{R}_+$ 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 $f : \mathbb{R}_+ \to \mathbb{R}_+$ 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 $\delta(G) \le \frac{1}{2}|V(G)|$ and the longest cycle length $L$.*

One may note that Theorem 2 actually follows from Theorem 1 (again, by Menger's theorem, see Lemma 2). 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 [26]. 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. [26]. 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 Lemma 14 for details).

To see how this stuctural 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 [26] 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 $f(2\delta(G)+k) \ge f(k) \ge 2\delta(G)+4/5 \cdot f(k)$. Hence, by taking the longest of the cycles of length $(2 + \sigma_1) \cdot f(k)$ 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 $k' \ge 0$ 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 [26]. 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 [26] 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 $G$ in almost even parts. Our contribution is formulated in the following lemma.

**Lemma 1** *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 1 in the following way. Since there are $\mathcal{O}(|V(G)|^2)$ vertex pairs in $G$, our algorithms iterates over all vertex pairs. If a pair $u, v$ separates the graph in 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 approximation of the longest $(u, v)$-path. By Lemma 1, 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 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 present in Sect. 5.

## 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 [26] 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 Sect. 4, we give a complete definition of Erdős–Gallai decomposition. In this overview, we aim to avoid most technical details in order to provide an intuition of the structure of the decomposition and how our algorithm employs it.

Much 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 7; 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 [26] 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 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 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

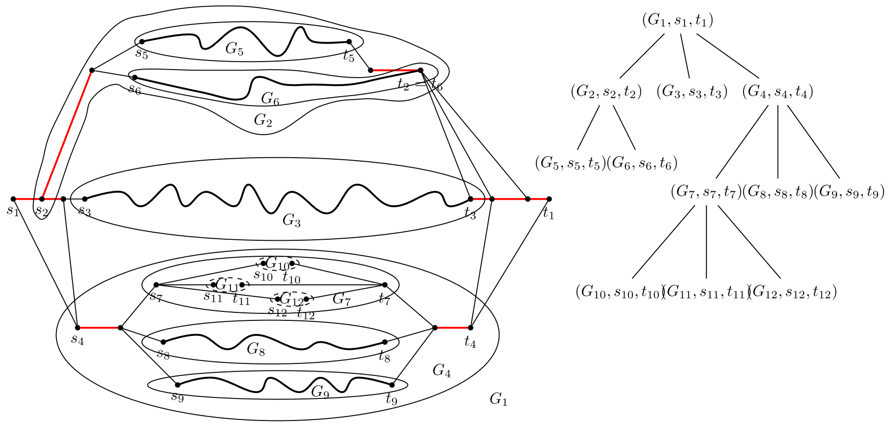
$G_1$
$G_2$
$G_3$
$G_4$
$G_5$
$G_6$
$G_7$
$G_8$
$G_9$
$G_{10}$
$G_{11}$
$G_{12}$
$s_1$ $s_2$ $s_3$ $s_4$ $s_5$ $s_6$ $s_7$ $s_8$ $s_9$ $s_{10}$ $s_{11}$ $s_{12}$
$t_1$ $t_2$ $t_3$ $t_4$ $t_5$ $t_6$ $t_7$ $t_8$ $t_9$ $t_{10}$ $t_{11}$ $t_{12}$
$(G_1, s_1, t_1)$
$(G_2, s_2, t_2)$ $(G_3, s_3, t_3)$ $(G_4, s_4, t_4)$
$(G_5, s_5, t_5)$ $(G_6, s_6, t_6)$
$(G_7, s_7, t_7)$ $(G_8, s_8, t_8)$ $(G_9, s_9, t_9)$
$(G_{10}, s_{10}, t_{10})$ $(G_{11}, s_{11}, t_{11})$ $(G_{12}, s_{12}, t_{12})$

**Fig. 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 (Color figure online)

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 Sect. 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 Fig. 1.

In Sect. 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 contains it fully, with all technical details and formal proofs.

## 3 Preliminaries

In this section, we define 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 $f\colon D \to \mathbb{R}$ is *subadditive* if $f(x+y) \le f(x)+f(y)$ 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 $f : \mathbb{R}_+ \to \mathbb{R}$, such that an algorithm achieveng the respective approximation exists. Throughout the proofs we will additionally assume that $f(x) \le x$ for every $x \in \mathbb{R}_+$. For any integer $x \ge 3$, 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]). We use $V[G]$ and $E(G)$ to denote the sets of vertices and edges, respectively, of a graph $G$. Throughout the paper, we use $n$ and $m$ to denote the number of vertices and the number of edges of a considered graph if it does not create confusion. For a set $X \subseteq V(G)$, $G[X]$ is used to denote the subgraph of $G$ *induced* by $X$ and we write $G - X$ to denote the subgraph of $G$ induced by $V(G) \setminus X$. For a single-vertex set $\{v\}$, we write $G - v$ instead of $G - \{v\}$. For a vertex $v$, $N_G(v)$ denotes the *(open) neighborhood* of $v$, that is, the set of the neighbors of $v$ in $G$. For a set set $X \subseteq V(G)$, $N_G(X) = \big(\bigcup_{v\in X} N_G(v)\big)\setminus X$. The *degree* of a vertex $v$ is $\deg_G(v) = |N_G(v)|$. We denote by $\delta(G) = \min_{v\in V(G)} \deg_G(v)$ the *minimum degree* of $G$. We may omit the subscript in the above notation if the considered graph is clear from the context. We remind that the *edge contraction* operation for $uv \in E(G)$ replaces $u$ and $v$ by a single vertex $w_{uv}$ that is adjacent to every vertex of $N_G(\{u, v\})$. A set of vertices $X \subseteq V(G)$ is *vertex cover* if every edge of $G$ has at least one endpoint in $X$.

A *path* $P$ in a graph $G$ is a subgraph of $G$ whose set of vertices can be written as $\{v_0, \dots, v_k\}$ where $E(P) = \{v_{i-1}v_i \mid i \in [k]\}$. We may write a path $P$ as the sequence of its vertices $v_0, \dots, v_k$. The vertices $v_0$ and $v_k$ are called *endpoints* of $P$ and other vertices are *internal*. For a path $P$ with endpoints $s$ and $t$, we say that $P$ is an $(s, t)$-path. Two paths $P_1$ and $P_2$ are *(vertex-)disjoint* if they have no common vertex and *internally disjoint* if no internal vertex of either of the paths is a vertex of the other path. A *cycle* $C$ in $G$ is a subgraph of $G$ with $V(C) = \{v_1, \dots v_k\}$ and $E(C) = \{v_{i-1}v_i \mid i \in [k]\}$, where $k \ge 3$ and it is assumed that $v_0 = v_k$. The *length* of a path (a cycle, respectively) is the number of its edges. For two internally disjoint paths $P_1 = v_0, \dots, v_k$ and $P_2 = u_0, \dots, v_s$ sharing exactly one endpoint $v_k = u_0$, we write $P_1 P_2$ to denote their *concatenation*, that is, the path $v_0, \dots, v_k, u_1, \dots, u_s$. If $P_1$ and $P_2$ share both endpoints and at least one of them has internal vertices, we write $P_1 P_2$ to denote the cycle composed by the paths. A path $P$ (a cycle $C$, respectively) is *Hamiltonian* if $V(P) = V(G)$ $(V(C) = V(G)$, respectively).

Recall that $G$ is *connected* if for every two vertices $s$ and $t$, $G$ contains an $(s,t)$-path. A *(connected) component* of $G$ is an inclusion maximal connected induced subgraph. A connected graph $G$ with at least three vertices is 2-*connected* if for every $v \in V(G)$, $G - v$ is connected. A vertex $v$ of a connected graph $G$ with at least two vertices is a *cut-vertex* if $G - v$ is disconnected. A *block* of a connected graph $G$ is an inclusion maximal induced subgraph without cut-vertices. Note that if $G$ has at least two vertices, then each block is either isomorphic to $K_2$ or a 2-connected graph. For a block $B$ of $G$, a vertex $v \in V(B)$ that is not a cut-vertex of $G$ is called *inner*. Blocks in a connected graph form a tree structure (viewing each block as a vertex of the forest and two blocks are adjacent if they share a cut-vertex). The blocks corresponding to the leaves of the block-tee, are called *leaf-blocks*. For $s, t \in V(G)$, $S \subseteq V(G) \setminus \{s, t\}$ is an $(s,t)$-*separator* if $G - S$ has no $(s,t)$-path; we also say that $S$ *separates* $s$ from $t$. We also say that $S$ separates two sets of vertices $A$ and $B$ if $S$ separates each vertex of $A$ from every vertex of $B$.

The following useful observation follows immediately from Menger's theorem (see, e.g., [16, 41]).

**Lemma 2** *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 3** *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 $G$.*

***Proof*** Let $G$ be a graph and let $s, t \in V(G)$ be distinct vertices. We assume without loss of generality that $G$ is connected. Let $P$ be a longest $(s,t)$-path in $G$ and let $L$ be its length. If $st$ is a bridge of $G$, then $G$ has a unique $(s,t)$-path and and its length is one. In this case, our algorithm returns this path that trivially can be found in polynomial time. Assume that this is not the case. Then $st \notin E(P)$ and $L \geq 2$.

We construct two copies $G_1$ and $G_2$ of $G$. Denote by $s_1$ and $s_2$ the copies of $s$ in $G_1$ and $G_2$, respectively. Similarly, let $t_1$ and $t_2$ be the copies of $t$, and denote by $P_1$ and $P_2$ the copes of $P$ in $G_1$ and $G_2$, respectively. Next, we construct the graph $G'$ by unifying $s_1$ and $s_2$, and $t_1$ and $t_2$ (if $st \in E(G)$, the edges $s_1 t_1$ and $s_2 t_2$ are unified as well). Denote by $s'$ the vertex of $G'$ obtained from $s_1$ and $s_2$, and let $t'$ be the vertex obtained from $t_1$ and $t_2$. Note that $P_1$ and $P_2$ are internally disjoint $(s', t')$-paths in $G'$. In particular, this implies that $s'$ and $t'$ are vertices of the same block $B$ of $G'$, and $P_1$ and $P_2$ are paths in $B$. Therefore, $B$ contains the cycle $C = P_1P_2$ of length $2L$. We obtain that the longest cycle length in $B$ is at least $2L$. We call $\mathcal{A}$ on $B$ and this algorithm outputs a cycle $C$ of length at least $f(2L)$. Note that $B$ is distinct from $K_2$, i.e., is 2-connected. By Lemma 2, $B$ has an $(s', t')$-path $P'$ of length at least $\frac{1}{2}|V(C)| \geq \frac{1}{2}f(2L)$ that can be constructed in polynomial time. Notice that $\{s', t'\}$ separates $V(G_1) \setminus \{s_1, t_1\}$ from $V(G_2)\setminus\{s_2, t_2\}$. Hence, $P'$ is either an $(s_1, t_1)$-path in $G_1$ or $(s_2, t_2)$-path in $G_2$. Assume that $P'$ is a path in $G_1$ (the other case is symmetric). Since $G_1$ is a copy of $G$, the copy of $P'$ in $G$ is an $(s, t)$-path of length at least $\frac{1}{2}f(2L)$ as required by the lemma.

