



Parameterized Complexity of Diameter

Matthias Bentert¹ · André Nichterlein¹

Received: 18 July 2019 / Accepted: 23 August 2022 / Published online: 5 September 2022
© The Author(s) 2022

Abstract

DIAMETER—the task of computing the length of a longest shortest path—is a fundamental graph problem. Assuming the Strong Exponential Time Hypothesis, there is no $O(n^{1.99})$ -time algorithm even in sparse graphs (Roditty L, Williams, VV in Fast approximation algorithms for the diameter and radius of sparse graphs. In: Proceedings of the 45th Symposium on Theory of Computing Conference (STOC '13), pp 515–524. ACM, 2013). To circumvent this lower bound, we investigate which parameters allow for running times of the form $f(k)(n + m)$ where k is the respective parameter and f is a computable function. To this end, we systematically explore a hierarchy of structural graph parameters.

Keywords FPT in P · Multivariate algorithmics · Strong exponential time hypothesis · Finegrained reductions

1 Introduction

The diameter is arguably among the most fundamental graph parameters. Most known algorithms for determining the diameter first compute the shortest path between each pair of vertices (APSP: ALL-PAIRS SHORTEST PATHS) and then return the maximum [1]. The currently fastest algorithms for APSP in weighted graphs have a running time of $O(n^3 / 2^{\Omega(\sqrt{\log n})})$ in dense graphs [13] and $O(nm + n^2 \log n)$ in sparse graphs [30], respectively. In this work, we focus on the unweighted case. Formally, we study the following problem:

An extended abstract appeared in the Proceedings of the 11th International Conference on Algorithms and Complexity (CIAC '19). This version contains additional detail and omitted proofs.

✉ Matthias Bentert
matthias.bentert@tu-berlin.de

André Nichterlein
andre.nichterlein@tu-berlin.de

¹ Algorithmics and Computational Complexity, Faculty IV, TU Berlin, Berlin, Germany

DIAMETER

Input: An undirected, connected, unweighted graph $G = (V, E)$.

Task: Compute the length of a longest shortest path in G .

The (theoretically) fastest algorithm for DIAMETER runs in $O(n^{2.373})$ time and is based on fast matrix multiplication [40]. This upper bound can (presumably) not be improved by much as Roditty and Williams [39] showed that solving DIAMETER in $O((n + m)^{2-\varepsilon})$ time for any $\varepsilon > 0$ breaks the SETH (Strong Exponential Time Hypothesis [28, 29]). Seeking for ways to circumvent this lower bound, we follow the line of “parameterization for polynomial-time solvable problems” [25] (also referred to as “FPT in P”). This approach is recently actively studied and sparked a lot of research [1, 4, 10, 16, 22, 23, 31, 32, 34]. Given some parameter k , we aim for an algorithm with a running time of $f(k)(n + m)$ that solves DIAMETER. Starting FPT in P for DIAMETER, Abboud et al. [1] observed that, unless the SETH fails, the function f has to be an *exponential* function if k is the treewidth of the graph. We extend their research by systematically exploring the parameter space looking for parameters where f can be a polynomial. If such running times contradict conditional lower bounds, then we seek for matching upper bounds of the form $f(k)(n + m)$ or $f(k)n^2$ where f is exponential.

In a second step, we combine parameters that are known to be small in many real-world graphs. We concentrate on social networks which often have special characteristics, including the “small-world” property and a power-law degree distribution [33, 35–38]. We therefore combine parameters related to the diameter with parameters related to the h -index¹; both parameters can be expected to be orders of magnitude smaller than the number of vertices in large social networks.

Related Work. Due to its importance, DIAMETER is extensively studied. Algorithms employed in practice have usually a worst-case running time of $O(nm)$, but are much faster in experiments. See e.g. Borassi et al. [6] for a recent example which also yields good performance bounds using average-case analysis [7]. Concerning worst-case analysis, the theoretically fastest algorithms are based on matrix multiplication and run in $O(n^{2.373})$ time [40] and any $O((n + m)^{2-\varepsilon})$ -time algorithm refutes the SETH [39].

The following results on approximating DIAMETER are known: It is easy to see that a simple breadth-first search gives a linear-time 2-approximation. Aingworth et al. [2] improved the approximation factor to $3/2$ at the expense of the higher running time of $O(n^2 \log n + m\sqrt{n \log n})$. The lower bound of Roditty and Williams [39] also implies that approximating DIAMETER within a factor of $3/2 - \delta$ in $O(n^{2-\varepsilon})$ time refutes the SETH. Moreover, for any $\varepsilon, \delta > 0$ a $(3/2 - \delta)$ -approximation in $O(m^{2-\varepsilon})$ time or a $(5/3 - \delta)$ -approximation in $O(m^{3/2-\varepsilon})$ time also refute the SETH [3, 11]. These lower bounds have been extended to cover more trade-offs between approximation factor and running time [17]. On planar graphs, there is an approximation scheme with near linear running time [43]; the fastest exact algorithm for DIAMETER on planar graphs runs in $O(n^{1.667})$ time [24].

¹ The h -index of a graph G is the largest number ℓ such that G contains at least ℓ vertices of degree at least ℓ .

Concerning FPT in P, DIAMETER can be solved in $2^{O(k)}n^{1+\varepsilon}$ time for any $\varepsilon > 0$, where k is the treewidth of the graph [10]. However, the reduction for the lower bound of Roditty and Williams [39] implies that for any $\varepsilon > 0$ a $2^{o(k)}n^{2-\varepsilon}$ -time algorithm refutes the SETH, where k is either the vertex cover number, the treewidth, or the combined parameter h -index and domination number. Moreover, this reduction also implies that the SETH is refuted by any $f(k)(n+m)^{2-\varepsilon}$ -time algorithm for DIAMETER for any computable function f and $\varepsilon > 0$ when k is the (vertex deletion) distance to chordal graphs. Evald and Dahlgaard [21] adapted the reduction by Roditty and Williams and proved that any $f(k)(n+m)^{2-\varepsilon}$ -time algorithm for DIAMETER parameterized by the maximum degree k for any computable function f refutes the SETH. Coudert et al. [16] proposed algorithms running in $O(k^{O(1)} \cdot n + m)$ time for the parameters modular-width, split-width, neighborhood diversity, and P_4 -sparseness. Moreover, they showed that any $2^{o(k)}n^{2-\varepsilon}$ -time algorithm where k is the clique-width of the graph would refute the SETH. Recently, Ducoffe [19] gave an algorithm whose running time matches this lower bound. Ducoffe et al. [20] analyzed the parameterized complexity of DIAMETER with respect to parameters (distance) VC-dimension.

