Parameterization Above a Multiplicative Guarantee

Fedor V. Fomin
Department of Informatics, University of Bergen, Norway
fomin@ii.uib.no

Petr A. Golovach
Department of Informatics, University of Bergen, Norway
petr.golovach@uib.no

Daniel Lokshtanov
University of California, Santa Barbara, USA
daniello@ucsb.edu

Fahad Panolan
Indian Institute of Technology Hyderabad, India
fahad@iith.ac.in

Saket Saurabh
Department of Informatics, University of Bergen, Norway
The Institute of Mathematical Sciences, HBNI and IRL 2000 ReLaX, Chennai, India
saket@imsc.res.in

Meirav Zehavi
Ben-Gurion University of the Negev, Beersheba, Israel
meiravze@bgu.ac.il

Abstract

Parameterization above a guarantee is a successful paradigm in Parameterized Complexity. To the best of our knowledge, all fixed-parameter tractable problems in this paradigm share an *additive form* defined as follows. Given an instance (I, k) of some (parameterized) problem Π with a *guarantee* $g(I)$, decide whether I admits a solution of size at least (at most) $k + g(I)$. Here, $g(I)$ is usually a lower bound (resp. upper bound) on the maximum (resp. minimum) size of a solution. Since its introduction in 1999 for Max SAT and Max Cut (with $g(I)$ being half the number of clauses and half the number of edges, respectively, in the input), analysis of parameterization above a guarantee has become a very active and fruitful topic of research.

We highlight a *multiplicative* form of parameterization above a guarantee: Given an instance (I, k) of some (parameterized) problem Π with a guarantee $g(I)$, decide whether I admits a solution of size at least (resp. at most) $k \cdot g(I)$. In particular, we study the Long Cycle problem with a multiplicative parameterization above the girth $g(I)$ of the input graph, and provide a parameterized algorithm for this problem. Apart from being of independent interest, this exemplifies how parameterization above a multiplicative guarantee can arise naturally. We also show that, for any fixed constant $\epsilon > 0$, multiplicative parameterization above $g(I)^{1+\epsilon}$ of Long Cycle yields para-NP-hardness, thus our parameterization is tight in this sense. We complement our main result with the design (or refutation of the existence) of algorithms for other problems parameterized multiplicatively above girth.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases Parameterized Complexity, Above-Guarantee Parameterization, Girth

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.39

11th Innovations in Theoretical Computer Science Conference (ITCS 2020).
Editor: Thomas Vidick; Article No. 39; pp. 39:1–39:13

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Funding *Fedor V. Fomin*: Research Council of Norway via the project MULTIVAL.
Petr A. Golovach: Research Council of Norway via the project MULTIVAL.
Daniel Lokshtanov: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 715744), and United States - Israel Binational Science Foundation grant no. 2018302.
Saket Saurabh: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 819416), and Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.
Meirav Zehavi: Israel Science Foundation grant no. 1176/18, and United States – Israel Binational Science Foundation grant no. 2018302.

1 Introduction

The goal of parameterized complexity is to find ways of solving NP-hard problems more efficiently than brute force: our aim is to restrict the combinatorial explosion to a parameter that is hopefully much smaller than the input size. Formally, a *parameterization* of a problem is the assignment of an integer k to each input instance, and we say that a parameterized problem is *fixed-parameter tractable (FPT)* if there is an algorithm that solves the problem in time $f(k) \cdot n^{O(1)}$, where n is the size of the input and f is an arbitrary computable function depending on the parameter k only. There is a long list of NP-hard problems that are FPT under various parameterizations: finding a vertex cover of size k, finding a cycle of length k, finding a maximum independent set in a graph of treewidth at most k, etc. For more background, the reader is referred to the monographs [12, 14, 21].

Choosing a suitable parameter plays an important role in the field of parameterized complexity. Traditionally, the *solution size* has been the most sought after parameter. However, in various circumstances this is not a good parameter. To illustrate this, consider the following problems: Max Sat and Max Cut. Observe that there always exists a truth assignment that satisfies half of the clauses, and there is always a max-cut containing at least half the edges. Thus, if we choose solution size as the parameter, then we get the following trivial algorithm: if $k \leq m/2$, then return yes; else $m \leq 2k$, so now any brute-force algorithm is an FPT algorithm. Thus if we want to design a nontrivial parameterized algorithm, then solution size is *not a suitable parameter*. In particular, a general message here is as follows.

> The natural parameterization of, say, 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.

Thus, for such cases, it is more natural to parameterize the problem by *the difference between the solution size and the bound*. This perspective is known as "above guarantee" parameterization. This approach was introduced by Mahajan and Raman [33] for the Max Sat and Max Cut problem. This approach was successfully applied to many various problems (see, e.g., [1, 11, 24, 25, 26, 34, 10] for a few illustrative examples).

In some of the above examples there is an explicit lower bound on the solution size, given in terms of the input size. However, in many cases, we do not have explicit lower bounds. We substantiate this with the example of classic Vertex Cover problem. In this problem, we are given a graph G and a positive integer k, and the goal is to test whether there exists a set of vertices C of size at most k that is a vertex cover (i.e., every edge has at least one endpoint in C). Clearly, the size of a *maximum matching* of G or the value of the linear programming relaxation of an integer linear program for Vertex Cover is a lower bound on the size of a vertex cover of G. Observe that these lower bounds are graph dependent

and not the size dependent. This is what we mean by implicit lower bounds. Coming back to Vertex Cover, in polynomial time it is possible to reduce the size of the graph, without changing the answer, such that we are guaranteed that any vertex cover must contain half of the remaining vertices. Thus, again we can employ any brute-force algorithm and design an FPT algorithm for Vertex Cover. This has led to the study of the problem Vertex Cover Above LP (or Vertex Cover Above Matching), which has played a central role in the development of the field of parameterized complexity [12].