Since $G'$ can be constructed in polynomial time and the unique block $B$ of $G'$ containing $s'$ and $t'$ can be found in polynomial (linear) time (see, e.g., [41]), the overall running time is polynomial. □

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 [26] 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 3** (Theorem 4 in [26]) *There exists an FPT algorithm for* Long $(s,\ t)$-Cycle *parameterized by k.*

For completeness, we show the corollary next.

**Corollary 1** *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.*

***Proof*** Construct a new graph $H$ that consists of the graph $G$ together with two additional vertices $u$ and $v$. The vertex $u$ has exactly two neighbors in $H$, $s$ and $t$, and the neighbors of $v$ are $s'$ and $t'$. Now run the algorithm for Long $(s, t)$-Cyclewith the parameter $k + 4$ to find a cycle in $H$ going through the vertices $u$ and $v$. If such a cycle is found, then removing the vertices $u$ and $v$ from it yields a pair of disjoint paths between $\{s, t\}$ and $\{s', t'\}$ in $G$ of total length at least $k$. In the other direction, if there is a pair of desired disjoint paths in $G$, then together with the vertives $u$ and $v$ they constitute a cycle of length at least $k + 4$ in $H$. □

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

**Corollary 2** (Corollary 3 in [26]) *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 provide the formal 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 $f : \mathbb{R}_+ \to \mathbb{R}_+$ 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 [26] 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 Erdos–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 [26]. Some of the results from [26] need to be modified in order to be used for our purposes, we supply such results with full proofs. Note that the statements in [26] 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 1** (Erdős–Gallai decomposition and Erdős–Gallai component, Definition 2 in [26]) 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 $H$ holds $|V(H)| \geq 3$ and one of the following.

(R1) $H$ 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;
(R2) $H$ 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 $P_2$;
(R3) The same as (R2), 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 component*s for an Erdős–Gallai decomposition
is defined as follows. First, for each component $H$ of type (R1), $H$ is an Erdős–Gallai component of the Erdős–Gallai decomposition. Second, for each $H$ of type (R2), or of type (R3), 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 to bound the structure of optimal solutions in a number of ways. First, Fomin et al. [26] observe that the longest $(s, t)$-path necessarily visits an Erdős–Gallai component.

**Lemma 4** (Lemma 7 in [26]) *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 5** (Lemma 5 in [26]) *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 6** (Lemma 8 in [26]) *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 analogue of Lemma 4 in [26], however our statement here is slightly modified. Next, we recall the statement from Sect. 2 and provide a proof.

**Lemma 7** *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.*

***Proof*** Invoke Lemma 4 of [26] on $G, s, t$ with $B := \{s,t\}$ and $k := \lfloor \delta(G-\{s,t\})/4 \rfloor - 2$. Note that the condition $4k+8 \leq \delta(G-\{s,t\})$ required by that lemma is satisfied. Now, we either get an $(s,t)$-path of length $\delta(G-\{s,t\})+k$, or an $(s,t)$-path $P$ with $V(P)\cup\{s,t\} = V(G)$, or the required Erdős–Gallai decomposition with the paths $P, P_1, P_2$. Clearly $\delta(G-\{s,t\})+k > \frac{5}{4}\delta(G-\{s,t\})-3$, so if a path of length $\delta(G-\{s,t\})+k$ is found, we are done. If an $(s,t)$-path $P$ has $V(P)\cup\{s,t\} = V(G)$, then it is a hamiltonian path in $G$, so we are done in the second case as well.

If we obtain an Erdős–Gallai decomposition, then we additionally need to check whether there exists an $(s,t)$-path that goes through at least two Erdős–Gallai components of the Erdős–Gallai decomposition induced by $P_1$ and $P_2$ in $G$. To this end, iterate over all ordered pairs of Erdős–Gallai components in the Erdős–Gallai decomposition. For each pair, apply Lemma 6 to each of the two Erdős–Gallai components and obtain two triples $(K_1, s_1, t_1)$ and $(K_2, s_2, t_2)$. There is an $(s,t)$-path entering both Erdős–Gallai components in the order given by the pair if and only if there exist three disjoint paths between the pairs $(s, a_1)$, $(b_1, a_2)$, $(b_2, t)$, where $(a_i, b_i)$ is a permutation of $(s_i, t_i)$ for each $i \in [2]$.

When the permutations are fixed, such paths, if they exist, can be found in polynomial time using the famous algorithm of Robertson and Seymour for $k$-Disjoint Paths [48]. Since $\delta(K_i - \{s_i, t_i\}) \geq \delta(G-\{s,t\})-2$ for each $i \in [2]$, these three paths together with two $(s_i, t_i)$-paths inside $K_i$ combine into an $(s,t)$-path of length at least $2\delta(G-\{s,t\})-4 > \frac{5}{4}\delta(G-\{s,t\})-3$, so the algorithm outputs this path and stops. If the disjoint path triple was not found on any of the steps, then there is indeed no $(s,t)$-path entering at least two Erdős–Gallai components. □

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 [26] that yields long paths inside separable components.

**Lemma 8** (Lemma 6 in [26]) *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 9 has been already stated in Sect. 2, here we recall the statement and provide a proof.

**Lemma 9** *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 $G$.*

***Proof*** For clarity, we denote $\delta := \delta(G-\{s,t\})$. Let $Q$ be the given $(s,t)$-path in $G$. Denote by $S$ the set of the first $k$ vertices on $Q$ and by $T$ the set of the last $k$ vertices on $Q$. Let $s'$ be the first vertex on $Q$ that is not in $S$ and $t'$ be the last vertex on $Q$ that is not in $T$. Since $Q$ consists of more than $2k$ vertices, $s', t' \notin S \cup T$. The length of the $(s,s')$-subpath of $Q$ and the length of the $(t',t)$-subpath of $Q$ are equal to $k$.

The total length of $P_1$ and $P_2$ is at most $k$, hence $|V(P_1) \cup V(P_2)| \leq k+2$. The length of the $(s',t')$-subpath of $Q$ is at least $2k+5 > 2|V(P_1) \cup V(P_2)|$. Hence, this subpath contains at least one edge of $G$ that is not incident to vertices in $|V(P_1) \cup V(P_2)|$. Denote the endpoints of this edge by $u$ and $v$. Since $Q$ does not enter any Erdős–Gallai component, this edge is an edge of a non-leaf-block of some separable connected component $H$ of $G - V(P_1 \cup P_2)$. The component $H$ corresponds to either (R2) or (R3) in the definition of Erdős–Gallai decomposition. Without loss of generality, we assume that $H$ corresponds to (R2).

We now consider two cases depending on the structure of $H-(S\cup T)$. If $S \cup T$ separates $u$ or $v$ from all cut vertices of the leaf-blocks in $H$, then we have a set of size $2k$ in $H$ that satisfies the condition of Lemma 8. Take a vertex $w$ in $H$ that has a neighbour in $V(P_2)$ in $G$. By Lemma 8, a $(w,c)$-path of length at least

$$\frac{1}{2}\delta(H)-2k \geq \frac{1}{2}\delta(G-V(P_1\cup P_2))-2k \geq \frac{1}{2}\delta(G-\{s,t\})-\frac{5}{2}k-1$$

exists in $H$ for some cut vertex $c$ of some leaf-block $L$ of $H$. In this leaf-block, we have a vertex $z$ with a neighbour in $V(P_1)$. By Corollary 2, we have a $(c,z)$-path of length at least $\delta(L-c) \geq \delta(G-\{s,t\})-2$ inside $L$. Combine the two paths and obtain a $(z,w)$-path of length at least $\frac{3}{2}\delta(G-\{s,t\})-\frac{5}{2}k-3$ inside $H$. Finally, prepend to this path a prefix of $P_1$ connecting $s$ with the neighbour of $z$, and append to this path a suffix of $P_2$ connecting the neighbour of $w$ with $t$. The length increases by at least two as $s \neq z$ and $w \neq t$. The obtained path is an $(s,t)$-path of length at least $\frac{3}{2}\delta(G-\{s,t\})-\frac{5}{2}k-1$.

The second case is when from $v$ we can reach a cut vertex $c$ of some leaf-block $L$ in $H$ while avoiding vertices in $S \cup T$. Note that $V(Q) \cap V(L-c) = \emptyset$ by the properties of Erdős–Gallai decomposition. Then choose $w$ as an arbitrary vertex in $L-c$ with a neighbour in $V(P_1)$. Now construct a $(v,s)$-path $Q'$ in the following way. First, follow an arbitrary $(v,c)$-path in $H-(S\cup T)$. Then continue with a $(c,w)$-path of length at least $\delta(L-c) \geq \delta(G-\{s,t\})-2$ inside $L-c$ that exists by Corollary 2. Note that this path has no common vertices with $Q$. Finish $Q'$ by going from $w$ to the neighbour of $w$ in $P_1$ and follow $P_1$ backwards down to $s$.

Let $x$ be the last vertex before $c$ on $Q'$ that belongs to $V(Q)$. Let $y$ be the first vertex after $w$ on $Q'$ that belongs to $V(Q)$. Both $x, y$ are defined correctly since $v, s \in V(Q)$. Consider the $(x,y)$-subpath of $Q'$. It strictly contains the $(c,w)$-path inside $L$, so its length is at least $\delta(G - \{s, t\}) - 1$. Also, the length of the $(s, x)$-subpath of $Q$ and the length of the $(x, t)$-subpath of $Q$ is at least $k$ as $x \notin S \cup T$.

We now construct a long $(s, t)$-path in $G$. If $y$ is contained in the $(s, x)$-subpath of $Q$, then the $(s, t)$-path is constructed in the following way: follow $P_1$ from $s$ to $y$, then follow $Q'$ backwards from $y$ down to $x$, and finish by following $Q$ from $x$ to $t$. The length of this path is at least $\delta(G - \{s, t\}) - 1 + k$. If $y$ belongs to the $(x, t)$-subpath of $Q$, start by taking the $(s, x)$-subpath of $Q$, then follow $Q'$ from $x$ to $y$ and finish by following $P_2$ from $y$ to $t$. This path also has length at least $k + \delta(G - \{s, t\}) - 1$. The proof is complete. □

## 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 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 Fig. 1. We recall the formal definition from Sect. 2.