Our Contribution. We make progress towards systematically classifying the complexity of DIAMETER parameterized by structural graph parameters. Figure 1 gives an overview of previously known and new results and their implications. We define the graph parameters for which we provide results in the sections where they are used; we refer to Brandstädt et al. [8] for definitions of the remaining parameters in Fig. 1.

In Sect. 4, we follow the “distance from triviality parameterization” [27] aiming to extend known tractability results for special graph classes to graphs with small modulators. For example, DIAMETER is linear-time solvable on trees. We obtain an $O(k \cdot n)$ -time algorithm for the parameter feedback edge number k (edge deletion number to trees). However, this is our only $k^{O(1)}(n+m)$ -time algorithm in this section. For the remaining parameters, it is already known that such algorithms refute the SETH. For the parameter distance k to cographs we therefore provide a $2^{O(k)}(n+m)$ -time algorithm. Finally, for the parameter odd cycle transversal k , we use the recently introduced notion of *General-Problem-hardness* [4] to show that DIAMETER parameterized by k is “as hard” as the unparameterized DIAMETER problem. In Sect. 5, we investigate parameter combinations. We prove that a $k^{O(1)}(n+m)^{2-\varepsilon}$ -time algorithm where k is the combined parameter diameter and maximum degree would refute the SETH. Complementing this lower bound, we provide an $f(k)(n+m)$ -time algorithm where k is the combined parameter diameter and h -index.

Many of our algorithmic results for DIAMETER transfer easily to the edge-weighted case by simply exchanging breadth-first search with Dijkstra’s algorithm and thus getting a logarithmic overhead in the running time. Whenever this is the case, we state the result for the edge-weighted case which we call WEIGHTED DIAMETER. The focus of our work (and hence the overview in Fig. 1) is still on the unweighted case. Thus, we provide hardness results only for the easier, unweighted variant DIAMETER.

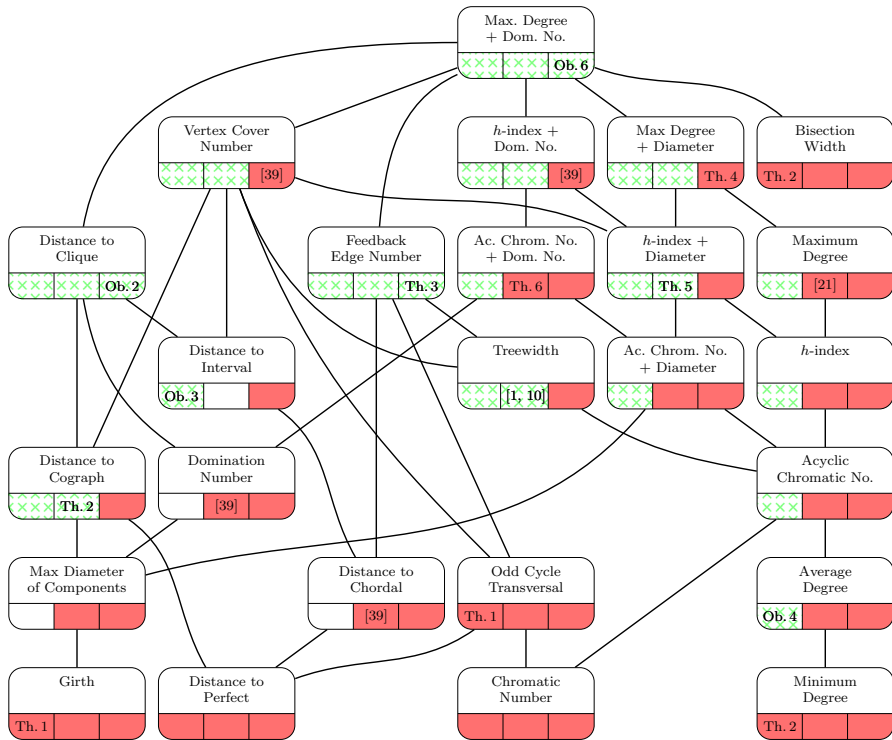


Fig. 1 Overview of the relation between the structural parameters and the respective results for DIAMETER. An edge from a parameter α to a parameter β below of α means that β can be upper-bounded in a polynomial (usually linear) function in α (see also [41]). The three small boxes below each parameter indicate whether there exists (from left to right) an algorithm running in $f(k)n^2$, $f(k)(n \log n + m)$, or $k^{O(1)}(n \log n + m)$ time, respectively. If a small box is green (and filled with a crosshatch pattern), then a corresponding algorithm exists and the box to the left is also green. Similarly, a red box indicates that a corresponding algorithm is a breakthrough. More precisely, if a middle box (right box) is red, then an algorithm running in $f(k) \cdot (n + m)^{2-\epsilon}$ (or $k^{O(1)} \cdot (n + m)^{2-\epsilon}$) time refutes the SETH. If a left box is red, then an algorithm with running time $f(k)n^2$ implies an $O(n^2)$ time algorithm for DIAMETER in general. Hardness results for a parameter α imply the same hardness results for the parameters below α . Similarly, algorithms for a parameter β imply algorithms for the parameters above β . We remark that in the above hierarchy only the algorithm behind the green box for the parameter distance to interval requires additional input related to the parameter (here the modulator to an interval graph) (Color figure online)

2 Preliminaries

We set $\mathbb{N} := \{0, 1, 2, \dots\}$ and $\mathbb{N}^+ := \mathbb{N} \setminus \{0\}$. For $\ell \in \mathbb{N}^+$ we set $[\ell] := \{1, 2, \dots, \ell\}$. We use mostly standard graph notation. For a graph $G = (V, E)$ we set $n := |V|$ and $m := |E|$. All graphs in this work are undirected. The degree of a vertex v is the number of edges that v is incident to. For a vertex subset $V' \subseteq V$, we denote with $G[V']$ the graph induced by V' . We set $G - V' := G[V \setminus V']$. A path $P = v_0 \dots v_a$ is a graph with vertex set $\{v_0, \dots, v_a\}$ and edge set $\{\{v_i, v_{i+1}\} \mid 0 \leq i < a\}$. For $u, v \in V$, we denote with $\text{dist}_G(u, v)$ the distance between u and v in G , that is, the number of edges (the sum of edge weights in weighted graphs) in a shortest path between u and v . If G

is clear from the context, then we omit the subscript. We denote by $d(G)$ the diameter of G , that is, the length of the longest shortest path in G . For WEIGHTED DIAMETER we consider edge weights to be positive integers:

WEIGHTED DIAMETER

Input: A connected graph $G = (V, E)$ and edge weights $\tau : E \rightarrow \mathbb{N}^+$.

Task: Compute $d(G)$.