In this paper, we take the philosophy of above guarantee parameterization a significant conceptual step forwards. In particular,

> The goal of this paper is to study another classical problem – namely, Long Cycle – from the viewpoint of a new implicit parameterization, that is neither standard nor an additive above guarantee parameterization.

The Long Path problem is to decide, given a directed or undirected n-vertex graph G and an integer k, whether G contains a path on at least k vertices, that is, a self-avoiding walk with at least k vertices. Similarly, the Long Cycle problem is to decide whether G contains a cycle of length at least k, that is, a closed self-avoiding walk with at least k vertices. These problems are natural generalization of the classical Hamiltonian Path and Hamiltonian Cycle problems and have been actively studied. In particular, there is a plethora of results about parameterized complexity of Long Path and Long Cycle (see, e.g., [4, 5, 9, 8, 18, 22, 29, 30, 31, 38]) since the early work of Monien [35]. Let us just mention here that the fastest known randomized algorithm for Long Path is due to Björklund et al. [4] and runs in time $1.657^k \cdot n^{\mathcal{O}(1)}$, whereas the fastest known deterministic algorithm is due to Tsur [37] and runs in time $2.554^k \cdot n^{\mathcal{O}(1)}$. Respectively for Long Cycle, the randomized algorithm with the currently best running time of $4^k \cdot n^{\mathcal{O}(1)}$ was given by Zehavi in [40], and the best deterministic algorithm was given by Fomin et al. in [20] and runs in time $4.884^k \cdot n^{\mathcal{O}(1)}$.

For Long Path, the investigation of above guarantee parameterizations was initiated by Bezáková et al. in [2]. Let s and t be two vertices of a graph G. Clearly, the length of any (s,t)-path in G is lower bounded by the shortest distance, $d(s,t)$, between these vertices. Based on this straightforward observation, Bezáková et al. in [2] introduced the Longest Detour problem that asks, given a graph G, two vertices s,t, and a positive integer k, whether G has an (s,t)-path with at least $d(s,t)+k$ vertices. They proved that for undirected graphs, this problem can be solved in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$. That is, it is FPT. For the variant of the problem where the question is whether G has an (s,t)-path with *exactly* $d(s,t)+k$ vertices, a randomized algorithm with running time $2.746^k \cdot n^{\mathcal{O}(1)}$ and a deterministic algorithm with running time $6.745^k \cdot n^{\mathcal{O}(1)}$ were obtained. In a recent work, Fomin et al. [17] studied the parameterization of Long Path and Long Cycle above the degeneracy d of the input graph G. Formally, a graph G has degeneracy at most d if every subgraph of G has a vertex of degree at most d. A classic result by Erdős and Gallai [15] from 1959 states that any graph of degeneracy $d > 1$ has a cycle (and hence also path) on at least d vertices. If the graph G is not guaranteed to be 2-connected, then deciding whether G contains a cycle of length $d+2$ is already NP-hard, and therefore parameterization above degeneracy does not make sense. However, when we add the requirement that G is 2-connected, then then deciding whether G contains a cycle of length $d+k$ parameterized by k can be done in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ and hence FPT. A similar situation holds for Long Path where connectivity replaces 2-connectivity.

1.1 Multiplicative Above Guarantee Parameterization

To the best of our knowledge, all successful above guarantee parameterizations are additive. Roughly speaking, this means that if we have a lower bound τ on the optimal solution size, then we ask whether we can find a solution of size (at least or at most) $\tau + p$ in time $f(p) \cdot n^{\mathcal{O}(1)}$. Our main message of this paper is the following.

> The goal of this paper is to introduce the notion of multiplicative above guarantee parameterization (which is neither standard nor an additive above guarantee parameterization). In multiplicative above guarantee parameterization, we ask whether we can find a solution of size (at least or at most) $\tau \cdot p$ in time $g(p) \cdot n^{\mathcal{O}(1)}$. We will illustrate our new definition by designing several FPT algorithms parameterized above a multiplicative guarantee.

We remark that a few problems in [36] were stated in a way fitting parameterization above a multiplicative guarantee. However, these were shown to be para-NP-hard [27, 28].

Recall that the *girth* of a graph G, denoted by $\gamma(G)$, is the length of the shortest cycle in G. First, consider the problem where we are given a graph G and an integer k and the objective is to test whether there is a cycle of length at least $\gamma(G)+k$ in G. If $\gamma(G) \leq 2k+6$, then clearly we can solve the problem in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ by using the algorithm in [39]. We now show that when $2k + 6 < \gamma(G)$, the problem is solvable in polynomial time. To this end, let (G, k) be a yes-instance, and let C be a hypothetical solution. That is, C is a cycle of length at least $\gamma(G) + k$. Let $g = \gamma(G)$. Let $u, v, w, x \in V(C)$ such that when we traverse C in some order from u, we encounter v at distance $\lfloor \frac{g}{2} \rfloor - 1$ from u, next we encounter w at additional distance $\lfloor \frac{g}{2} \rfloor - 1$ from v, and lastly we encounter x at additional distance $k' = (g + k) - (2\lfloor \frac{g}{2} \rfloor - 2)$ from w. Notice that $k' \in \{k + 2, k + 3\}$. Since the girth of G is g and $k + 3 < \frac{g}{2}$, we have that shortest paths between u and v, v and w, and w and x are unique and they are part of the cycle C. Therefore, we may compute the shortest paths between the pairs above, and later check whether u is reachable from x after deleting these shortest paths. Going over every possible choice of $u, v, w, x \in V(C)$, this leads to a polynomial time algorithm when $2k + 6 < \gamma(G)$. This implies that it can be decided in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ whether a graph has a cycle with at least $\gamma(G) + k$ vertices.