**Definition 2** (Nested Erdős–Gallai decomposition) A sequence of triples $(G_1, s_1, t_1)$, $(G_2, s_2, t_2)$, …, $(G_\ell, s_\ell, t_\ell)$ 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 7 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 7 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 6 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 6 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 7 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 10** *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 $G$, $s$, $t$.*

***Proof*** The algorithm proceeds recursively, starting with the triple $(G_1, s_1, t_1) = (G, s, t)$. For the given triple $(G_i, s_i, t_i)$, if $\delta(G_i - \{s_i, t_i\}) < 16$, the algorithm stops. Otherwise, invoke the algorithm of Lemma 7 on $(G_i, s_i, t_i)$. If this returns a path $P_i$ of length at least $\frac{5}{4}\delta(G_i - \{s_i, t_i\}) - 3$, the algorithm stops. On the other hand, if an Erdős–Gallai decomposition is returned, for each Erdős–Gallai component $M$ run the algorithm of Lemma Lemma 6 on $M$ to obtain a triple $(G_j, s_j, t_j)$, where $j$ is the lowest free index among the triples produced so far. Run the main algorithm recursively on each of the triples generated on this step.

By definition, the algorithm above produces a nested Erdős–Gallai decomposition. To show that the running time is polynomial, first observe that running the algorithm without the subsequent recursive calls is clearly polynomial. Assume this running time is bounded by $\alpha n^c$ for some constant $\alpha > 0$ and $c \geq 1$, where $n = |V(G)\backslash\{s, t\}|$ and $(G, s, t)$ is the current instance. We show by induction on the depth of the resulting nested Erdős–Gallai decomposition that the running time of the recursive algorithm is at most $\alpha n^{c+1}$. If the instance does not spawn any recursive calls, this trivially holds. Otherwise, assume $\ell$ new instances $(G_{j_1}, s_{j_1}, t_{j_1})$, ..., $(G_{j_\ell}, s_{j_\ell}, t_{j_\ell})$ are produced, denote $n_i = |V(G_{j_i}\backslash\{s_{j_i}, t_{j_i}\}|$. Note that $\ell \geq 2$ since there are always at least 2 components in an Erdős–Gallai decomposition. By induction, the running time is bounded by $\alpha \cdot \left(n^c + \sum_{i=1}^{\ell} n_i^{c+1}\right)$. We now bound $\sum_{i=1}^{\ell} n_i^{c+1}$, observe first that $\sum_{i=1}^{\ell} n_i \leq n$, as all the sets $V(G_{j_i}\backslash\{s_{j_i}, t_{j_i}\}$ are disjoint and do not contain $s$ or $t$. We use the following numerical observation proven in [26].

**Claim 1** (Proposition 3 in [26]) *Let $a_1, a_2, \ldots, a_q$ be a sequence of $q \geq 2$ positive integers with $\sum_{i=1}^{q} a_i = n$. Let $x > 1$ be an integer. Then $\sum_{i=1}^{q} a_i^x \leq (n-1)^x + 1 \leq n^x - n^{x-1}$.*

By Claim 1, we can bound the running time by

$$\alpha \cdot \left(n^c + \sum_{i=1}^{\ell} n_i^{c+1}\right) \leq \alpha \cdot \left(n^c + n^{c+1} - n^c\right) = \alpha n^{c+1},$$

completing the proof. □

Clearly, it follows that the size of a nested Erdős–Gallai decomposition returned by Lemma 10 is also polynomial. Observe also that the construction algorithm invokes Lemma 7 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 1, 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 11) 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 2, 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 1, with the parameter 0.

```
nested_compress ($(G_1,s_1,t_1),(G_2,s_2,t_2),\dots,(G_\ell,s_\ell,t_\ell)$)
  Input: a nested Erdős–Gallai decomposition for $G$, $s$ and $t$.
  Output: the compressed graph $H$.
1.1  $H \longleftarrow G$;
1.2  foreach $i \in \{2,\dots,\ell\}$ do
1.3      $j \longleftarrow e(i)$;
1.4      $d_i \longleftarrow |\{s_j,t_j\} \setminus \{s_i,t_i\}|$;
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 $s_it_i$ to $H$ and mark it with $G_i$;
1.11     end
1.12 end
1.13 return $H$;
```
**Algorithm 1:** The algorithm compressing a given graph $G$ with a given nested Erdős–Gallai decomposition.

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

```
nested_decompress($(G_1, s_1, t_1), (G_2, s_2, t_2), \ldots, (G_\ell, s_\ell, t_\ell), H, Q$)
  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 $i \in \{2, \ldots, \ell\}$ such that $d_i > 0$ and $Q$ enters $G_i$ do
2.2      $j \longleftarrow e(i)$;
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          $S_1, S_2 \longleftarrow$ 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 $h \longleftarrow$ 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     $k' \longleftarrow \lfloor (|E(Q) \cap E(G_h)| - 5)/8 \rfloor$;
2.15     if $|E(P_h)| \geq \delta(G_h - \{s_h, t_h\}) + k'$ then
2.16         $R \longleftarrow P_h$;
2.17     else
2.18         $R \longleftarrow$ result of Lemma 9 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 $Q$;
```

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

```
long_nested_st_path($(G_1, s_1, t_1), (G_2, s_2, t_2), \ldots, (G_\ell, s_\ell, t_\ell)$)
  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  $H \longleftarrow$ nested_compress($(G_1, s_1, t_1), (G_2, s_2, t_2), \ldots, (G_\ell, s_\ell, t_\ell)$);
3.2  $Q \longleftarrow$ long_st_path_approx($H, s, t$);
3.3  $Q \longleftarrow$ nested_decompress($(G_1, s_1, t_1), (G_2, s_2, t_2), \ldots, (G_\ell, s_\ell, t_\ell), H, Q$);
3.4  foreach $i \in [\ell]$ do
3.5      $P_i \longleftarrow$ 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      $Q \longleftarrow$ the longest of {$Q$, two_long_disjoint_paths($G, \{s,t\}, \{s_i, t_i\}, 0) \cup P_i$};
3.7  end
3.8  return $Q$;
```

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

more edges that we collect while going 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 6 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 2** *For each $j \in [\ell]$, let $G_{j_1}, \ldots, G_{j_c}$ 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 has 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 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 11** *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 $\delta(G - \{s, t\}) + r/8 - 3$.*

***Proof*** Observe that in the tree of the nested Erdős–Gallai decomposition, the path $Q$ visits a rooted subpath of components $G_i$. That is, there are indices $j_1, j_2, \ldots, j_c \in [\ell]$ such that $j_1 = 1$ and $e(j_{i+1}) = j_i$ for each $i \in [c-1]$. This holds since in a Erdős–Gallai decomposition on each level, $Q$ visits at most one Erdős–Gallai component by Lemma 7. Here we say that $Q$ visits a component $G_i$ if $Q$ contains an edge of $G_i$ that was not contracted in $H$, and for non-decomposed components $G_i$ this means that $Q$ contains the edge $s_i t_i$ in $H$.

By Lemma 6, $\delta(G_{j_{i+1}}) \geq \delta(G_{j_i}) - d_{j_{i+1}}$. Let $h \in [\ell]$ be the largest integer such that $Q$ enters $G_h$, $h = j_c$. Denote by $p$ be the number of edges in $E(Q) \setminus E(G_h)$ and by $y$ the length of the $(s_h, t_h)$-subpath of $Q$, then $p + q = r$.

We now analyze the length of $Q$ after performing the replacement operations in Lines 2.1– 2.9. Denote by $Y$ the set of all $i \in [c-1]$ such that no contraction was made in Line 1.6 between $\{s_{j_i}, t_{j_i}\}$ and $\{s_{j_{i+1}}, t_{j_{i+1}}\}$. For each $i \in Y$ with $d_{j_{i+1}} > 0$, performing the replacement operation in Line 2.7 between $\{s_{j_i}, t_{j_i}\}$ and $\{s_{j_{i+1}}, t_{j_{i+1}}\}$ in $Q$ yields

$$|E(Q) \cap E(G_{j_i}) \setminus E(G_{j_{i+1}})| \geq d_{j_{i+1}} + 1 \geq 3d_{j_{i+1}}/2.$$

Let $p'$ be the length of $Q$ outside of $G_h$ after all these replacements, from the above $p' \geq \frac{3}{2} \sum_{i\in Y} d_{j_{i+1}}$. Also, $p' \geq p$ since the replacement only takes place if it makes the path longer.

Denote by $X := [c-1] \setminus Y$ the set of all $i \in [c-1]$ such that a contraction was made in Line 1.6 between $\{s_{j_i}, t_{j_i}\}$ and $\{s_{j_{i+1}}, t_{j_{i+1}}\}$. For each $i \in X$ the algorithm reverses the respective edge contractions done in Line 1.6 in $Q$. This increases the length of $Q$ by $d_{j_{i+1}}$, so after Line 2.9 it holds that $|E(Q) \backslash E(G_h)| \geq p' + \sum_{i \in X} d_{j_{i+1}}$.

We now observe that any long $(s_h, t_h)$-subpath in $G_h$ can be combined with $Q$ to preserve at least a constant fraction of $p$ in the offset.

**Claim 3** *After Line 2.9, replacing the $(s_h, t_h)$-subpath of $Q$ with a path $P$ in $G_h$ of length $\delta(G_h - \{s_h, t_h\}) + k'$, where $k'$ is a nonnegative integer, yields an $(s, t)$-path in $G$ of length at least*

$$\delta(G - \{s, t\}) + k' + p/3.$$

***Proof of Claim 3*** The length of the resulting path is at least

$$\begin{aligned}
|E(Q) \setminus E(G_j)| + |E(P)| &\geq p' + \sum_{i \in X} d_{j_{i+1}} + \delta(G_h - \{s_h, t_h\}) + k' \\
&\geq p' + \sum_{i \in X} d_{j_{i+1}} + \delta(G - \{s, t\}) \\
&\quad - \sum_{i \in [c-1]} d_{j_{i+1}} + k' \geq \delta(G - \{s, t\}) + k' + p' - \sum_{i \in Y} d_{j_{i+1}} \\
&\geq \delta(G - \{s, t\}) + k' + p/3.
\end{aligned}$$

Note that the last inequality holds since $p'$ is at least $\frac{3}{2} \sum_{i \in Y} d_{j_{i+1}}$ and also at least $p$. The path obtained at this point is an $(s, t)$-path in $G$ with possibly some contracted edges, since not all edge contractions were reversed. Reverse all remaining edge contractions affecting $Q$ and obtain an $(s, t)$-path in $G$ of at least the same length. □

For estimating the length of the $(s_h, t_h)$-subpath, consider two cases depending on the type of $G_h$.

$G_h$ **is not decomposed.** In this case, $Q$ contains the edge $s_h t_h$ in $H$, and in Line 2.12 this edge is replaced with the path $P_h$. By Claim 3, this yields a path of length at least $\delta(G - \{s, t\}) + (r - 1)/3$, since the length of $P_h$ is at least $\delta(G_h - \{s_h, t_h\})$, and $p = r - 1$.

$G_h$ **is decomposed.** By the choice of $j$, the $(s_h, t_h)$-subpath of $Q$ does not enter any Erdős–Gallai component in the Erdős–Gallai decomposition induced by $P_{j,1}$ and $P_{j,2}$ in $G_h$.

By Claim 3, an $(s_h, t_h)$-path of length $\delta(G_h - \{s_h, t_h\}) + k'$ inside $G_h$ combined with the outer part of $Q$ obtains an $(s, t)$-path of length at least $\delta(G - \{s, t\}) + p/3 + k'$ inside $G$. We now focus on identifying a long enough $(s_h, t_h)$-path inside $G_h$.