Parameterized Complexity and GP-hardness. A language $L \subseteq \Sigma^* \times \mathbb{N}$ is a *parameterized problem* over some finite alphabet Σ , where $(x, k) \in \Sigma^* \times \mathbb{N}$ denotes an instance of L and k is the parameter. The language L is called *fixed-parameter tractable* if there is an algorithm that on input $(x, k) \in \Sigma^* \times \mathbb{N}$ decides whether $(x, k) \in L$ in $f(k) \cdot |x|^{O(1)}$ time, where f is some computable function only depending on k and $|x|$ denotes the size of x . For a parameterized problem L , the language $\hat{L} = \{x \in \Sigma^* \mid \exists k : (x, k) \in L\}$ is called the *unparameterized problem* associated to L . We use the notion of General-Problem-hardness which formalizes the types of reduction that allow us to exclude parameterized algorithms as they would lead to faster algorithms for the general, unparameterized, problem.

Definition 1 [4, Definition 2] Let $P \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem, let $\hat{P} \subseteq \Sigma^*$ be the unparameterized decision problem associated to P , and let $g : \mathbb{N} \rightarrow \mathbb{N}$ be a polynomial. We call P ℓ -General-Problem-hard(g) (ℓ -GP-hard(g)) if there exists an algorithm \mathcal{A} transforming any input instance I of \hat{P} into a new instance (I', k') of P such that

- (G1) \mathcal{A} runs in $O(g(|I|))$ time,
- (G2) $I \in \hat{P} \iff (I', k') \in P$,
- (G3) $k' \leq \ell$, and
- (G4) $|I'| \in O(|I|)$.

We call P *General-Problem-hard(g)* (*GP-hard(g)*) if there exists an integer ℓ such that P is ℓ -GP-hard(g). We omit the running time and call P ℓ -General-Problem-hard (ℓ -GP-hard) if g is a linear function.

Showing GP-hardness for some parameter k allows to lift algorithms for the parameterized problem to the unparameterized setting as stated next.

Lemma 1 [4, Lemma 3] Let $g : \mathbb{N} \rightarrow \mathbb{N}$ be a polynomial, let $P \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem that is GP-hard(g), and let $\hat{P} \subseteq \Sigma^*$ be the unparameterized decision problem associated to P . If there is an algorithm solving each instance (I, k) of P in $O(f(k) \cdot g(|I|))$ time, then there is an algorithm solving each instance I' of \hat{P} in $O(g(|I'|))$ time.

Applying Lemma 1 to DIAMETER yields the following. First, having an $f(k)n^{2.3}$ time algorithm with respect to a parameter k for which DIAMETER is GP-hard would yield a faster DIAMETER algorithm. Moreover, from the known SETH-based hardness results [3, 11, 39], we get the following.

Observation 1 If the SETH is true and DIAMETER is GP-hard($n^{2-\varepsilon}$) with respect to some parameter k for some $\varepsilon > 0$, then there is no $f(k) \cdot n^{2-\varepsilon'}$ time algorithm for any $\varepsilon' > 0$ and any function f .

Graph Classes and Parameter Definitions. Finally, we give an overview over the different graph classes and graph parameters used throughout the paper. To this end, let $G = (V, E)$ be a graph. A *clique* is a graph in which each pair of vertices is connected by an edge. An *independent set* is an edgeless graph. A *cograph* is a graph in which each component has diameter at most two. Equivalently, a cograph is a graph without induced paths of length three. An *interval graph* is a graph where each vertex can be represented by an interval of real numbers such that two vertices are adjacent if and only if their respective intervals overlap. The vertex set of a *bipartite graph* can be partitioned into two independent sets.

- distance to Π minimum number of vertices needed to be removed from G such that it becomes a graph in Π ;
- vertex cover number distance to independent sets;
- odd cycle transversal distance to bipartite graphs;
- feedback edge number minimum number of edges needed to be removed from G such that it becomes a forest;
- bisection width minimum number of edges needed to be removed from G such that it becomes a disconnected graph in which each connected component has at most $\lceil n/2 \rceil$ vertices;
- maximum degree highest degree of any vertex;
- average degree average over all vertex degrees;
- minimum degree smallest degree of any vertex;
- h -index maximum value h such that there are at least h vertices of degree at least h ;
- girth size of the smallest induced cycle;
- domination number minimum size of a set W of vertices such that each vertex in $V \setminus W$ has at least one neighbor in W ;
- acyclic chromatic number minimum number of colors needed to color each vertex with one of the given colors such that each subgraph induced by all vertices of one color is an independent set and each subgraph induced by all vertices of two colors is acyclic.

3 Basic Observations

In this section, we present several simple observations that complete the overview in Fig. 1. More precisely, we show algorithms with respect to the parameters distance c to clique, distance i to interval graphs, average degree a , maximum degree Δ , diameter d , and domination number γ (in the order they are listed).

Distance to clique. We start with the parameter distance c to clique and provide an algorithm with running time $O(c \cdot (n + m))$ time. Since distance to clique is the vertex cover number in the complement graph, it can be 2-approximated in linear time (without computing the complement graph).

Observation 2 DIAMETER parameterized by distance c to clique takes $O(c \cdot (n + m))$ time.

Proof Let $G = (V, E)$ be the input graph and let c be its distance to clique. Let G' be the respective induced clique graph. Compute in linear time the degree of each vertex and the number $n = |V|$ of vertices. Iteratively check for each vertex v whether its degree is $n - 1$. If $\deg(v) = n - 1$, then v can be deleted as it is in every largest clique and thus decrease n by one and the degree of each other vertex by one. If not, then we can find a vertex w which is not adjacent to v in $O(\deg(v))$ time. Put v and w in the solution set, delete both vertices and all incident edges and adjust the number of vertices and their degree accordingly. Observe that v and w cannot be contained in the same clique and therefore $v \in K$ or $w \in K$. Putting both vertices in the solution set results in a 2-approximation. This algorithm takes $O(\deg(v) + \deg(w))$ time per deleted pair v, w of vertices. Since $\sum_{v \in V} \deg(v) \in O(n + m)$ this procedure takes $O(n + m)$ time.

We use the algorithm described above to compute a set K such that $G' = G - K$ is a clique and $|K| \leq 2k$ in linear time. Since G' is a clique, its diameter is one if there are at least two vertices in the clique. We therefore assume that there is at least one vertex in the deletion set K . Compute for each vertex $v \in K$ a breadth-first search rooted in v in linear time and return the largest distance found. The returned value is the diameter of G as each longest induced path is either of length one or has at least one endpoint in K . The procedure takes $O(|K| \cdot (n + m) + n + m) = O(c \cdot (n + m))$ time. \square

Note that for WEIGHTED DIAMETER a result similar to Observation 2 would yield a faster algorithm for DIAMETER: In a clique C with n vertices and edge weights either 1 or n , one can encode any connected unweighted graph G by giving edges in G weight one in C and any non-edge in G a weight of n in C . It is easy to see that G has the same diameter as C . Thus, an algorithm for WEIGHTED DIAMETER with running time $O(c \cdot (n + m))$ would imply an $O(n^2)$ algorithm for DIAMETER and, hence, drastically improve on the state-of-the-art.