The informal argument above shows that Long Cycle parameterized additively above girth is not more "interesting" than just normal Long Cycle, where the input lower bound on solution size is the parameter. In a sense, it hints that the guarantee may be strengthened. In light of this, we introduce a new above guarantee version of Long Cycle (resp. Long Path), termed Long Cycle (resp. Long Path) parameterized multiplicatively above girth, as follows. The input consists of a graph G and an integer k, and the objective is to decide whether there is a cycle (resp. path) on at least $k \cdot \gamma(G)$ vertices.

1.2 Our Contribution

We first prove our main result, which states that Long Cycle (and Long Path) parameterized multiplicatively above girth is FPT. Specifically, we prove the following theorem.

▶ **Theorem 1.** Long Cycle *(and* Long Path*) parameterized multiplicatively above girth is solvable in time* $2^{\mathcal{O}(k^2)}n$.

As a complementary result to the above theorem, we assert that our parameterization is tight in the following sense.

▶ **Theorem 2.** *For any fixed constant $\epsilon > 0$,* Long Cycle *(and* Long Path*) parameterized multiplicatively above $g^{1+\epsilon}$ is para-NP-hard, where g is the girth of the input graph.*

Next, we further extend the scope of our problem domain by showing that also Vertex Cover, Connected Vertex Cover and Max Internal Spanning Tree parameterized multiplicatively above girth are FPT. Given a graph G and an integer k, the objectives of these problems are to decide whether G has a vertex cover of size at most $k \cdot \gamma(G)$, a connected vertex cover (that is, a vertex cover that induces a connected subgraph) of size at most $k \cdot \gamma(G)$, and a spanning tree with at least $k \cdot \gamma(G)$ internal vertices, respectively. Here, $\gamma(G)/2$ is a lower bound on the size of an optimal solution.

▶ **Theorem 3.** Vertex Cover*,* Connected Vertex Cover *and* Max Internal Spanning Tree *parameterized multiplicatively above girth are solvable in time $2^{\mathcal{O}(k \log k)}n$.*

Lastly, we observe that parameterization above girth (even additively) can often yield NP-hardness when k is a fixed constant. Specifically, we give Feedback Vertex Set and Cycle Packing as illustrative simple examples.

▶ **Theorem 4.** Feedback Vertex Set *and* Cycle Packing *parameterized additively above girth are para-NP-hard.*

2 Preliminaries

For terminology not explicitly defined here, we refer to the book of Diestel [13]. Throughout the paper, we consider finite undirected graphs. For a graph G, let $V(G)$ and $E(G)$ denote its vertex set and edge set, respectively. When G is clear from context, let $n = |V(G)|$ and $m = |E(G)|$. Given a subset $U \subseteq V(G)$, let $G[U]$ be the subgraph of G induced by U. The *girth* of G is the length of the shortest cycle in G,[1] and is denoted by $\gamma(G)$. A *vertex cover* of G is a set of vertices in G whose removal from G yields an edgeless graph. The *vertex cover number* of G is the smallest size of a vertex cover of G. A *feedback vertex set* of G is a set of vertices in G whose removal from G yields a forest. The *feedback vertex set number* of G is the smallest size of a feedback vertex set of G. For any $t \in \mathbb{N}$, let C_t denote the cycle on t vertices and K_t denote the complete graph on t vertices.

A *subdivision* of an edge $e = \{u, v\} \in E(G)$ is its replacement by a new degree-2 vertex whose neighbors are u and v. A *subdivision* of G is any graph that can be obtained by subdividing some of the edges of G (where a single edge can be subdivided multiple times). Let $\mathsf{Paths}(G)$ be the set of all (simple) paths in G. Given two (simple) paths P_1 and P_2 that share one or two endpoints and are internally vertex-disjoint, let $P_1 + P_2$ denote the (simple) path or cycle (if both endpoints are shared) obtained by concatenating P_1 and P_2. The *size* of a cycle or path is its number of vertices, and its *length* is its number of edges. A graph H is a *topological minor* of G if G contains a subgraph that is isomorphic to some subdivision of H. More explicitly, this notion is defined as follows.

▶ **Definition 5** (Topological Minor)**.** *A graph H is a* topological minor *of a graph G if there exist injective functions $\phi : V(H) \to V(G)$ and $\varphi : E(H) \to \mathsf{Paths}(G)$ such that for all $e = \{h, h'\} \in E(H)$, the endpoints of $\varphi(e)$ are $\phi(h)$ and $\phi(h')$, for all distinct $e, e' \in E(H)$, the paths $\varphi(e)$ and $\varphi(e')$ are internally vertex-disjoint, and there do not exist a vertex v in the image of ϕ and an edge $e \in E(H)$ such that v is an internal vertex on $\varphi(e)$.*

[1] If G does not contain any cycle, then its girth is defined as ∞.

The Catersian product of two graphs is defined as follows.