Let $k' := \lfloor (q - 5)/8 \rfloor$, so $q \geq 8k' + 5 \geq 4k' + 5$. If $P_h$ is longer than $\delta(G_h - \{s_h, t_h\}) + k'$, then plugging $P_h$ into Claim 3 gives an $(s, t)$-path of length at least $\delta(G - \{s, t\}) + p/3 + k' + 1 \geq \delta(G - \{s, t\}) + p/3 + (q - 5)/8 > \delta(G - \{s, t\}) + r/8 - 1$. Otherwise, we apply Lemma 9 to $G_h$, $P_h$ and the $(s_h, t_h)$-subpath of $Q$ to obtain an $(s_h, t_h)$-path $R$ in $G_h$.

If the length of $R$ is at least $\delta(G_h - \{s_h, t_h\}) + k' - 1$, Claim 3 gives the desired bound of $\delta(G - \{s, t\}) + r/8 - 3$. Otherwise, $\frac{1}{2}\delta(G_h - \{s_h, t_h\}) - \frac{5}{2}k' < k'$, then

$7k' > \delta(G_h - \{s_h, t_h\})$. It follows that $q > \delta(G_h - \{s_h, t_h\}) + q/8 + 5$. Hence, by applying Claim 3 to the initial $(s_h, t_h)$-subpath of $Q$ we get a path of length at least $\delta(G - \{s, t\}) + p/3 + q/8 + 5 > \delta(G - \{s, t\}) + r/8$. Since Algorithm 2 takes the longest of $R$ and the original subpath of $Q$, both cases are covered. □

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 4** *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 $Q$.*

***Proof of Claim 4*** Let $i$ be such that a contraction is made for $G_i$ in Line 1.6 with $e(i) = j$ and $d_i > 0$. There are no two disjoint paths between $\{s_j, t_j\}$ and $\{s_i, t_i\}$ of total length at least $d_i + 1$.

Without loss of generality, we assume that $s_i \neq s_j$, $t_i \neq s_j$ and $s_i \neq t_j$ and the edge $s_j s_i$ is contracted. If $s_i \notin V(Q)$, then $Q$ is not affected in $H$. We assume that $s_i \in V(Q)$. If $s_i$ is the second vertex in $V(Q)$, then $s_j s_i$ is the first edge of $V(Q)$ as required.

Suppose now that $s_i$ is not the second vertex in $Q$. Then the $(s_j, s_i)$-subpath of $Q$ is of length at least two. If $t_i$ does not belong to this subpath, we add the trivial (of length zero or one) $(t_i, t_j)$-path in $G_j$ and obtain two disjoint paths between $\{s_j, t_j\}$ and $\{s_i, t_i\}$ of total length at least $2 + |\{t_i, t_j\}| - 1 > d_i$, which is a contradiction.

Hence, $t_i$ is present on the $(s_j, s_i)$-subpath of $Q$. Then $t_i \neq t_j$, so $d_i = 2$ and $s_i, t_i, s_j, t_j$ are all distinct. We have an $(s_j, t_i)$-subpath of $Q$ and an $(s_i, t_j)$-subpath of $Q$ which are disjoint. If one of them is of length at least two, then we have two disjoint paths of total length more than $d_i$. Hence, $s_j t_i$ and $s_i t_j$ are the first and the last edge in $Q$. The proof of the claim is complete. □

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

**Lemma 12** `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 $G$.*

***Proof*** Let $T$ be the longest $(s, t)$-path in $G$, $|E(T)| = L = \delta(G - \{s, t\}) + k$. Our aim is to show that either $T$ yields a sufficiently long path in $H$ to use Lemma 11, or conclude that after contractions most of the path stays inside one Erdős–Gallai component, the deepest component visited. In case of the latter, we show that it suffices to take a long path inside this component.

We now introduce some notations for $T$ with respect to the nested Erdős–Gallai decomposition structure, similarly to the proof of Lemma 11. Let $h \in [\ell]$ be the largest integer such that $T$ enters $G_h$. Let $j_1, j_2, \ldots, j_c$ be such that $j_c = j$, $j_1 = 1$ and $e(j_{i+1}) = j_i$ for each $i \in [c - 1]$. Denote by $Y$ the set of all $i \in [c - 1]$ such that no contraction was made in Line 1.6 between $\{s_{j_i}, t_{j_i}\}$ and $\{s_{j_{i+1}}, t_{j_{i+1}}\}$. Denote $X := [c - 1] \setminus Y$.

Consider what happens to the path $T$ in the graph $H$ between two consecutive nested components $G_{j_i}$ and $G_{j_{i+1}}$. Since $T$ is the longest $(s,t)$-path in $G$, edges in $E(T)\cap E(G_{j_i})\backslash E(G_{j_{i+1}})$ form two disjoint paths between $\{s_{j_i},t_{j_i}\}$ and $\{s_{j_{i+1}},t_{j_{i+1}}\}$ of longest possible total length. If $|E(T)\cap E(G_{j_i})\backslash E(G_{j_{i+1}})|=d_{j_{i+1}}$, all edges in $E(T)\cap E(G_{j_i})\backslash E(G_{j_{i+1}})$ are contracted in $H$.

Otherwise, $|E(T)\cap E(G_{j_i})\backslash E(G_{j_{i+1}})|>d_{j_{i+1}}$. By Claim 4, all edges in $E(T)\cap E(G_{j_i})\setminus E(G_{j_{i+1}})$ are present in $H$, except for possibly $d_{j_{i+1}}$ of them (the first and/or the last). Also, recall that by properties of nested Erdős–Gallai decomposition, $T$ does not enter any $G_i$ with $e(i)=j_i$ except for $G_{j_{i+1}}$. Then the removal of the internal vertices of non-decomposed components does not affect edges in $E(T)\cap E(G_{j_i})\backslash E(G_{j_{i+1}})$. Hence, at least $|E(T)\cap E(G_{j_i})\backslash E(G_{j_{i+1}})|-d_{j_{i+1}}$ of the edges are present in $H$ in this case, which is at least one third of the edges in $E(T)\cap E(G_{j_i})\setminus E(G_{j_{i+1}})$ since $d_{j_{i+1}}\le 2$.

Let $T'$ be the path $T$ with all contractions applied to $H$. If $G_h$ is not decomposed, we assume that $T'$ contains the edge $s_ht_h$ marked with $G_h$. By the above, we have

$$\begin{aligned}|E(T')\backslash E(G_h)| &\ge \sum_{i\in[c-1]}|E(T)\cap E(G_{j_i})\backslash E(G_{j_{i+1}})|-d_{j_{i+1}}\\ &\ge \frac{1}{3}(|E(T)\backslash E(G_h)|-\sum_{i\in X}d_{j_{i+1}}).\end{aligned}$$

The last inequality holds since for each $i\in Y$ with $d_{j_{i+1}}>0$, $|E(T)\cap E(G_{j_i})\backslash E(G_{j_{i+1}})|-d_{j_{i+1}}$ is at least $\frac{1}{3}|E(T)\cap E(G_{j_i})\setminus E(G_{j_{i+1}})|$, and for all the remaining indices $i$, $|E(T)\cap E(G_{j_i})\backslash E(G_{j_{i+1}})|=d_{j_{i+1}}$. Denote $p:=|E(T')\backslash E(G_h)|$, and from the above obtain the equivalent

$$|E(T)\setminus E(G_h)|\le 3p+\sum_{i\in X}d_{j_{i+1}}. \tag{1}$$

If $p\ge k/4$, then an $(s,t)$-path of length at least $k/4+1$ is present in $H$. In this case, an approximation of the longest $(s,t)$-path in $H$ in Line 3.2 of `long_nested_st_path` gives a path of length at least $f(k/4+1)\ge f(k)/4$. By Lemma 11, running `nested_decompress` on this path results in an $(s,t)$-path of length at least $\delta(G-\{s,t\})+f(k)/32-3$ in $G$, so in this case the proof is finished.

Otherwise, $p<k/4$. Denote by $T_h$ the $(s_h,t_h)$-subpath of $T$. For simplicity, denote $\delta:=\delta(G-\{s,t\})$ and $\delta_h:=\delta(G_h-\{s_h,t_h\})$. We consider two cases.

*$G_h$* **is decomposed.** In this case, $T_h$ does not enter any Erdős–Gallai component of $G_h$. By Claim 4, at most two edges of $T_h$ can be contracted in $H$, denote the number of such edges by $d'_h$. Also, at least one edge is not contracted, so $|E(T_h)|-d'_h\ge|E(T_h)|/3$. Thus, $T'$ is an $(s,t)$-path of length at least $p+|E(T_h)|-d'_h\ge\frac{1}{3}\left(|E(T)|-\sum_{i\in X}d_{j_{i+1}}\right)$ in $H$. If $\sum_{i\in X}d_{j_{i+1}}\ge\delta+k/4-3$, then `long_nested_st_path` outputs an $(s,t)$-path of length at least $\delta+k/4-3$ by Claim 2, regardless of the length of $P_h$ at Line 3.5.

Otherwise, $T'$ is an $(s,t)$-path in $H$ of length at least $\frac{1}{3}(3k/4+3)=k/4+1$. Analogously to the case $p\ge k/4$, `long_nested_st_path` then finds a path of length at least $f(k)/4$, and the path $Q$ returned by `nested_decompress` is of length at least $\delta(G - \{s, t\}) + f(k)/32 - 3$.

$G_h$ **is not decomposed.** Here, our goal is to show that taking in $G_h$ one of the $(s_h, t_h)$-paths computed on Line 3.5 together with an arbitrary connection from $\{s, t\}$ to $\{s_h, t_h\}$ gives a long enough $(s, t)$-path in $G$. Let $k_h := |E(T_h)| - \delta_h$. Note that by the choice of $T$, $T_h$ is the longest $(s_h, t_h)$-path in $G_h$. We first show the following.

**Claim 5** *If $G_h$ is not decomposed, then at Line 3.6 the length of $P_h$ is at least $\delta_h + f(k_h)/8 - 3$.*

***Proof of Claim 5*** First note that if the length of $P_h$ from definition of nested Erdős–Gallai decomposition is at least $|V(G_h)| - 1$, then $P_h$ is a hamiltonian $(s_h, t_h)$-path in $G_h$. Then its length is maximum possible and is equal to $|E(T_h)| = \delta_h + k_h \geq \delta_h + f(k_h)$. Hence, we can assume that the length of $P_h$ given by nested Erdős–Gallai decomposition is at least $\frac{5}{4}\delta_h - 3$.

If $f(k_h) \geq \frac{8}{7}\delta_h$, then `long_st_path_approx(`$G_h, s_h, t_h$`)`, the blackbox $(s, t)$-path approximation algorithm, returns a path of length at least $f(\delta_h + k_h) \geq f(k_h) \geq \delta_h + f(k_h)/8$, so we are done.