Distance to interval graphs. We next discuss the parameter distance to interval graphs. We first provide a general observation stating that a size k deletion set to some graph class can be used to design a $O(k \cdot n^2)$ -time algorithm if ALL-PAIRS SHORTEST PATHS can be solved in $O(n^2)$ time on graphs in the respective graph class.

Proposition 1 *Let Π be a graph class such that ALL-PAIRS SHORTEST PATHS can be solved in $O(n^2)$ time on Π . If the (vertex) deletion set K to Π is given, then ALL-PAIRS SHORTEST PATHS can be solved in $O(|K| \cdot n^2)$ time.*

Proof First, we compute G' , that is, the graph without the deletion set K , and solve ALL-PAIRS SHORTEST PATHS on it in $O(n^2)$ time. Next, we perform Dijkstra’s algorithm in the input graph G from each vertex $b \in K$ in overall $O(k \cdot (n \log n + m))$ time and store the distance between b and every other vertex a in a table. The last step can be seen as running the classical Floyd-Warshall algorithm for each vertex in K : compute for each pair $a, c \in V \setminus K$

$$\text{dist}_G(a, c) := \min\{\text{dist}_{G'}(a, c), \min_{b \in K} \{\text{dist}_G(a, b) + \text{dist}_G(b, c)\}\},$$

that is, the minimum distance of a path in the original graph. Observe that a shortest path either travels through some vertex $b \in K$ or not. The distance between a and c in G is in the former case $\text{dist}_G(a, b) + \text{dist}_G(b, c)$ and in the latter case $\text{dist}_{G'}(a, c)$. The algorithm takes overall $O(k \cdot n^2)$ time. \square

It is known that (unweighted) ALL-PAIRS SHORTEST PATHS can be solved in $O(n^2)$ time on interval graphs [14, 42]. Thus we obtain the following.

Observation 3 DIAMETER parameterized by the distance i to interval graphs is solvable in $O(i \cdot n^2)$ time provided that the deletion set is given.

Computing the deletion set takes $O(6^i \cdot (n + m))$ time [12] if it is not given. We are not aware of a $i^{O(1)} \cdot n^2$ -time constant factor approximation algorithm to circumvent the exponential factor in i . Finding (or excluding) such an approximation algorithm remains a task for future work. As interval graphs contain cliques, it follows again that generalizing Observation 3 to the weighted case would improve upon the state-of-the-art algorithm for WEIGHTED DIAMETER.

Average degree. We next consider the average degree a . Observe that $2m = n \cdot a$ and therefore the standard algorithm (run Dijkstra's algorithm n times) takes $O(n \cdot (n \log n + m)) = O(n^2(\log n + a))$ time.

Observation 4 WEIGHTED DIAMETER parameterized by average degree a is solvable in $O((a + \log n) \cdot n^2)$ time.

Maximum degree and diameter. We look at two parameter combinations related to both maximum degree and diameter. Usually, this parameter is not interesting as the graph size can be upper-bounded by this parameter and thus fixed-parameter tractability with respect to this combined parameter is trivial. The input size is, however, only exponentially bounded in the parameter, so it might be tempting to search for fully polynomial algorithms. In Sect. 5.2 we exclude such a fully polynomial algorithm. Thus, the subsequent algorithm is basically optimal.

Observation 5 WEIGHTED DIAMETER parameterized by diameter d and maximum degree Δ is solvable in $O(\Delta^{2d} \cdot (d \log \Delta + \Delta))$ time.

Proof Since we may assume that the input graph only consists of one connected component, every vertex is found by any breadth-first search. Any breadth-first search may only reach depth d , where d is the diameter of the input graph, and as each vertex may only have Δ neighbors there are at most $1 + \sum_{i=1}^d \Delta \cdot (\Delta - 1)^{i-1} \leq 1 + \sum_{i=1}^d \Delta^{i-1} \cdot (\Delta - 1) = \Delta^d$ vertices (since in each “depth layer i ” there are at most $\Delta \cdot (\Delta - 1)^{i-1}$ vertices). Since $m \leq n \cdot \Delta$ the $O(n \cdot (n \log n + m))$ -time algorithm (n rounds of Dijkstra's algorithm) runs in $O(\Delta^{2d} \cdot (d \log \Delta + \Delta))$ time. \square

Maximum degree and domination number. Observe that for any graph of n vertices, domination number γ , and maximum degree Δ it holds that $n \leq \gamma \cdot (\Delta + 1)$ as each vertex is in a dominating set or is a neighbor of at least one vertex in it. The next observation follows from $m \leq n \cdot \Delta$.

Observation 6 WEIGHTED DIAMETER parameterized by domination number γ and maximum degree Δ is solvable in $O(\gamma^2 \Delta^2 (\Delta + \log(\gamma \Delta)))$ time.

The reduction of Roditty and Williams [39] can also be used to show that the SETH is refuted by any $f(\gamma)(n + m)^{2-\epsilon}$ -time algorithm for DIAMETER for any computable function f even if a minimum dominating set is given. This lower bound result is in stark contrast to a simple algorithm running in $O(\gamma(n + m))$ time that returns either the diameter or the diameter minus one.

Observation 7 Given a dominating set of size γ for an unweighted graph, one can approximate the diameter with an additive factor of one in $O(\gamma(n + m))$ time.

Proof The algorithm is as follows: Run a breadth-first search from each vertex in the dominating set D and return the largest distance found. This can be done $O(\gamma(n + m))$ time. Clearly, the value ℓ returned by the algorithm is at most the diameter d of the input graph, that is, $\ell \leq d$. It remains to show that $d \leq \ell + 1$.

To this end, let u, v be the two furthest vertices, that is, $\text{dist}(u, v) = d$. Observe that if either u or v is in the dominating set D , then the algorithm returned $\ell = d$. Thus, consider the case that neither u nor v are in D . Since D is a dominating set, there is a vertex $w \in D$ that is a neighbor of u . Since $w \in D$, the returned value is at least $\ell \geq \text{dist}(w, v)$. Hence, we have $d = \text{dist}(u, v) \leq \text{dist}(w, v) + 1 \leq \ell + 1$. \square

Note that, although computing a minimum dominating set is NP-hard, a simple greedy algorithm computes a $(1 + \log n)$ -approximation. Thus, if the dominating set is not given, the worst-case running time of the above plus-one-approximation changes to $O(\gamma(n + m) \log n)$, which is still far better than the lower bound for exactly computing the diameter.