▶ **Definition 6** (Cartesian Product of Graphs). *The* Cartesian product $G \times H$ *of two graphs* G *and* H *is the graph whose vertex set is the Cartesian product* $V(G) \times V(H)$*, where any two vertices* (u, u') *and* (v, v') *are adjacent if and only if one of the following holds: (i)* $u = v$ *and* $\{u', v'\} \in E(H)$*; (ii)* $u' = v'$ *and* $\{u, v\} \in E(G)$*.*

Treewidth is a measure of how "treelike" is a graph, formally defined as follows.

▶ **Definition 7** (Treewidth). *A* tree decomposition *of a graph* G *is a pair* (T, β) *of a tree* T *and* $\beta : V(T) \to 2^{V(G)}$*, such that*

1. *for any edge* $\{x, y\} \in E(G)$ *there exists a node* $v \in V(T)$ *such that* $x, y \in \beta(v)$*, and*
2. *for any vertex* $x \in V(G)$*, the subgraph of* T *induced by the set* $T_x = \{v \in V(T) : x \in \beta(v)\}$ *is a non-empty tree.*

The width *of* (T, β) *is* $\max_{v \in V(T)}\{|\beta(v)|\} - 1$*. The* treewidth *of* G*, denoted by* $\mathtt{tw}(G)$*, is the minimum width over all tree decompositions of* G*.*

The treewidth of a graph can be efficiently approximated up to a constant factor as follows.

▶ **Proposition 8** ([7]). *There exists an algorithm that, given a graph* G *and an integer* t*, in time* $2^{\mathcal{O}(t)}n$ *either determines that the treewidth of* G *is larger than* t*, or outputs a tree decomposition of* G *of width at most* $5t + 4$*.*

The following relation between treewidth and feedback vertex set number is folklore.

▶ **Proposition 9** ([12]). *There exists an algorithm that, given a graph* G *with a feedback vertex set* U*, in time* $\mathcal{O}(|U|n)$ *outputs a tree decomposition of* G *of width* $|U|$*.*

3 FPT Algorithm for Long Cycle

In this section, we consider the parameterized complexity of Long Cycle parameterized multiplicatively above girth, and prove Theorems 1 and 2. For our positive result, we required the following definition and propositions. First, we give the definition of a graph called a t-prism, which has $2t$ vertices and $3t$ edges (see Fig. 1).

▶ **Definition 10** (Prism). *For any* $t \in \mathbb{N}$*, the* t-prism *is the Cartesian product* $C_t \times K_2$*.*

The following proposition asserts that a graph of sufficiently high treewidth necessarily contains a large prism as a topological minor.

▶ **Proposition 11** ([3]). *For any* $k \in \mathbb{N}$*, any graph* G *of treewidth at least* $60k^2$ *contains a* k*-prism as a topological minor.*[2]

In case the treewidth of the graph is low, we will be able to solve the problem by making use of the following proposition.

▶ **Proposition 12** ([6]). *There exists an algorithm that, given a graph* G *and a tree decomposition of* G *of width* t*, in time* $2^{\mathcal{O}(t)}n$ *outputs a cycle (if one exists) and a path in* G *of maximum sizes.*

[2] Although the result is mentioned in terms of minors, the proof given in [3] yields the result stated here.

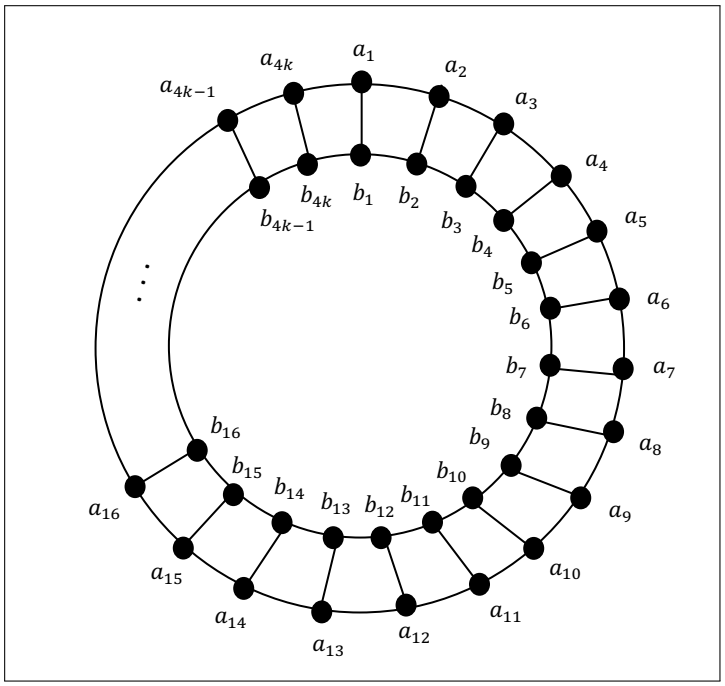

Figure 1 A $4k$-prism.

The main combinatorial lemma we need for our positive result is as follows. This lemma handles the case where the treewidth of the graph is large.