Otherwise, $f(k_h) \leq \frac{8}{7}\delta_h$. If $f(k_h) \leq 24$, then it suffices for $P_h$ to have length $\delta_h$. In this case `long_eg_st_path(`$G_h$, $s_h$, $t_h$`)`, the exact $(s, t)$-path algorithm from Corollary 2, returns a $(s_h, t_h)$-path of length at least $\delta_h$.

It only remains to deal with the case where $f(k_h) > 24$, and $\delta_h \geq \frac{7}{8} f(k_h)$. Since $G_h$ is not decomposed and $\delta_h > 16$, by definition of nested Erdős–Gallai decomposition, $P_h$ is of length at least

$$\frac{5}{4}\delta_h - 3 \geq \delta_h + \frac{1}{4}\delta_h - 3 \geq \delta_h + \frac{1}{4} \cdot \frac{7}{8} f(k_h) - 3 \geq \delta_h + \frac{1}{8} f(k_h) - 3.$$

□

Using (1), we can lower-bound $k_h$ by

$$k_h = |E(T)| - |E(T) \setminus E(G_h)| - \delta_h \geq |E(T)| - 3p - \sum_{i \in X} d_{j_{i+1}} - \delta_h = \delta + k - 3p - \sum_{i \in X} d_{j_{i+1}} - \delta_h. \tag{2}$$

On Line 3.6, $P_h$ is transformed into an $(s, t)$-path in $G$ of length at least $|E(P_h)| + \sum_{i \in X \cup Y} d_{j_{i+1}}$, by Claim 2. By Claim 5, this length is at least

$$\begin{aligned} &\delta_h + f(k_h)/8 - 3 + \sum_{i \in X \cup Y} d_{j_{i+1}} \\ &\qquad \geq \delta + (\delta_h - \delta) + f\left(k + \delta - 3p - \sum_{i \in X} d_{j_{i+1}} - \delta_h\right)/8 + \sum_{i \in X \cup Y} d_{j_{i+1}} - 3 \\ &\qquad \geq \delta + \underbrace{(\delta_h + \sum_{i \in X \cup Y} d_{j_{i+1}} - \delta)}_{\geq 0 \text{ by Lemma 6}} + f\left((k - 3p) - (\delta_h + \sum_{i \in X} d_{j_{i+1}} - \delta)\right)/8 - 3 \end{aligned}$$

$$\begin{aligned}&\geq \delta + f\left(\delta_h + \sum_{i\in X\cup Y} d_{j_{i+1}} - \delta\right) + f\left((k-3p) - (\delta_h + \sum_{i\in X} d_{j_{i+1}} - \delta)\right)/8 - 3\\ &\geq \delta + f(k-3p)/8 - 3\\ &\geq \delta + f(k/4)/8 - 3 \geq \delta + f(k)/32 - 3.\end{aligned}$$

Here for the first inequality we use (2), then the properties of the function $f$, and the fact that we are in the case where $k > 4p$. Observe that for each $x \in \mathbb{Z}_+$, $f(x) \leq x$, since we are given an algorithm that finds an $(s, t)$-path of length $f(x)$ in any graph with the longest $(s, t)$-path of length $x$. With this, we have shown that in each case the returned $(s, t)$-path is of desired length, and the proof is complete. □

Finally, observe that the running time of Algorithm 3 is polynomial in the size of the given nested Erdős–Gallai decomposition. By Lemma 10, its size is polynomial in size of the input graph $G$. This concludes the proof of Theorem 2.

## 5 Approximation for Cycles

This section is devoted to the proof of Theorem 1 that establishes a way of lifting the approximation guarantee of the longest cycle in a 2-connected graph $G$ to the offset above $2\delta(G)$.

**Theorem 1** *Let $f : \mathbb{R}_+ \to \mathbb{R}_+$ 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 $L$.*

We first recall the concept of a Dirac decomposition and its properties that were established by Fomin et al. [26]. Further in this section, we prove the novel crucial property that in a graph admitting a Dirac decomposition, there exists a separating pair of vertices with a long path between this pair. Finally, we combine these results together with our approximation for $(s, t)$-path from the previous section (Theorem 2) to obtain the lifting algorithm in Theorem 1.

### 5.1 Dirac decomposition

This subsection contains the definition and properties of a Dirac decomposition, including the algorithmic result that allows to construct a Dirac decomposition from a given 2-connected graph $G$. We start with the definition of a Dirac decomposition, which can be seen as an analogue of an Erdős–Gallai decomposition for cycles.

**Definition 3** (Dirac decomposition and Dirac component, Definition 5 in [26]) Let $G$ be a 2-connected graph and let $C$ be a cycle in $G$ of length at least $2\delta(G)$. We say that two disjoint paths $P_1$ and $P_2$ in $G$ induce *a Dirac decomposition for $C$* in $G$ if

- The cycle $C$ is of the form $C = P_1P'P_2P''$, where each of the paths $P'$ and $P''$ has at least $\delta(G) - 2$ edges.

- For every connected component $H$ of $G - V(P_1 \cup P_2)$ holds $|V(H)| \geq 3$ and one of the following.

(D1) $H$ is 2-connected, 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;
(D2) $H$ 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 $P_2$;
(D3) The same as (D2), 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 $P_1$.

- There is exactly one connected component $H$ in $G - V(P_1 \cup P_2)$ with $V(H) = V(P')\setminus\{s', t'\}$, where $s'$ and $t'$ are the endpoints of $P'$. Analogously, there is exactly one connected component $H$ in $G - V(P_1 \cup P_2)$ with $V(H) = V(P'')\setminus\{s'', t''\}$.

The set of *Dirac component*s for a Dirac decomposition
is defined as follows. First, for each component $H$ of type (D1), $H$ is a Dirac component of the Dirac decomposition. Second, for each leaf-block of each $H$ of type (D2), or of type (D3), this leaf-block is also a Dirac component of the Dirac decomposition.

First, we recall an important property of a Dirac decomposition that restricts how a cycle can pass through a Dirac component.

**Lemma 13** (Lemma 17 in [26]) *Let $G$ be a 2-connected graph and $C$ be a cycle in $G$. Let paths $P_1, P_2$ induce a Dirac decomposition for $C$ in $G$. Let $M$ be a Dirac component of the Dirac decomposition and $P$ be a path in $G$ such that $P$ contains at least one vertex in $V(P_1) \cup V(P_2)$. If $P$ enters $M$, then all vertices of $M$ hit by $P$ appear consecutively on $P$.*

We now restate the result of [26] on the construction of a Dirac decomposition for a given graph. Note that here we state it in a slightly different form, which is more convenient in the setting of this paper. The statement is given below and the main difference is highlighted in bold.

**Lemma 14** (Lemma 20 in [26]) *Let $G$ be an $n$-vertex 2-connected graph and $k$ be an integer such that $\delta(G) \geq 12$, $0 < k \leq \frac{1}{24}\delta(G)$, and*

$$2k + 12 \leq \delta(G) < \boldsymbol{\frac{n}{2}}.$$

*Then there is an algorithm that, given a* ***non-hamiltonian*** *cycle $C$ of length less than $2\delta(G) + k$ in polynomial time finds either*

- *Longer cycle in $G$, or*
- *Vertex cover of $G$ of size at most $\delta(G) + 2k$, or*
- *Two paths $P_1, P_2$ that induce a Dirac decomposition for $C$ in $G$.*

To clarify this form, note that in the original statement of Lemma 20 in [26], the upper bound for $\delta(G-B)$ is $\frac{n}{2}-\frac{|B|+k}{2}$, where $B$ is a given set of small-degree vertices. In the proof of Lemma 20 in [26], one can easily note that the only reason for this bound is the existence of at least one vertex in $V(G)\setminus V(C)\setminus B$. Since in our work $B$ is always empty, this is equivalent to $V(G)\neq V(C)$, i.e. non-hamiltonicity of $C$. Hence, the replacement of this bound with the condition on non-hamiltonicity of $C$ is legitimate.

We finish this subsection with another important property of Dirac decomposition stating that there is a long cycle that enters at least one Dirac component. Unfortunately, this property, Lemma 19 in [26], is stated in a way requiring the offset above $2\delta(G)$ for this long cycle to be much smaller than $\delta(G)$. Here we provide this property in the form that does not require this and is much more convenient in our setting. Since it differs significantly from the original statement, we provide a proof of this result that is based on the proof of Lemma 19 from [26].

**Lemma 15** (Modified Lemma 19 from [26]) *Let $G$ be a graph and $P_1, P_2$ induce a Dirac decomposition for a cycle $C$ of length at most $2\delta(G-B)+\kappa$ in $G$ such that $2\kappa \leq \delta(G)$. If there exists a cycle of length at least $2\delta(G)+k$ in $G$ that contains at least one vertex in $V(P_1)\cup V(P_2)$, then there exists a cycle of length at least $2\delta(G)+k/2-1$ in $G$ that enters a Dirac component.*

***Proof*** Suppose that there exists a cycle $C'$ of length at least $2\delta(G)+k$ in $G$ that contains at least one vertex in $V(P_1)\cup V(P_2)$. If $C'$ already contains an edge of a Dirac component, we are done. We now assume that $C'$ does not contain any edge of any Dirac component. We show how to use $C'$ to construct a cycle of length at least $2\delta(G)+k/2-1$ in $G$ that contains an edge of a Dirac component of the given Dirac decomposition.

Let $W$ be the set of all vertices of $G$ that are vertices of non-leaf-blocks of (D2)-type or (D3)-type components in the Dirac decomposition. We start with the following claim.

**Claim 6** $|W\cap V(C')|>0$.

***Proof of Claim 6*** This is a counting argument. Note that $C'$ cannot contain an edge with both endpoints inside a Dirac component of $G$. Since Dirac components of $G$ are (D1)-type components of the Dirac decomposition and leaf-blocks of (D2)-type or (D3)-type connected components, each edge of $C'$ has an endpoint either in $V(P_1)\cup V(P_2)$, or inside a non-leaf-block of a (D2)-type or a (D3)-type connected component. The union of the vertex sets of the non-leaf-blocks forms the set $W$. Hence, $(W\cap V(C'))\cup V(P_1)\cup V(P_2)$ is a vertex cover of $C'$.

Note that a vertex cover of any cycle consists of at least half of its vertices. Then

$$2|(W\cap V(C'))\cup V(P_1)\cup V(P_2)|\geq |V(C')|.$$

By definition of a Dirac decomposition, $|V(P_1) \cup V(P_2)| \leq \kappa - 2$. Immediately we get that

$$2|W \cap V(C')| \geq 2\delta(G) + k - 2|V(P_1) \cup V(P_2)| \geq 2\delta(G) + k - 2(\kappa - 2) > 0.$$

□

We now take a vertex $w_1 \in W \cap V(C')$. The following claim allows constructing a long chord of $C'$ starting in $w_1$.