4 Deletion Distance to Special Graph Classes

In this section, we investigate parameterizations that measure the distance to special graph classes. The hope is that when DIAMETER can be solved efficiently in a special graph class Π , then DIAMETER can be solved if the input graph is “almost” in Π . We study the following parameters in this order: odd cycle transversal (which is the same as distance to bipartite graphs), distance to cographs, and feedback edge number. Note that the lower bound of Abboud et al. [1] for the parameter vertex cover number (i.e. vertex deletion to edgeless graphs) already implies that there is no $2^{o(k)}(n + m)^{2-\epsilon}$ -time algorithm for k being one of the first two parameters in our list unless the SETH breaks, since each of these parameters is smaller than the vertex cover number (see Fig. 1).

Odd Cycle Transversal. We show that DIAMETER parameterized by odd cycle transversal and girth is 4-GP-hard. Consequently, solving DIAMETER in $f(k) \cdot n^{2.3}$ for any computable function f implies an $O(n^{2.3})$ -time algorithm for DIAMETER—which would improve the currently best (unparameterized) algorithm. The girth of a graph is the length of a shortest cycle in it.

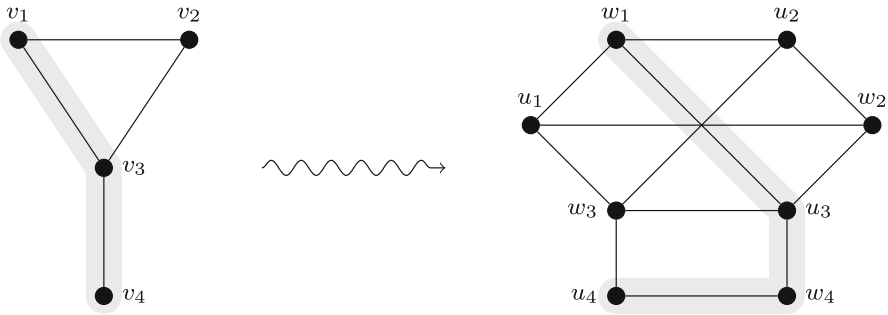


Fig. 2 Example for the construction in the proof of Theorem 1. The input graph given on the left side has diameter two and the constructed graph on the right side has diameter three. In each graph one longest shortest path is highlighted

Theorem 1 DIAMETER is 4-GP-hard with respect to the combined parameter odd cycle transversal and girth.

Proof Let $G = (V, E)$ be an arbitrary undirected graph where $V = \{v_1, v_2, \dots, v_n\}$. We construct a new graph $G' = (V', E')$ as follows: $V' := \{u_i, w_i \mid v_i \in V\}$, and $E' := \{\{u_i, w_j\}, \{u_j, w_i\} \mid \{v_i, v_j\} \in E\} \cup \{\{u_i, w_i\} \mid v_i \in V\}$.

An example of this construction can be seen in Fig. 2.

We will now prove that all properties of Definition 1 hold. It is easy to verify that the reduction can be implemented in linear time and therefore the resulting instance is of linear size as well. Observe that $\{u_i \mid v_i \in V\}$ and $\{w_i \mid v_i \in V\}$ are both independent sets and therefore G' is bipartite. Notice further that for any edge $\{v_i, v_j\} \in E$ there is an induced cycle in G' containing the vertices $\{u_i, w_i, u_j, w_j\}$. Since G' is bipartite there is no induced cycle of length three in G' and thus the girth of G' is four.

Lastly, we show that $d(G') = d(G) + 1$ by proving that if $\text{dist}(v_i, v_j)$ is odd, then $\text{dist}(u_i, w_j) = \text{dist}(v_i, v_j)$ and $\text{dist}(u_i, u_j) = \text{dist}(v_i, v_j) + 1$, and if $\text{dist}(v_i, v_j)$ is even, then $\text{dist}(u_i, u_j) = \text{dist}(v_i, v_j)$ and $\text{dist}(u_i, w_j) = \text{dist}(v_i, v_j) + 1$. Since $\text{dist}(u_i, w_i) = 1$ and $\text{dist}(u_i, w_j) = \text{dist}(u_j, w_i)$, this will conclude the proof.

Let $P = v_{a_0} v_{a_1} \dots v_{a_d}$ be a shortest path from v_i to v_j where $v_{a_0} = v_i$ and $v_{a_d} = v_j$. Let $P' = u_{a_0} w_{a_1} u_{a_2} w_{a_3} \dots$ be a path in G' . Clearly, P' is also a shortest path as there are no edges $\{u_i, w_j\} \in E'$ where $\{v_i, v_j\} \notin E$.

If d is odd, then $u_{a_0} w_{a_1} \dots w_{a_d}$ is a path of length d from u_i to w_j and $u_{a_0} w_{a_1} \dots w_{a_d} u_{a_d}$ is a path of length $d + 1$ from u_i to u_j . If d is even, then $u_{a_0} w_{a_1} \dots w_{a_{d-1}} u_{a_d}$ is a path of length d from u_i to u_j and $u_{a_0} w_{a_1} \dots w_{a_{d-1}} u_{a_d} w_{a_d}$ is a path of length $d + 1$ from u_i to w_j . Notice that G' is bipartite and thus $\text{dist}(u_i, u_j)$ must be even and $\text{dist}(u_i, w_j)$ must be odd. □

Distance to cographs. A graph is a cograph if and only if it does not contain a P_4 as an induced subgraph, where a P_4 is a path on four vertices. Providing an algorithm that matches the lower bound of Abboud et al. [1], we will show that DIAMETER parameterized by distance k to cographs can be solved in $O(k \cdot (n + m) + 2^{O(k)})$ time. To this end, we will use the following lemma covering the algorithm in a more general setting than we use.

Lemma 2 *Let $G = (V, E)$ be an edge-weighted graph and let $K \subseteq V$ a vertex subset such that each connected component in $G - K$ has diameter at most two. Then, the diameter of G can be computed in $O(k \cdot (n \log n + m + 2^{4k}))$ time, where $k := |K|$.*

Proof Our algorithm has three main steps:

1. Compute the distance from each vertex in K to each vertex in V .
2. Compute the largest distance between two vertices in $G - K$ assuming no vertex in K is in a shortest path, that is, compute the diameter of each connected component in $G - K$.
3. Compute the largest distance between two vertices in $G - K$ assuming at least one vertex in K is in each shortest path.

The first two steps are rather easy: First, we perform in $O(k \cdot (n \log n + m))$ time Dijkstra’s algorithm in G from each vertex $v \in K$ and store the distance between v and every other vertex $w \in V$ in a table. Second, we compute all connected components and their diameter in $G' := G - K$ in linear time and store for each vertex the information in which connected component it is. Note that we only need to check for each connected component C , whether C induces a clique in G' and all edge-weights are one in C ; otherwise C ’s diameter is by assumption two.