▶ **Lemma 13.** *For any $k \in \mathbb{N}$, any graph G that contains the $4k$-prism as a topological minor also has a cycle on at least $\gamma(G) \cdot k$ vertices.*

Proof. Let H be the $4k$-prism. Notice that the vertex set of H can be denoted by $V(H) = \{a_1, a_2, \ldots, a_{4k}, b_1, b_2, \ldots, b_{4k}\}$ so that $H[\{a_1, a_2, \ldots, a_{4k}\}]$ and $H[\{b_1, b_2, \ldots, b_{4k}\}]$ are cycles and $\{\{a_i, b_i\} : i \in \{1, 2, \ldots, 4k\}\} \subseteq E(H)$ (see Fig. 1). Let $\phi\colon V(H) \to V(G)$ and $\varphi\colon E(H) \to \mathsf{Paths}(G)$ be some two functions that witness that H is a topological minor of G. Let $S_1, \ldots, S_{2k}$ be a set of $2k$ vertex-disjoint cycles of length 4 in H defined as follows: For each $i \in \{0, 1, \ldots, 2k-1\}$, $S_i = H[\{a_{2i+1}, a_{2i+2}, b_{2i+2}, b_{2i+1}\}]$ (see Fig. 1). Additionally, we define two (not vertex-disjoint) cycles C and C' in H as follows: $C = H[\{a_1, a_2, b_2, b_3, a_3, a_4, \ldots, a_{4k-1}, a_{4k}, b_{4k}\}]$ and $C' = H[\{a_1, b_1, b_2, a_2, a_3, b_3, \ldots, b_{4k}, a_{4k}, a_1\}]$ (see Fig. 1).

Now, for each $i \in \{0, 1, \ldots, 2k-1\}$, let $A_i = \varphi(\{a_{2i+1}, a_{2i+2}\}) + \varphi(\{a_{2i+2}, b_{2i+2}\}) + \varphi(\{b_{2i+2}, b_{2i+1}\}) + \varphi(\{b_{2i+1}, a_{2i+1}\})$. This notation is well defined as required to concatenate paths – specifically, here we concatenated internally vertex-disjoint paths sharing exactly one endpoint except for the last concatenation where both endpoints are shared (by the paths $\varphi(\{a_{2i+1}, a_{2i+2}\}) + \varphi(\{a_{2i+2}, b_{2i+2}\}) + \varphi(\{b_{2i+2}, b_{2i+1}\})$ and $\varphi(\{b_{2i+1}, a_{2i+1}\})$). Roughly speaking, A_i is the cycle to which φ maps S_i. Notice that $\{A_i\}_{i=0}^{2k-1}$ is a collection of $2k$ vertex-disjoint cycles in G. Therefore, together they contain at least $2\gamma(G) \cdot k$ edges.

Additionally, let $B = \varphi(\{a_1, a_2\}) + \varphi(\{a_2, b_2\}) + \varphi(\{b_2, b_3\}) + \varphi(\{b_3, a_3\} + \cdots + \varphi(\{a_{4k}, b_{4k}\} + \varphi(\{b_{4k}, a_1\})$. Again, this notation is well defined as required to concatenate paths. Similarly, let $B' = \varphi(\{a_1, b_1\}) + \varphi(\{b_1, b_2\}) + \varphi(\{b_2, a_2\}) + \varphi(\{a_2, a_3\} + \cdots + \varphi(\{b_{4k}, a_{4k}\} + \varphi(\{a_{4k}, a_1\})$. Roughly speaking, B and B' are the cycles to which φ maps C and C', respectively. Notice that $E(H) = E(C) \cup E(C')$. Thus, we deduce that $\bigcup_{i=0}^{2k-1} E(A_i) \subseteq E(C) \cup E(C')$. From this,

we further deduce least one among the cycles B and B' must contain at least half the edges in $\bigcup_{i=0}^{2k-1} E(A_i)$. Because we already showed that $|\bigcup_{i=0}^{2k-1} E(A_i)| \geq 2\gamma(G) \cdot k$, this means that at least one among the cycles B and B' has length (and therefore also size) at least $\gamma(G) \cdot k$. This completes the proof. ◀

By Proposition 11, any graph G of treewidth at least $60 \cdot (4k)^2 = 960k^2$ contains a $4k$-prism as a topological minor. Thus, we derive the following corollary to Lemma 13.

▶ **Corollary 14.** *For any $k \in \mathbb{N}$, any graph G of treewidth at least $960k^2$ has a cycle on at least $\gamma(G) \cdot k$ vertices.*

We are now ready to prove our main theorem.

▶ **Theorem 1.** Long Cycle *(and* Long Path*) parameterized multiplicatively above girth is solvable in time $2^{\mathcal{O}(k^2)}n$.*

Proof. Given an instance (G, k) of Long Cycle (Long Path), the algorithm is executed as follows. First, it calls the algorithm in Proposition 8 with $t = 960k^2$ in time $2^{\mathcal{O}(t)}n = 2^{\mathcal{O}(k^2)}n$ to either determine that the treewidth of G is larger than t, or output a tree decomposition of G of width at most $5t + 4$. In the first case, it concludes that (G, k) is a Yes-instance of Long Cycle (resp. Long Path), which is correct by Corollary 14. In the second case, it calls the algorithm in Proposition 12 to obtain a cycle (resp. path) of maximum size in G in time $2^{\mathcal{O}(5t+4)}n = 2^{\mathcal{O}(k^2)}n$, and concludes that (G, k) is a Yes-instance of Long Cycle (resp. Long Path) if and only if the size of this cycle (resp. path) is at least $\gamma(G) \cdot k$. ◀

▶ **Theorem 2.** *For any fixed constant $\epsilon > 0$,* Long Cycle *(and* Long Path*) parameterized multiplicatively above $g^{1+\epsilon}$ is para-NP-hard, where g is the girth of the input graph.*

Proof. Consider some fixed constant $\epsilon > 0$. Our proof is based on a reduction from the Hamiltonian Cycle (resp. Hamiltonian Path) problem, which is known to be NP-hard [23], to Long Cycle (resp. Long Path) parameterized multiplicatively above $g^{1+\epsilon}$. In the Hamiltonian Cycle (resp. Hamiltonian Path) problem, we are given a graph G and the objective is to decide whether G contains a (simple) cycle (resp. path) on n vertices. In what follows, we only describe the proof for Long Cycle (since the proof for Long Path is similar).