**Claim 7** *Let $H$ be a (D2)-type or a (D3)-type component in the Dirac decomposition. $C'$ does not contain any inner vertex of the leaf-blocks of $H$.*

***Proof of Claim 7*** Suppose that $C'$ contains some vertex $u \in V(H')$ that is an inner vertex of some leaf-block $L$ of $H$. As $L$ is a Dirac component of $G$, $C'$ cannot contain any edge of $L$, so $C'$ should enter $L$ from $V(P_1) \cup V(P_2)$ through $u$ and leave it immediately. By definition of Dirac decompositions, the only option to enter or leave $L$ is to go through the only vertex in $V(P_1)$ (if $H$ is of type (D2)) or in $V(P_2)$ (if $H$ is of type (D3)). As $C'$ cannot contain any vertex twice, this is not possible. □

Now construct the chord of $C'$ starting in $w_1$. Since $w_1$ is a vertex of a separable component $H$, there is a cut vertex $c_1$ of a leaf-block $L_1$ of $H$ reachable from $w_1$ inside $H$. The leaf-block $L_1$ contains also at least one vertex $v_1 \neq c_1$ that is adjacent to a vertex in $V(P_1)$ (if $H$ is of type (D2)) or to $V(P_2)$ (if $H$ is of type (D3)) outside $H$. We know that $\delta(L_1 - c_1) \geq \delta(G - c_1) - 1 \geq \delta(G) - 2$, since the only outside neighbour of vertices in $L_1 - c_1$ is a single vertex in $V(P_1)$ or $V(P_2)$. By Corollary 2, there exists an $(c_1, v_1)$-path inside $L_1$ of length at least $\delta(G) - 2$. Combine this with a $(w_1, c_1)$-path inside $H$ and obtain a $(w_1, v_1)$-path inside $H$.

Note that the constructed $(w_1, v_1)$-path can contain vertices from $V(C')$ apart from $w_1$. Let $w_1' \in V(C')$ be the vertex from $V(C')$ on the $(w_1, v_1)$-path farthest from $w_1$. Note that the $(w_1', v_1)$-subpath still contains the $(c_1, v_1)$-path as a subpath by Claim 7. Hence, we obtain a $(w_1', v_1)$-path of length at least $\delta(G) - 2$ inside $H$ that does not contain any vertex in $V(C') \setminus \{w_1'\}$. To obtain a long chord of $C'$, it is left to reach the vertex in $V(P_1) \cup V(P_2)$ from $v_1$ outside $H$, and then follow the cycle $C$ until a vertex $v_1'$ of $C'$ is reached. This is always possible since $V(C) \cap V(C') \supseteq (V(P_1) \cup V(P_2)) \cap V(C') \neq \emptyset$. We obtain a chord of length at least $\delta(G) - 1$ connecting $w_1'$ and $v_1'$.

The $(w_1', v_1')$-chord of $C'$ splits $C'$ into two $(w_1', v_1')$-arcs, and one of the arcs has length at least $\delta(G) + k/2$. Combine this arc with the chord and obtain a cycle of length at least $2\delta(G) + k/2 - 1$ in $G$. This cycle contains an edge of a leaf-block of $H$, i.e. of a Dirac component. The proof is complete. □

## 5.2 Existence of a Separating Pair

This subsection encapsulates the new combinatorial result behind Dirac decomposition that is crucial to our proof of Theorem 1. It helps us avoid using the Dirac decomposition explicitly in our algorithm, so that we can instead reduce to the algorithm for approximating $(s, t)$-paths. The formal statement is recalled next.

**Lemma 1** *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 $2\kappa \le \delta(G)$. 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 $G$.*

***Proof*** Consider the longest cycle $C'$ in $G$. We assume that this cycle is of length at least $2\delta(G)+k$ and consider four cases.

**Case 1.** *$C'$ is completely contained in some connected component $H$ of $G - V(P_1 \cup P_2)$, and $H$ is 2-connected.* Then $H$ is a Dirac component of type (D1). Since the matching size between $V(H)$ and $V(P_i)$ for each $i \in \{1, 2\}$ is exactly one, by Kőnig's theorem all edges between $V(H)$ and $V(P_i)$ are covered by a single vertex. Denote this vertex by $u$ for $i = 1$ and by $v$ for $i = 2$. Since $G$ is 2-connected, $u$ and $v$ are distinct. As $\{u, v\}$ separates $H$ from the rest of the graph and $V(C) \not\subseteq V(H \cup P_1 \cup P_2)$, we have that $G - \{u, v\}$ is not connected. It is left to show that there exists a long $(u, v)$-path in $G$. Toward this, denote $u' = u$ if $u \in V(H)$, and $u' \in N_G(u) \cap V(H)$ if $u \in V(P_1)$. Choose $v' \in V(H)$ similarly, i.e. $v'$ either equals $v$ or is a neighbour of $v$. Since $G$ is 2-connected, there is always a way to choose distinct $u'$ and $v'$. By Lemma 2, there is a $(u', v')$-path of length at least $\delta(G) + k/2$ in $H$, hence there is also a $(u, v)$-path of length at least $\delta(G) + k/2$ in $G$.

**Case 2.** *$C'$ is completely contained in a leaf-block of a connected component $H$ of $G - V(P_1 \cup P_2)$.* That is, $C'$ is contained in a Dirac component of type (D2) or (D3). The choice of $u$ and $v$ is similar to Case 1. That is, if the Dirac component is of type (D2), choose $u$ such that $u \in N_G(V(H)) \cap V(P_1)$ and choose $v$ equal to the cut vertex of the Dirac component. $G - \{u, v\}$ is not connected as $\{u, v\}$ separates the Dirac component from the rest of $H$. There is an $(u, v)$-path of length at least $\delta(G) + k/2 + 1$ since there exists a $(z, v)$-path of length at least $\delta(G) + k/2$ by Lemma 2, where $z \in N_G(u) \cap V(H - v)$. The choice of $u$ and $v$ for type (D3) is symmetrical.

**Case 3.** *$C'$ is completely contained in a non-leaf-block of a connected component $H$ of $G - V(P_1 \cup P_2)$.* In this case, $C'$ is not contained in a Dirac component. Denote the non-leaf-block of $H$ that contains $C'$ by $K$. Without loss of generality, we assume that $H$ corresponds to (D2), i.e. $|N_G(V(H)) \cap V(P_1)| = 1$. By Menger's theorem, there are either two vertices separating $V(K)$ from $V(P_1 \cup P_2)$ in $G$ or three disjoint paths going from $V(K)$ to $V(P_1 \cup P_2)$. If the former is the case, denote these two vertices by $u$ and $v$. Obviously, $G - \{u, v\}$ is not connected. There are two disjoint paths going from $V(K)$ to $V(P_1 \cup P_2)$, and one of these paths contains $u$ and the other contains $v$. Connect the endpoints of these paths in $V(K)$ using a path of length at least $\delta(G) + k/2$ inside $V(K)$ given by Lemma 2. Clearly, the obtained long path contains an $(u, v)$-path of length at least $\delta(G) + k/2$ as a subpath.

If the latter is the case, then two of the three paths necessarily end in $V(P_2)$. Let these two paths start respectively from $u_1$ and $u_2$ in $V(K)$ and end in $v_1$ and $v_2$ in $V(P_2)$. Note that these paths use only edges of $H$ and edges between $V(H)$ and $V(P_2)$ in $G$. Moreover, none of the paths has an internal vertex in $V(K)$ or $V(P_2)$. By Lemma 2, there is an $(u_1, u_2)$-path of length at least $\delta(G)+k/2$ in $K$. Now construct a cycle in $G$ by combining the $(u_1, u_2)$-path with the $(u_1, v_1)$-path, $(u_2, v_2)$-path, and the subpath of $C$ that goes between $v_1$ and $v_2$ outside of $P_2$. Since that subpath contains $P'$ and $P''$ from the definition of Dirac decomposition as subpaths, the obtained cycle is of length at least $(\delta(G) + k/2) + 1 + 1 + 2 \cdot (\delta(G) - 2) \geq 3\delta(G) + k/2 - 2 \geq 2\delta(G) + k/2$. This cycle contains an edge of $P'$, so it enters a Dirac component, so we can replace $C'$ with this cycle and apply the following case.

**Case 4.** *$C'$ has a common vertex with $V(P_1 \cup P_2)$. By* Lemma 15, *we can assume that $C'$ enters a* Dirac component *$K$ but its length is at least* $2\delta(G) + k/2 - 1$. Following Case 1 and Case 2, we know that there are $u, v \in V(K \cup P_1 \cup P_2)$ such that in $G - \{u, v\}$ vertices in $V(K)$ are separated from the rest of the graph. By Lemma 13, vertices in $V(K)$ appear consecutively on $C'$, so vertices and edges of $C'$ induce a path inside $K$. Since $C'$ is not contained in $V(K)$, at least one of $u$ and $v$ is present in $C'$, so we have two cases depending on $|V(C') \cap \{u, v\}|$. If $u, v \in V(C')$, then the longest arc of $C'$ going between $u$ and $v$ simply yields a path of length at least $(2\delta(G) + k/2 - 1)/2 = \delta(G) + (k - 2)/4$. If exactly one of $u$ and $v$ is present on $C'$, without loss of generality we assume $u \in V(C')$. Then $V(C') \subseteq V(K) \cup \{u\}$, as $C'$ does not pass through $v$—the only other entry to $K$. Similarly to Case 1 and Case 2, we have a vertex $v' \in V(K)$, which is either equal to $v$ or is a neighbour of $v$. Take a shortest path from $v'$ to $V(C')$ inside $K$. Denote its endpoint by $w$. Prolong the path starting in $v'$ with the longest arc of $C'$ that goes between $w$ and $u$. This yields a $(v', u)$-path, hence a $(v, u)$-path, of length at least $\delta(G) + (k - 2)/4$ in $G$. □

### 5.3 Proof of Theorem 1

In this subsection, we combine Theorem 2 and the results presented earlier in this section into the proof of Theorem 1.

***Proof of Theorem 1*** Assume that we are given a blackbox algorithm that finds a cycle of length $f(L)$ in a graph with the longest cycle length $L$. We now describe the desired approximation algorithm that finds a cycle of length at least $2\delta(G) + h(k)$ based on the blackbox algorithm, where

$$h(k) = \frac{1}{128} f(k) - 8.$$

The input to our algorithm is a graph $G$, let $L$ be the length of the longest cycle in $G$ and $k = L - 2\delta$. For convenience, denote $\delta := \delta(G)$. The goal of our algorithm is to find a cycle of length at least $2\delta + h(k)$ in $G$. Note that the algorithm does not estimate $h(k)$ in any way, it merely outputs the longest cycle that was found during its run. We focuse on showing that this cycle always has length at least $2\delta + h(k)$.

The pseudocode of our algorithm is presented in Algorithm 4. The first few lines of the algrotihm are dedicated to eliminating various corner cases where either the blackbox approximation suffices directly, or a long Dirac cycle. This will help us avoid dealing with extreme parameter values later in the analysis.