For the third step, we need to introduce some notation Let $K = \{x_1, x_2, \dots, x_k\}$. The type of a vertex $u \in V \setminus K$ is a vector of length k where the i th entry describes the distance from u to x_i with the addition that any value above three is set to 4. We say a type is non-empty if there is at least one vertex with this type. We compute for each vertex $u \in V \setminus K$ its type. Additionally we store for each non-empty type the connected component all its vertices are in or that there are at least two different connected components containing a vertex of that type. This takes $O(n \cdot k)$ time and there are at most 4^k many different types.

For step 3, we iterate over all of the $O(4^{2k})$ pairs of types (including the pairs where both types are the same) and compute the largest distance between vertices of these types. We will first argue that any pair of vertices y and z in different connected components are a vertex pair of the respective types that has largest distance. Afterwards, we show how to compute their distance in $O(k)$ time. If both types only appear in the same connected component, then the distance between the two vertices of these types is at most two. Hence, we can discard this case (one can check in linear time whether the diameter of G is at least two). If two types appear in different connected components, then a longest shortest path between vertices of the respective types contains at least one vertex in K . Observe that since each connected component has diameter at most two, at least each third vertex in any longest shortest path must be in K . Thus, a shortest y - z -path contains at least one vertex $x_i \in K$ with $\text{dist}(x_i, y) < 3$. By definition, each vertex with the same type as y has the same distance to x_i and therefore the same distance to z unless there is no shortest path from it to z that passes through x_i , that is, it is in the same connected component as z . Thus, we can choose two arbitrary vertices of the respective types in different connected components and compute their distance. Note that checking whether there are two vertices of the two respective types in different connected components can be done with a table lookup since we have already precomputed whether each type appears in at least two connected components

(and stored the unique connected component otherwise). Observe that the shortest path from y to z contains x_i and therefore $\text{dist}(y, x_i) + \text{dist}(x_i, z) = \text{dist}(y, z)$. Hence, if there are two vertices in different connected components, then we can compute the distance between y and z in $O(k)$ time by computing $\min_{x \in K} \{\text{dist}(y, x) + \text{dist}(x, z)\}$. (Note that the distances from x are precomputed.)

Overall, the algorithm takes $O(k \cdot (n \log n + m + 2^{4k}))$ time to compute the diameter of G . \square

Note that the algorithm described in the above proof does not verify if K is indeed a vertex set such that each connected component in $G - K$ has diameter at most two. Indeed, even in the unweighted case to distinguish diameter two and three in $O(n^{2-\varepsilon})$, $\varepsilon > 0$, time would refute the SETH [1]. Thus, the above algorithm cannot efficiently verify if the input meets the stated conditions. Hence, when using Lemma 2, we need a way to ensure this condition.

Recall that a cograph does not contain a P_4 as an induced subgraph. Thus, any unweighted cograph has diameter at most two (but not every diameter-two graph is a cograph, consider e.g. a cycle on five vertices). Moreover, given a graph G one can determine in linear time whether G is a cograph and can return an induced P_4 if this is not the case [9, 15]. This implies that in $O(k \cdot (n + m))$ time one can compute a set $K \subseteq V$ with $|K| \leq 4k$ such that $G - K$ is a cograph: Iteratively add all four vertices of a returned P_4 into the solution set and delete those vertices from G until it is P_4 -free. Thus, we can compute a set K that satisfies the conditions of Lemma 2 and the following theorem is immediate.

Theorem 2 DIAMETER can be solved in $O(k \cdot (n + m + 2^{16k}))$ time when parameterized by distance k to cographs.

Proof Let $G = (V, E)$ be the input graph with distance k to cograph. Let K be a set of vertices such that $G' = G - K$ is a cograph with $|K| \leq 4k$. Recall that K can be computed in $O(k \cdot (n + m))$ time.

Thus, applying Lemma 2 yields a running time of $O(k \cdot (n + m + 2^{16k}))$. Note that since we are in the unweighted setting, we can replace Dijkstra's algorithm in the proof of Lemma 2 by a simple breadth-first search and thus get rid of the log-factor in the running time. \square

Note that a clique is also a cograph. Thus, following the same argumentation given after Observation 2, it follows that a generalization of Theorem 2 to the weighted case would significantly improve the state-of-the-art algorithm for DIAMETER.

Feedback edge number. We will prove that WEIGHTED DIAMETER parameterized by feedback edge number k can be solved in $O(k \cdot n \log n)$ time. One can compute a minimum feedback edge set K (with $|K| = k$) in linear time by taking all edges not in a spanning tree. Recently, this parameter was used to speed up algorithms computing maximum matchings [31]. In the remainder of this section we will prove the following.

Theorem 3 WEIGHTED DIAMETER parameterized by feedback edge number k can be solved in $O(k \cdot n \log n)$ time.

The algorithm behind the above theorem works roughly in two steps: In a first step, we apply data reduction rules. On the one hand, these rules can shrink the graph considerably. On the other hand, these rules also create a special structure: After these rules are exhaustively applied, there are “few” vertices of degree at least three; moreover, these high-degree vertices are connected via “few” paths. In the second step, the algorithm uses this structure in a case distinction to compute the diameter in $O(k \cdot n \log n)$ time.

Note that the data reduction rules delete vertices from the graph. However, since at the time of deletion, we do not know whether these vertices are contained in a shortest path defining the diameter, we need to keep additional information. In particular, we introduce a second weight function pen (for pending) and an integer s . Intuitively, $\text{pen}(v)$ stores the length of a longest shortest path P with one endpoint being v and the other endpoint in P being already deleted by the data reduction rules. The role of s is to store the length of a longest shortest path where *both* endpoints are already deleted. This leads to the following formal problem definition:

DOUBLY WEIGHTED DIAMETER

Input: An undirected, connected graph $G = (V, E)$, weight functions $\tau : E \rightarrow \mathbb{N}^+$ and $\text{pen} : V \rightarrow \mathbb{N}$, and $s \in \mathbb{N}$.

Task: Compute $\max\{d^{\text{pen}}(G), s\}$, where

$$d^{\text{pen}}(G) := \max_{v, w \in V} \{\text{dist}_G^{\text{pen}}(v, w)\} := \max_{v, w \in V} \{\text{pen}(v) + \text{dist}_G(v, w) + \text{pen}(w)\}.$$

Notice that if all pen -weights and s are set to 0, then the problem is the same as **WEIGHTED DIAMETER**. We therefore start with initializing all pen -weights and s to 0 and applying our reduction rule that removes degree-one vertices from the graph. The main idea of the reduction rule is simple: If a degree-one vertex u is removed, then the value $\text{pen}(v)$ (v is the unique neighbor of u) is adjusted and we store in an additional variable s the length of a longest shortest path that cannot be recovered from the reduced graph. This addresses the case that a longest shortest path has both its endpoints in pending trees (trees removed by our reduction rule) that are connected to the same vertex. Initially, s is set to zero. The first reduction rule is defined as follows (see Fig. 3 for an example illustrating the subsequent two reduction rules).