We now describe the reduction. To this end, let G be an instance of Hamiltonian Cycle. Let b be the smallest positive integer such that $b(1 + \epsilon) \in \mathbb{N}$. (Note that b depends only on ϵ.) Let $a = b(1 + \epsilon) - 1$. Notice that $b \leq a$ (since $b \cdot \epsilon$ is a positive integer). Now, subdivide each edge in G $n^a - 1$ times, and let G' be the resulting graph. Let H be the disjoint union of G' and C, where C is a cycle of length n^b. Finally, set $k = 1$, and let (H, k) be the output instance of Long Cycle parameterized multiplicatively above $g^{1+\epsilon}$. Notice that here, g is the girth of H, i.e. $g = \gamma(H)$.

Notice that for any $\ell \in \mathbb{N}$, for any cycle of size ℓ in G, there is a corresponding cycle of size $\ell \cdot n^a$ in G', and vice versa. This implies that any cycle in G' has size at least $3n^a$. Thus, as $b \leq a$, there is a unique shortest cycle in H, which is C. Therefore, $\gamma(H) = n^b$. Then, any cycle (resp. path) C' on at least $(\gamma(H))^{1+\epsilon}k = n^{b(1+\epsilon)} = n^{1+a}$ vertices in H is fully contained in G'. From this, we conclude that for any cycle of size n in G, there is a corresponding cycle of size $(\gamma(H))^{1+\epsilon}k$ in H, and vice versa. This yields the correctness of the reduction. In particular, a parameterized algorithm for Long Cycle parameterized multiplicatively above $g^{1+\epsilon}$ would solve (H, k) is polynomial time (because $k = 1$), and thereby yield a polynomial-time for Hamiltonian Cycle. This completes the proof. ◀

4 FPT Algorithms for (Connected) Vertex Cover and Max Internal Spanning Tree

In this section, we consider the parameterized complexity of (Connected) Vertex Cover and Max Internal Spanning Tree parameterized multiplicatively above girth, and prove Theorem 3. The following proposition asserts that every graph contains either a small feedback vertex set or a large number of vertex-disjoint cycles (with a logarithmic gap), which can also be computed efficiently.

▶ **Proposition 15** ([16, 32]). *For some fixed constant $c \in \mathbb{N}$, there exists an algorithm that, given any $r \in \mathbb{N}$ and a graph G, in time $r^{\mathcal{O}(1)}n$ outputs either a feedback vertex set of G of size at most $cr \log r$ or r vertex-disjoint cycles in G.*

In what follows, we use c to refer to the constant in this proposition. In case the treewidth of the graph is low, we will be able to solve the problem by making use of the following proposition.

▶ **Proposition 16** ([12, 19]). *There exist algorithms for* Vertex Cover, Connected Vertex Cover *and* Max Internal Spanning Tree *that run in time $2^{\mathcal{O}(t)}n$, where t is the treewidth of the input graph.*[3]

The main combinatorial lemmas required for our algorithms are as follows. They handle the case where the feedback vertex set number (and hence also treewidth) of the graph is low. First, for (Connected) Vertex Cover, we need the following lemma.

▶ **Lemma 17.** *For any $r \in \mathbb{N}$, any graph G that contains r vertex-disjoint cycles has vertex cover number at least $\frac{\gamma(G)}{2} \cdot r$.*

Proof. Let $\mathcal{C}$ be a collection of r vertex-disjoint cycles of G. Consider any vertex cover U of G. Then, U must contain at least half the vertices of each cycle in $\mathcal{C}$ in order to contain an endpoint of every edge of that cycle. Since each cycle in $\mathcal{C}$ contains at least $\gamma(G)$ vertices, the lemma follows. ◀

Second, for Max Internal Spanning Tree, we need the following lemma.

▶ **Lemma 18.** *For any $r \in \mathbb{N}$, any* connected *graph G that contains r vertex-disjoint cycles has a spanning tree with at least $(\gamma(G) - 1) \cdot r - 1$ internal vertices.*[4]

Proof. Let $\mathcal{C}$ be a collection of r vertex-disjoint cycles of G. Let H be the graph obtained from G by replacing each cycle $C \in \mathcal{C}$ by a single new vertex v_C (that is made adjacent to all vertices previously adjacent to at least one vertex in C). Let T be some spanning tree of H (whose existence follows from the supposition that G, and hence also H, is connected).

Now, we traverse T in preorder, and when we encounter a new vertex of the form v_C, we perform the following operations. Let p be the parent of v_C in T (if it is not the root), and let u be some vertex in C that is adjacent to p (if v_C is the root, let u be any vertex in C). Then, replace v_C in T by a path P from u to one of its neighbors in C so that this path P contains exactly all the edges of C except one (between u and the chosen neighbor).

[3] Such an algorithm for Max Internal Spanning Tree is not given explicitly, but it is easily seen to follow from the approach of dynamic programming over tree decompositions using representative sets as in [19].

[4] Notice that the supposition that r is positive also implies that G is not a forest and hence $\gamma(G)$ is finite.