If $2\delta \geq n$, the algorithm will find and output a Hamiltonian cycle in $G$ following Dirac's theorem on Line 4.5. For the rest of the analysis, we assume $2\delta < n$. On Line 4.1 our algorithm applies the blackbox $f(L)$-approximation algorithm to $G$.

If $f(L) \geq \frac{49}{24}\delta$, then the resulting cycle is of length at least $2\delta + (f(L) - 2\delta) \geq 2\delta + \frac{1}{49}f(L)$, which is at least $2\delta + h(k)$. As the algorithm never makes the current cycle shorter, in this case the output will be automatically valid. We now also assume that $f(L) < \frac{49}{24}\delta$.

```
longest_cycle_above_degree_approx($G$)
  Input: 2-connected graph $G$ of minimum degree $\delta$
  Output: a cycle $C$ of length at least $2\delta + h(k)$, where $k = L - 2\delta$ and $L$ is the length of the
          longest cycle in $G$
4.1  $C \longleftarrow$ longest_cycle_approx($G$);
4.2  if long_dirac_cycle($G, \emptyset, 1$) is YES then
4.3      $C \longleftarrow$ the longest of $C$ and the computed cycle of length at least $2\delta + 1$ in $G$;
4.4  else
4.5      return the cycle of length $2\delta$ in $G$;
4.6  end
4.7  if $\delta \leq 24$ then
4.8      return $C$;
4.9  end
4.10 while $|V(C)| - 2\delta < \lfloor\frac{1}{24}\delta\rfloor$ and Lemma 14 applied to G and C gives a longer cycle do
4.11     $C \longleftarrow$ a longer cycle in $G$;
4.12 end
4.13 if $|V(C)| \geq \lfloor 2\frac{1}{24}\delta\rfloor$ or Lemma 14 gives the vertex cover of G then
4.14     return $C$;
4.15 end
4.16 foreach $u, v \in V(G)$ such that $G - \{u, v\}$ is not connected do
4.17     $Q, R \longleftarrow$ empty paths;
4.18     foreach connected component $H$ in $G - \{u, v\}$ do
4.19         $S \longleftarrow$ longest_st_path_above_degree_approx($G[V(H) \cup \{u, v\}] + uv, u, v$);
4.20         $Q, R \longleftarrow$ two longest paths among $Q, R, S$;
4.21     end
4.22     $C \longleftarrow$ the longest of $C$, $Q \cup R$;
4.23 end
4.24 return $C$;
```

**Algorithm 4:** The algorithm finding a cycle of length at least $2\delta(G) + h(k)$ in a 2-connected graph $G$.

On Line 4.2 our algorithm applies the FPT algorithm for LONG DIRAC CYCLE to find a cycle of length at least $2\delta + 1$ in $G$ in polynomial time. If such cycle is found, then our algorithm keeps the longest of this cycle and previously computed approximation. If $h(k) = \frac{1}{128}f(k) - 8 \leq 1$, then this cycle is a required approximation. On the other hand, if a cycle of length at least $2\delta + 1$ does not exist in $G$, then $k = L - 2\delta = 0$, so a cycle of length $2\delta$ is a valid approximation. Hence, in this case our algorithm just outputs a cycle of length at least $2\delta$ guaranteed by Dirac's theorem in this case and stops.

We now can assume that $f(k) \geq 9 \cdot 128$. Since $f(L) < \frac{49}{24}\delta$, it follows that $\delta > 24$. Thus, on Line 4.7, if $\delta \leq 24$, our algorithm just stops as the required approximation cycle was already encountered by the algorithm.

Now we reach the main case of the algorithm, where we use the structural results on Dirac decomposition to find a long cycle. Before Line 4.9, the current cycle $C$ has length at least $2\delta$. If the length of $C$ is less than $\lfloor 2\frac{1}{24}\delta \rfloor$, then the algorithm of Lemma 14 is applied to the graph $G$ and the cycle $C$ with the parameter $k' = |V(C)| - 2\delta + 1$. If the outcome is a cycle longer than $C$, then it replaces $C$ with this cycle. If it still holds that $|V(C)| < \lfloor 2\frac{1}{24}\delta \rfloor$, then our algorithm applies Lemma 14 to $G$ and $C$ again. This process repeats until one of the three possible structures are found in $G$:

- Cycle $C$ of length at least $\lfloor 2\frac{1}{24}\delta \rfloor$;
- Vertex cover of size at most $\delta + 2(|V(C)| - 2\delta + 1)$;
- Two paths $P_1, P_2$ that induce a Dirac decomposition for $C$.

The first outcome is the desired $h(k)$-approximation of the offset since $|V(C)| \geq 2\delta + \frac{1}{49} f(L) - 1 \geq 2\delta + h(k)$ in this case. In the second outcome, since a vertex cover upper-bounds the length of any cycle, $L \leq 2 \cdot (2 \cdot |V(C)| - 3\delta + 2)$, hence $|V(C)| - 2\delta \geq \frac{L-4-2\delta}{4}$, so $|V(C)| \geq 2\delta + \frac{k}{4} - 1$. Automatically, $C$ is a valid approximation in this case as well. Thus if any of the two situations occur, our algorithm simply returns the current cycle $C$.

We move on to the most involved case where the two paths $P_1, P_2$ inducing a Dirac decomposition are found. Our goal now is to use Lemma 1 to find a separating pair of vertices that has a long path between them, and then use the already established algorithm from Theorem 2 to approximate the length of this path.

Hence, before moving further, we need to obtain an approximation algorithm for finding a long $(s, t)$-path. For this, we apply Lemma 3 to the blackbox $f(L)$-approximation algorithm for the longest cycle and obtain an algorithm that finds an $(s, t)$-path of length at least $\frac{1}{2} f(2p)$ in a graph with the longest $(s, t)$-path lenght $p$. Finally, we apply Theorem 2 to the latter algorithm and obtain an algorithm finding an $(s, t)$-path of length $\delta + (\frac{1}{64} f(2k') - 3)$ where $k' = p - \delta$ for the longest $(s, t)$-path length $p$.

Since $2(|V(C)| - 2\delta) < \delta$, we can apply Lemma 1 to $G$ and $C$. We obtain that there exists a pair of vertices $u, v \in V(G)$ such that $G - \{u, v\}$ is not connected and there is a path of length at least $\delta + (L - 2\delta - 2)/4$ between $u$ and $v$ in $G$. Towards encountering this pair of vertices, our algorithm iterates over all possible $u, v \in V(G)$ such that $G - \{u, v\}$ is not connected. Assume that the pair $\{u, v\}$ is fixed. Then for each connected component $H$ of $G - \{u, v\}$, our algorithm applies the $(s, t)$-path approximation algorithm to $G[V(H) \cup \{u, v\}] + uv$ to find a long $(u, v)$-path. Note that this is a legitimate application of the algorithm since $G[V(H) \cup \{u, v\}] + uv$ is 2-connected.

Now, each application of the algorithm yields some $(u, v)$-path. There are at least two connected components in $G - \{u, v\}$, so at least two $(u, v)$-paths are produced, and our algorithm simply combines the longest two paths among them in a cycle. The length of each path is at least $\delta - 2$. Moreover, if $\{u, v\}$ is the pair given by Lemma 1, for at least one component the longest $(u, v)$-path has length at least $\delta + (k - 2)/4$. Then for this connected component $H$ the approximation algorithm finds an $(u, v)$-path $Q$ of length at least $\delta(H - \{u, v\}) + \frac{1}{64} f(2k') - 3$, where

$$k' = \delta(G) + (k - 2)/4 - \delta(H - \{u, v\}).$$

Denote $x = (\delta(G) - \delta(H - \{u, v\})$. Note that $x \leq 2$ and can be negative. Then the length of $Q$ is at least

$$\delta(G) - x + \frac{1}{64} f(2(x + (k-2)/4)) - 3.$$

If $x \geq 0$, then the length of $Q$ is at least

$$\delta(G) - 2 + \frac{1}{64} f((k-2)/2) - 3.$$

If $x < 0$, then, as $-x \geq f(|x|)$, the length of $Q$ is at least

$$\begin{aligned} &\delta(G) + f(|x|) + \frac{1}{64} f(-2|x| + (k-2)/2) - 3 \\ &\quad\geq \delta(G) + \frac{1}{64} f(64|x|) + \frac{1}{64} f(-2|x| + (k-2)/2) - 3 \\ &\quad\geq \delta(G) + \frac{1}{64} f(62|x| + (k-2)/2) - 3 \\ &\quad\geq \delta(G) + \frac{1}{64} f((k-2)/2) - 3. \end{aligned}$$

In any case, the length of $Q$ is at least

$$\begin{aligned} &\delta + \frac{1}{64} f(k/2 - 1) - 5 \geq \delta + \frac{1}{128} f(k-2) - 5 \\ &\geq \delta + \frac{1}{128} f(k) - \frac{1}{128} f(2) - 5 \geq \delta + \frac{1}{128} f(k) - 6. \end{aligned}$$

Hence, if our algorithm combines the longest two paths given by the pair $\{u, v\}$, it obtains a cycle of length at least $2\delta + \frac{1}{128} f(k) - 8 = 2\delta + h(k)$. Since the algorithm outputs the longest cycle among those constructed, the proof is complete. □

Finally, we show the corollary of Theorem 1 that allows to approximate the longest path in a graph that is not necessarily 2-connected.

**Corollary 3** *Let $f : \mathbb{R}_+ \to \mathbb{R}$ be a non-decreasing subadditive function. If there exists a polynomial-time algorithm finding a cycle of length $f(L)$ in a 2-connected graph with the longest cycle length L, there is a polynomial time algorithm that outputs a path of length at least $2\delta(G) + f(L - 2\delta(G))/128 - 8$ in a graph G with $\delta(G) < \frac{1}{2}|V(G)|$ and the longest path length L.*

***Proof of Corollary 3*** Take a graph $G$. We assume that $G$ is connected, otherwise we apply the same algorithm to each of its connected components.

Add a universal vertex to $G$ and obtain a 2-connected graph $G'$. Note that $\delta(G') = \delta(G) + 1$ and there is a cycle of length at least $c$ in $G'$ if and only if there is a path of length at least $c - 2$ in $G$. Hence, $c - 2\delta(G') = c - 2\delta(G) - 2 = p - 2\delta(G)$ for the longest cycle length $c$ in $G'$ and the longest path length $p$ in $G$. Moreover, a cycle of length $c$ in $G'$ can be transformed into a path of length at least $c - 2$ in $G$.

Apply Theorem 1 to $G'$. The obtained cycle is of length at least $2\delta(G')+f(k)/128-8$ where $k = c - 2\delta(G')$ for the longest cycle length $c$ in $G'$. Finally, transform this cycle into a path of length at least $2\delta(G) + f(k)/128 - 8$ in $G$. □

## 6 Conclusion

In this article, we have shown a general theorem that allows 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 [30]. 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* $\mathrm{dg}(G)$ 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 [24] that a cycle of length at least $L = \mathrm{dg}(G) + k$ 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 $\mathrm{dg}(G) + \Omega(\log(L - \mathrm{dg}(G)))$. 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 othere words, they gave an algorithm that finds an $(s, t)$-path of length at least $L = \mathrm{dist}_G(s, t) + k$ in $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time. Here $\mathrm{dist}_G(s, t)$ is the distance between $s$ and $t$. Therefore, in undirected graph we can find an $(s, t)$-path of length $\mathrm{dist}_G(s, t) + \Omega(\log(L - \mathrm{dist}_G(s, t))$ 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 on the (in)approximability of long detours in directed graphs.