Reduction Rule 1 Let u be a vertex of degree one and let v be its neighbor. Delete u and the incident edge from G , set $s = \max\{s, \text{pen}(u) + \text{pen}(v) + \tau(\{u, v\})\}$ and $\text{pen}(v) = \max\{\text{pen}(v), \text{pen}(u) + \tau(\{u, v\})\}$.

Before we analyze the running time and correctness, we first present a second reduction rule that we apply after Reduction Rule 1 is not applicable anymore. Since the resulting graph has no degree-one vertices we can partition the vertex set of the remaining graph into vertices $V^{=2}$ of degree exactly two and vertices $V^{\geq 3}$ of degree at least three. Using standard argumentation we can show that $|V^{\geq 3}| \in O(\min\{k, n\})$ and all vertices in $V^{=2}$ are either in *pending cycles* or in *maximal paths* [5, Lemma 5]. A maximal path is an induced subgraph $P = x_0x_1 \dots x_a$ where $\{x_i, x_{i+1}\} \in E$ for

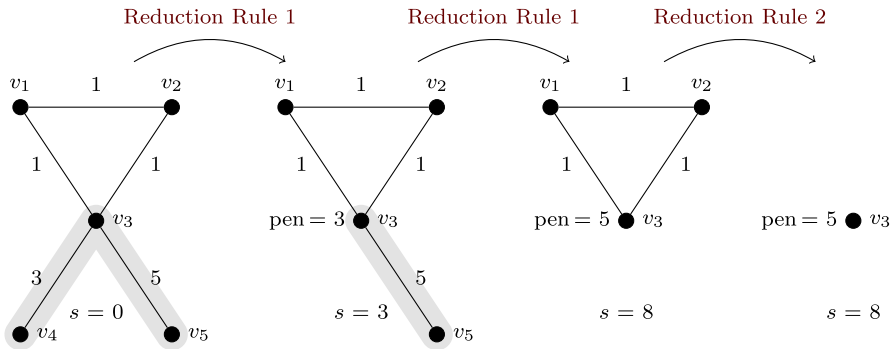


Fig. 3 Example for the application of Reduction Rules 1 and 2. On the left is the input graph, middle left and middle right are the results of applying Reduction Rule 1. On the right is the result of applying Reduction Rule 2 to the middle right graph. If no pen-value is displayed for a vertex v , then $\text{pen}(v) = 0$. The diameter-defining path is highlighted in the two left graphs and stored in s in the two right graphs (when the diameter-defining path is no longer contained in the remaining graph)

all $0 \leq i < a$, $x_0, x_a \in V^{\geq 3}$, $x_i \in V^{=2}$ for all $0 < i < a$, and $x_0 \neq x_a$. A pending cycle is basically the same except $x_0 = x_a$ and $\text{deg}(x_0)$ may possibly be two. The set \mathcal{C} of all pending cycles and \mathcal{P} of maximal paths can be computed in $O(n + m)$ time [5, Lemma 6]. The second reduction rule works similar to Reduction Rule 1, but instead of deleting degree-one vertices, it removes pending cycles.

Reduction Rule 2 Let $C = x_0x_1 \dots x_a$ be a pending cycle. Let x_k be the vertex that maximizes $\text{pen}(x_k) + \text{dist}(x_0, x_k)$ in C . Delete all vertices in C except for x_0 (and all incident edges) from G , set $s = \max\{s, d^{\text{pen}}(C)\}$ and $\text{pen}(x_0) = \max\{\text{pen}(x_0), \text{pen}(x_k) + \text{dist}(x_0, x_k)\}$.

We now prove the correctness of these two data reduction rules. That is, given an instance (G, τ, pen, s) of DOUBLY WEIGHTED DIAMETER let $(G', \tau', \text{pen}', s')$ be the instance created by applying a data reduction rule R once. Then, R is correct if $\max\{s, d^{\text{pen}}(G)\} = \max\{s', d^{\text{pen}}(G')\}$.

Lemma 3 *Reduction Rules 1 and 2 are correct.*

Proof Let $(G = (V, E), \tau, \text{pen}, s)$ be the input instance of DOUBLY WEIGHTED DIAMETER and $(G' = (V', E'), \tau', \text{pen}', s')$ the instance resulting of an application of Reduction Rule 1 to the degree-one vertex u with neighbor v or Reduction Rule 2 to a pending cycle $C = x_0, x_1, \dots, x_a$. We start with making some statements that are true for both reduction rules.

We first show that $d^{\text{pen}}(G) \geq d^{\text{pen}}(G')$, that is, the (pen-adjusted) diameter in G is at least as large as in G' . To this end, let $w, w' \in V'$ such that $d^{\text{pen}}(G') = \text{pen}'(w) + \text{dist}_{G'}(w, w') + \text{pen}'(w')$. Observe that if $w \neq v$ and $w' \neq v$ (for Reduction Rule 1) or $w \neq x_0$ and $w' \neq x_0$ (for Reduction Rule 2), then

$$\text{pen}'(w) + \text{dist}_{G'}(w, w') + \text{pen}'(w') \leq \text{pen}(w) + \text{dist}_G(w, w') + \text{pen}(w') \leq d^{\text{pen}}(G).$$

Thus, it remains to consider the case that $w' = v$ for Reduction Rule 1 and $w' = x_0$ for Reduction Rule 2 (the cases $w = v$ respectively $w = x_0$ are completely analogous). In the case of Reduction Rule 1 we have

$$\begin{aligned} & \text{pen}'(w) + \text{dist}_{G'}(w, w') + \text{pen}'(w') \\ &= \text{pen}(w) + \text{dist}_G(w, v) + \max\{\text{pen}(v), \tau(\{u, v\}) + \text{pen}(u)\} \leq d^{\text{pen}}(G). \end{aligned}$$

In the case of Reduction Rule 2 we have for the “furthest” vertex x_k from x_0 in C that

$$\begin{aligned} & \text{pen}'(w) + \text{dist}_{G'}(w, w') + \text{pen}'(w') \\ &= \text{pen}(w) + \text{dist}_G(w, x_0) + \max\{\text{pen}(x_0), \text{dist}(\{x_0, x_k\}) + \text{pen}(x_k)\} \leq d^{\text{pen}}(G). \end{aligned}$$

Thus, $d^{\text{pen}}(G) \geq d^{\text{pen}}(G')$.

Next, observe that $s \leq s'$. Moreover, observe that if $s \geq d^{\text{pen}}(G)$, then we have $\max\{s, d^{\text{pen}}(G)\} = s = s' = \max\{s', d^{\text{pen}}(G')\}$ since $s' \geq s \geq d^{\text{pen}}(G) \geq d^{\text{pen}}(G')$. Thus, it remains to consider the case $s < d^{\text{pen}}(G)$ and, hence, to show that $d^{\text{pen}}(G) = \max\{s', d^{\text{pen}}(G')\}$.