Here, make u be the child of p in T, and every child w of v_C is handled as follows: the subtree of w is "hanged" from a vertex of the path P such that w (or some vertex in the cycle represented by w, if w is a new vertex) is adjacent to it in G. Notice that at the end of this process, we obtain a spanning tree of G with the following property. For each cycle $C \in \mathcal{C}$, all vertices except possibly one are internal vertices, with the exception of possibly one cycle (if there was a cycle represented by the root of T) where all vertices except possibly two are internal vertices. Thus, this spanning tree has at least $(\gamma(G) - 1) \cdot r - 1$ internal vertices. This completes the proof. ◀

We are now ready to prove our main theorem.

▶ **Theorem 3.** Vertex Cover, Connected Vertex Cover *and* Max Internal Spanning Tree *parameterized multiplicatively above girth are solvable in time* $2^{\mathcal{O}(k \log k)}n$.

Proof. Given an instance (G, k) of (Connected) Vertex Cover (resp. Max Internal Spanning Tree), the algorithm is executed as follows. Without loss of generality, we suppose that G is connected in the case of Max Internal Spanning Tree, else we clearly have a No-instance of this problem. First, the algorithm calls the algorithm in Proposition 15 with $r = 2k + 1$ in time $r^{\mathcal{O}(1)}n = k^{\mathcal{O}(1)}n$ to obtain either a feedback vertex set of G of size at most $cr \log r$ or $r > 2k$ vertex-disjoint cycles in G. In the second case, by Lemma 17 G has vertex cover number strictly larger than $\frac{\gamma(G)}{2} \cdot 2k = \gamma(G) \cdot k$, and by Lemma 18, it also has a spanning tree with at least $(\gamma(G) - 1) \cdot 2k - 1 \geq \gamma(G) \cdot k$ internal vertices (where the last inequality follows from the facts that $\gamma(G) \geq 3$ and $k \in \mathbb{N}$). Thus, for Vertex Cover and Connected Vertex Cover the algorithm correctly determines that the answer is No, and for Max Internal Spanning Tree the algorithm correctly determines that the answer is Yes. In the first case, the algorithm calls the algorithm in Proposition 9 in time $\mathcal{O}(k \log k \cdot n)$ to obtain a tree decomposition of G of width at most $cr \log r = \mathcal{O}(k \log k)$. Then, it calls the algorithm in Proposition 12 in time $2^{\mathcal{O}(k \log k)}n$ to obtain a (connected) vertex cover of G of minimum size (resp. a spanning tree of G of maximum number of internal vertices), and concludes that (G, k) is a Yes-instance of (Connected) Vertex Cover (resp. Max Internal Spanning Tree) if and only if the output (connected) vertex cover has size at most $\gamma(G) \cdot k$ (resp. the output spanning tree has at least $\gamma(G) \cdot k$ internal vertices). ◀

5 Hardness for Feedback Vertex Set and Cycle Packing

Lastly, we prove the correctness of Theorem 4.

▶ **Theorem 4.** Feedback Vertex Set *and* Cycle Packing *parameterized additively above girth are para-NP-hard.*

Proof. We only prove the lemma for Feedback Vertex Set since the proof for Cycle Packing follows from symmetric arguments. For this purpose, we give a reduction from Feedback Vertex Set itself, which is an NP-hard problem [23]. Towards this, let (G, k) be an instance of Feedback Vertex Set. Then, obtain H from G by subdividing each edge of G k times, and adding a new cycle of size k. Clearly, G has a feedback vertex set of size at most k if and only if H has a feedback vertex set of size at most $k + 1$ (the extra 1 is required to hit the new cycle). Moreover, the girth of H is exactly k. Thus, an algorithm for Feedback Vertex Set parameterized additively above girth should solve $(H, 1)$ in polynomial time (since the parameter is 1), thereby determining whether the feedback vertex set number of H is at most k, and consequently solving (G, k). ◀

References

1. Noga Alon, Gregory Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. Solving MAX-r-SAT Above a Tight Lower Bound. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 511–517. SIAM, 2010.
2. Ivona Bezáková, Radu Curticapean, Holger Dell, and Fedor V. Fomin. Finding Detours is Fixed-Parameter Tractable. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017*, volume 80 of *LIPIcs*, pages 54:1–54:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.
3. E. Birmelé, J. A. Bondy, and B. A. Reed. Brambles, Prisms and Grids. In Adrian Bondy, Jean Fonlupt, Jean-Luc Fouquet, Jean-Claude Fournier, and Jorge L. Ramírez Alfonsín, editors, *Graph Theory in Paris: Proceedings of a Conference in Memory of Claude Berge*, pages 37–44. Birkhäuser Basel, Basel, 2007.
4. Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *CoRR*, abs/1007.1161, 2010. `arXiv:1007.1161`.
5. Hans L. Bodlaender. On Linear Time Minor Tests with Depth-First Search. *J. Algorithms*, 14(1):1–23, 1993. `doi:10.1006/jagm.1993.1001`.
6. Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. `doi:10.1016/j.ic.2014.12.008`.
7. Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A $c^k n$ 5-Approximation Algorithm for Treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016. `doi:10.1137/130947374`.
8. Jianer Chen, Joachim Kneis, Songjian Lu, Daniel Mölle, Stefan Richter, Peter Rossmanith, Sing-Hoi Sze, and Fenghui Zhang. Randomized divide-and-conquer: improved path, matching, and packing algorithms. *SIAM J. Comput.*, 38(6):2526–2547, 2009. `doi:10.1137/080716475`.
9. Jianer Chen, Songjian Lu, Sing-Hoi Sze, and Fenghui Zhang. Improved algorithms for path, matching, and packing problems. In *Proceedings of the18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 298–307. SIAM, 2007.
10. Robert Crowston, Michael R. Fellows, Gregory Z. Gutin, Mark Jones, Eun Jung Kim, Fran Rosamond, Imre Z. Ruzsa, Stéphan Thomassé, and Anders Yeo. Satisfying more than half of a system of linear equations over GF(2): A multivariate approach. *J. Comput. Syst. Sci.*, 80(4):687–696, 2014.
11. Robert Crowston, Mark Jones, Gabriele Muciaccia, Geevarghese Philip, Ashutosh Rai, and Saket Saurabh. Polynomial Kernels for lambda-extendible Properties Parameterized Above the Poljak-Turzik Bound. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 24 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43–54, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
12. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. `doi:10.1007/978-3-319-21275-3`.
13. Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, third edition, 2005.
14. Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. `doi:10.1007/978-1-4471-5559-1`.
15. P. Erdős and T. Gallai. On maximal paths and circuits of graphs. *Acta Math. Acad. Sci. Hungar*, 10:337–356 (unbound insert), 1959.
16. P Erdős and L Pósa. On independent circuits contained in a graph. *Canad. J. Math.*, 17:347–352, 1965.