**Author Contributions** All authors contributed equally to this work.

**Funding** Open access funding provided by University of Bergen (incl Haukeland University Hospital) Supported by the Research Council of Norway via the project BWCA (grant no. 314528) and DFG Research Group ADYN via Grant DFG 411362735.

## Declarations

**Conflict of interest** The authors declare no conflict of interest.

## References

1. Alon, N., Yuster, R., Zwick, U.: Color-coding. J. ACM **42**(4), 844–856 (1995). https://doi.org/10.1145/210332.210337
2. Alon, N., Gutin, G., Kim, E.J., et al.: Solving MAX-$r$-SAT above a tight lower bound. In: Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 511–517. SIAM (2010)
3. Bazgan, C., Santha, M., Tuza, Z.: On the approximation of finding a(nother) Hamiltonian cycle in cubic Hamiltonian graphs. J. Algorithms **31**(1), 249–268 (1999). https://doi.org/10.1006/jagm.1998.0998
4. Bezáková, I., Curticapean, R., Dell, H., et al.: Finding detours is fixed-parameter tractable. SIAM J. Discrete Math. **33**(4), 2326–2345 (2019). https://doi.org/10.1137/17M1148566
5. Björklund, A.: Determinant sums for undirected Hamiltonicity. SIAM J. Comput. **43**(1), 280–299 (2014). https://doi.org/10.1137/110839229
6. Björklund, A., Husfeldt, T.: Finding a path of superlogarithmic length. SIAM J. Comput. **32**(6), 1395–1402 (2003). https://doi.org/10.1137/S0097539702416761
7. Björklund, A., Husfeldt, T., Khanna, S.: Approximating longest directed paths and cycles. In: Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP), Lecture Notes in Computer Science, vol. 3142, pp. 222–233. Springer, Berlin (2004). https://doi.org/10.1007/978-3-540-27836-8_21
8. Björklund, A., Husfeldt, T., Kaski, P., et al.: Narrow sieves for parameterized paths and packings. CoRR arxiv:abs/1007.1161 (2010)
9. Bodlaender, H.L.: On linear time minor tests with depth-first search. J. Algorithms **14**(1), 1–23 (1993)
10. Bollobás, B.: Extremal graph theory. In: Handbook of Combinatorics, vol. 1, 2, pp. 1231–1292. Elsevier, Amsterdam (1995)
11. Bollobás, B., Scott, A.D.: Better bounds for Max Cut. In: Contemporary Combinatorics, Bolyai Society Mathematical Studies, vol. 10, pp. 185–246. János Bolyai Mathematical Society, Budapest (2002)
12. Bondy, J.A.: Basic graph theory: paths and circuits. In: Handbook of Combinatorics, vol. 1, 2, pp. 3–110. Elsevier, Amsterdam (1995)
13. Chen, G., Gao, Z., Yu, X., et al.: Approximating longest cycles in graphs with bounded degrees. SIAM J. Comput. **36**(3), 635–656 (2006)
14. Crowston, R., Jones, M., Muciaccia, G., et al.: 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), Leibniz International Proceedings in Informatics (LIPIcs), vol. 24, pp. 43–54. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl (2013)
15. Cygan, M., Fomin, F.V., Kowalik, Ł, et al.: Parameterized Algorithms. Springer, Berlin (2015)
16. Diestel, R.: Graph Theory, Graduate Texts in Mathematics, vol. 173, 5th edn. Springer, Berlin (2017)

17. Dirac, G.A.: Some theorems on abstract graphs. Proc. Lond. Math. Soc. **3**(2), 69–81 (1952)
18. Edwards, C.S.: Some extremal properties of bipartite subgraphs. Can. J. Math. **3**, 475–485 (1973)
19. Erdős, P., Gallai, T.: On maximal paths and circuits of graphs. Acta Math. Acad. Sci. Hungar. **10**, 337–356 (1959)
20. Feder, T., Motwani, R.: Finding large cycles in Hamiltonian graphs. Discrete Appl. Math. **158**(8), 882–893 (2010). https://doi.org/10.1016/j.dam.2009.12.006
21. Feder, T., Motwani, R., Subi, C.: Approximating the longest cycle problem in sparse graphs. SIAM J. Comput. **31**(5), 1596–1607 (2002). https://doi.org/10.1137/S0097539701395486
22. Fomin, F.V., Kaski, P.: Exact exponential algorithms. Commun. ACM **56**(3), 80–88 (2013). https://doi.org/10.1145/2428556.2428575
23. Fomin, F.V., Lokshtanov, D., Panolan, F., et al.: Efficient computation of representative families with applications in parameterized and exact algorithms. J. ACM **63**(4), 29:1-29:60 (2016). https://doi.org/10.1145/2886094
24. Fomin, F.V., Golovach, P.A., Lokshtanov, D., et al.: Going far from degeneracy. SIAM J. Discrete Math. **34**(3), 1587–1601 (2020). https://doi.org/10.1137/19M1290577
25. Fomin, F.V., Golovach, P.A., Lokshtanov, D., et al.: Multiplicative parameterization above a guarantee. ACM Trans. Comput. Theory **13**(3), 18:1-18:16 (2021). https://doi.org/10.1145/3460956
26. Fomin, F.V., Golovach, P.A., Sagunov, D., et al.: Algorithmic extensions of Dirac's theorem. In: Proceedings of the 2022 Annual ACM–SIAM Symposium on Discrete Algorithms (SODA), pp. 406–416 (2022). https://doi.org/10.1137/1.9781611977073.20
27. Fomin, F.V., Golovach, P.A., Sagunov, D., et al.: Longest cycle above Erdős–Gallai bound. In: 30th Annual European Symposium on Algorithms, ESA 2022, September 5–9, 2022, Berlin/Potsdam, Germany, LIPIcs, vol. 244. Schloss Dagstuhl—Leibniz-Zentrum für Informatik, pp. 55:1–55:15 (2022). https://doi.org/10.4230/LIPIcs.ESA.2022.55
28. Fürer, M., Raghavachari, B.: Approximating the minimum-degree Steiner tree to within one of optimal. J. Algorithms **17**(3), 409–423 (1994). https://doi.org/10.1006/jagm.1994.1042
29. Gabow, H.N.: Finding paths and cycles of superpolylogarithmic length. SIAM J. Comput. **36**(6), 1648–1671 (2007). https://doi.org/10.1137/S0097539704445366
30. Gabow, H.N., Nie, S.: Finding a long directed cycle. ACM Trans. Algorithms **4**(1), 66 (2008)
31. Gabow, H.N., Nie, S.: Finding long paths, cycles and circuits. In: Proceedings of the 19th International Symposium on Algorithms and Computation (ISAAC), Lecture Notes in Computer Science, vol. 5369, pp. 752–763. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-92182-0_66
32. Garg, S., Philip, G.: 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), pp. 1152–1166. SIAM (2016). https://doi.org/10.1137/1.9781611974331.ch80
33. Gutin, G., Kim, E.J., Lampis, M., et al.: Vertex cover problem parameterized above and below tight bounds. Theory Comput. Syst. **48**(2), 402–410 (2011)
34. Gutin, G., van Iersel, L., Mnich, M., et al.: Every ternary permutation constraint satisfaction problem parameterized above average has a kernel with a quadratic number of variables. J. Comput. Syst. Sci. **78**(1), 151–163 (2012)
35. Gutin, G.Z., Mnich, M.: A survey on graph problems parameterized above and below guaranteed values. https://doi.org/10.48550/arXiv.2207.12278, CoRR arxiv:2207.12278 (2022)
36. Gutin, G.Z., Patel, V.: Parameterized traveling salesman problem: beating the average. SIAM J. Discrete Math. **30**(1), 220–238 (2016)
37. Gutin, G.Z., Rafiey, A., Szeider, S., et al.: The linear arrangement problem parameterized above guaranteed value. Theory Comput. Syst. **41**(3), 521–538 (2007). https://doi.org/10.1007/s00224-007-1330-6
38. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity. J. Comput. Syst. Sci. **63**(4), 512–530 (2001)
39. Jansen, B.M.P., Kozma, L., Nederlof, J.: Hamiltonicity below Dirac's condition. In: Proceedings of the 45th International Workshop on Graph-Theoretic Concepts in Computer Science (WG), Lecture Notes in Computer Science, vol. 11789, pp. 27–39. Springer, Berlin (2019)
40. Karger, D.R., Motwani, R., Ramkumar, G.D.S.: On approximating the longest path in a graph. Algorithmica **18**(1), 82–98 (1997). https://doi.org/10.1007/BF02523689
41. Kleinberg, J.M., Tardos, É.: Algorithm Design. Addison-Wesley, Englewood Cliffs (2006)

42. Koutis, I.: Faster algebraic algorithms for path and packing problems. In: Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP), Lecture Notes in Computer Science, vol. 5125, pp. 575–586. Springer, Berlin (2008)
43. Koutis, I., Williams, R.: Algebraic fingerprints for faster algorithms. Commun. ACM **59**(1), 98–105 (2016). https://doi.org/10.1145/2742544
44. Lokshtanov, D., Narayanaswamy, N.S., Raman, V., et al.: Faster parameterized algorithms using linear programming. ACM Trans. Algorithms **11**(2), 15:1-15:31 (2014). https://doi.org/10.1145/2566616
45. Mahajan, M., Raman, V., Sikdar, S.: Parameterizing above or below guaranteed values. J. Comput. Syst. Sci. **75**(2), 137–153 (2009)
46. Mishra, S., Raman, V., Saurabh, S., et al.: The complexity of König subgraph problems and above-guarantee vertex cover. Algorithmica **61**(4), 857–881 (2011). https://doi.org/10.1007/s00453-010-9412-2
47. Monien, B.: How to find long paths efficiently. In: Analysis and Design of Algorithms for Combinatorial Problems (Udine, 1982), North-Holland Mathematical Studies, vol. 109, pp. 239–254. North-Holland, Amsterdam (1985). https://doi.org/10.1016/S0304-0208(08)73110-4
48. Robertson, N., Seymour, P.D.: Graph minors. xiii. The disjoint paths problem. J. Comb. Theory Ser. B **63**(1), 65–110 (1995). https://doi.org/10.1006/jctb.1995.1006
49. Vishwanathan, S.: An approximation algorithm for finding long paths in Hamiltonian graphs. J. Algorithms **50**(2), 246–256 (2004). https://doi.org/10.1016/S0196-6774(03)00093-2
50. Williams, R.: Finding paths of length $k$ in $O^*(2^k)$ time. Inf. Process. Lett. **109**(6), 315–318 (2009)