17 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Going Far From Degeneracy. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms (ESA 2019)*, volume 144 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 47:1–47:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ESA.2019.47.

18 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient Computation of Representative Families with Applications in Parameterized and Exact Algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.

19 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient Computation of Representative Families with Applications in Parameterized and Exact Algorithms. *J. ACM*, 63(4):29:1–29:60, 2016.

20 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Long directed (s, t)-path: FPT algorithm. *Inf. Process. Lett.*, 140:8–12, 2018.

21 F.V. Fomin, D. Lokshtanov, S. Saurabh, and M. Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2018.

22 Harold N. Gabow and Shuxin Nie. Finding a long directed cycle. *ACM Transactions on Algorithms*, 4(1), 2008. doi:10.1145/1328911.1328918.

23 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

24 Gregory Gutin, Eun Jung Kim, Michael Lampis, and Valia Mitsou. Vertex Cover Problem Parameterized Above and Below Tight Bounds. *Theory of Computing Systems*, 48(2):402–410, 2011. doi:10.1007/s00224-010-9262-y.

25 Gregory Gutin, Leo van Iersel, Matthias Mnich, and Anders Yeo. Every ternary permutation constraint satisfaction problem parameterized above average has a kernel with a quadratic number of variables. *J. Computer and System Sciences*, 78(1):151–163, 2012. doi:10.1016/j.jcss.2011.01.004.

26 Gregory Z. Gutin and Viresh Patel. Parameterized Traveling Salesman Problem: Beating the Average. *SIAM J. Discrete Math.*, 30(1):220–238, 2016.

27 Gregory Z. Gutin, Arash Rafiey, Stefan Szeider, and Anders Yeo. The Linear Arrangement Problem Parameterized Above Guaranteed Value. *Theory Comput. Syst.*, 41(3):521–538, 2007.

28 Gregory Z. Gutin, Stefan Szeider, and Anders Yeo. Fixed-Parameter Complexity of Minimum Profile Problems. *Algorithmica*, 52(2):133–152, 2008.

29 Falk Hüffner, Sebastian Wernicke, and Thomas Zichner. Algorithm Engineering for Color-Coding with Applications to Signaling Pathway Detection. *Algorithmica*, 52(2):114–132, 2008. doi:10.1007/s00453-007-9008-7.

30 Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. Divide-and-Color. In *Proceedings of the 34th International Workshop Graph-Theoretic Concepts in Computer Science (WG)*, volume 4271 of *Lecture Notes in Computer Science*, pages 58–67. Springer, 2008.

31 Ioannis Koutis. Faster Algebraic Algorithms for Path and Packing Problems. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5125 of *Lecture Notes in Comput. Sci.*, pages 575–586. Springer, 2008.

32 Daniel Lokshtanov, Amer E. Mouawad, Saket Saurabh, and Meirav Zehavi. Packing Cycles Faster Than Erdos-Posa. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 71:1–71:15, 2017.

33 Meena Mahajan and Venkatesh Raman. Parameterizing above Guaranteed Values: MaxSat and MaxCut. *J. Algorithms*, 31(2):335–354, 1999.

34 Meena Mahajan, Venkatesh Raman, and Somnath Sikdar. Parameterizing above or below guaranteed values. *J. Computer and System Sciences*, 75(2):137–153, 2009. doi:10.1016/j.jcss.2008.08.004.

35 B. Monien. How to find long paths efficiently. In *Analysis and design of algorithms for combinatorial problems (Udine, 1982)*, volume 109 of *North-Holland Math. Stud.*, pages 239–254. North-Holland, Amsterdam, 1985. doi:10.1016/S0304-0208(08)73110-4.

36 Maria J. Serna and Dimitrios M. Thilikos. Parameterized Complexity for Graph Layout Problems. *Bulletin of the EATCS*, 86:41–65, 2005.

37 Dekel Tsur. Faster deterministic parameterized algorithm for k-Path. *CoRR*, abs/1808.04185, 2018. arXiv:1808.04185.

38 Ryan Williams. Finding Paths of Length k in $O^*(2^k)$ Time. *Inf. Process. Lett.*, 109(6):315–318, 2009.

39 Meirav Zehavi. Mixing Color Coding-Related Techniques. In *ESA 2015*, volume 9294 of *Lecture Notes in Computer Science*, pages 1037–1049. Springer, 2015.

40 Meirav Zehavi. A randomized algorithm for long directed cycle. *Inf. Process. Lett.*, 116(6):419–422, 2016. doi:10.1016/j.ipl.2016.02.005.