We split this last part of the proof into two parts, where we first consider Reduction Rule 1 and then consider Reduction Rule 2 in the second part. For the first part, let $w, w' \in V$ such that $d^{\text{pen}}(G) = \text{pen}(w) + \text{dist}_G(w, w') + \text{pen}(w')$. We make a case distinction on the size of $\{w, w'\} \cap \{u, v\}$ (that is, whether w or w' are equal to u or v).

Case 1: $|\{w, w'\} \cap \{u, v\}| = 2$. Since $s < d^{\text{pen}}(G)$, we have by definition of s' that

$$d^{\text{pen}}(G) = \text{pen}(u) + \text{dist}_G(u, v) + \text{pen}(v) = \text{pen}(u) + \tau(\{u, v\}) + \text{pen}(v) = s'.$$

Since $d^{\text{pen}}(G') \leq d^{\text{pen}}(G)$, it follows that $\max\{s', d^{\text{pen}}(G')\} = s' = d^{\text{pen}}(G)$.

In the following two cases we assume that $d^{\text{pen}}(G) > \text{pen}(u) + \text{dist}_G(u, v) + \text{pen}(v)$; otherwise we are in Case 1. Hence, it follows that also $s' < d^{\text{pen}}(G)$ since $s < d^{\text{pen}}(G)$.

Case 2: $|\{w, w'\} \cap \{u, v\}| = 1$. Thus, we need to show $d^{\text{pen}}(G') \geq d^{\text{pen}}(G)$ (as we already proved $d^{\text{pen}}(G') \leq d^{\text{pen}}(G)$ and assume $s' < d^{\text{pen}}(G)$). To this end, let $w' \in \{u, v\}$ and $w \notin \{u, v\}$. Hence, we have

$$\begin{aligned} d^{\text{pen}}(G) &= \text{pen}(w) + \text{dist}_G(w, w') + \text{pen}(w') \\ &= \text{pen}(w) + \text{dist}_G(w, v) + \max\{\text{pen}(v), \text{pen}(u) + \tau(\{u, v\})\} \\ &= \text{pen}'(w) + \text{dist}_{G'}(w, v) + \text{pen}'(v) \leq d^{\text{pen}}(G'). \end{aligned}$$

Thus, $d^{\text{pen}}(G') = d^{\text{pen}}(G)$.

Case 3: $|\{w, w'\} \cap \{u, v\}| = 0$. Again, we need to show $d^{\text{pen}}(G') \geq d^{\text{pen}}(G)$. To this end, neither w nor w' are changed by Reduction Rule 1. Thus,

$$\text{pen}(w) + \text{dist}_G(w, w') + \text{pen}(w') = \text{pen}'(w) + \text{dist}_{G'}(w, w') + \text{pen}'(w') \leq d^{\text{pen}}(G').$$

This finishes the last case and concludes the proof for Reduction Rule 1.

We continue with the proof for Reduction Rule 2. To this end we consider two cases: Either $s' > d^{\text{pen}}(G')$ or $s' \leq d^{\text{pen}}(G')$.

Case 1: $s' \geq d^{\text{pen}}(G')$. We show that $s' = d^{\text{pen}}(G)$. Since $s' \geq d^{\text{pen}}(G')$, there is no shortest path of length $s' + 1$ in G' . Since G and G' only differ in C , it suffices to show that there is a shortest path of length s' in G and that there is no longer path that starts in C . By construction, there is a pair of vertices x_i, x_j in C such that $\text{dist}_G^{\text{pen}}(x_i, x_j) = s'$. Now assume that there is a shortest path of length at least $s' + 1$ in G that starts in C . By construction the path has to end outside of C as otherwise s' would be larger. Let v be the other endpoint of the path. Then, $d^{\text{pen}}(G') \geq \text{dist}_{G'}^{\text{pen}}(x_0, v) > s'$ —a contradiction.

Case 2: $s' < d^{\text{pen}}(G')$. We will show that $d^{\text{pen}}(G) \leq d^{\text{pen}}(G')$. We first define $V_C = \{x_0, x_1, \dots, x_{a-1}\}$ to be the set of vertices in C . Again, let $w, w' \in V$ such that $d^{\text{pen}}(G) = \text{pen}(w) + \text{dist}_G(w, w') + \text{pen}(w')$ and we make a case distinction on the size of $\{w, w'\} \cap V_C$.

Subcase 1: $|\{w, w'\} \cap V_C| = 0$. Since G and G' only differ in C , we have

$$\begin{aligned} d^{\text{pen}}(G) &= \text{pen}(w) + \text{dist}_G(w, w') + \text{pen}(w') \\ &= \text{pen}'(w) + \text{dist}_{G'}(w, w') + \text{pen}'(w') \leq d^{\text{pen}}(G'). \end{aligned}$$

Subcase 2: $|\{w, w'\} \cap V_C| = 2$. In this case by definition of s' , we have that $s' = d_G^{\text{pen}} \geq d_{G'}^{\text{pen}}$ —a contradiction.

Subcase 3: $|\{w, w'\} \cap V_C| = 1$. We assume without loss of generality that $w \notin V_C$ and $w' \in V_C$. Then we have

$$\begin{aligned} d^{\text{pen}}(G) &= \text{pen}(w) + \text{dist}_G(w, w') + \text{pen}(w') \\ &\leq \text{pen}(w) + \text{dist}_G(w, x_0) + \max\{\text{pen}(x_0), \text{pen}(w') + \text{dist}_G(\{x_0, w'\})\} \\ &= \text{pen}'(w) + \text{dist}_{G'}(w, x_0) + \text{pen}'(x_0) \leq d^{\text{pen}}(G'). \end{aligned}$$

This finishes the last case and concludes the proof. □

We now analyze the running time of Reduction Rules 1 and 2.

Lemma 4 *Given a pending cycle $C = x_0x_1 \dots x_a$, Reduction Rule 2 can be applied in $O(a)$ time.*

Proof First, in $O(a)$ time we compute k such that $\text{dist}(x_k, x_0) + \text{pen}(x_k)$ is maximized and if $k \neq 0$, then we set $s = \max\{s, \text{pen}(x_0) + \text{dist}(x_k, x_0) + \text{pen}(x_k)\}$. (For $k = 0$ we do not update s .) It remains to show how to compute $d^{\text{pen}}(C)$, the longest shortest path that starts and ends in C . To this end, we first compute the sum W of all edge-weights in C , that is, $W = \sum_{i=0}^{a-1} \tau(\{x_i, x_{i+1}\})$.

Next we define two distance measures d_{cl}, d_{c-c} (for clockwise and counter-clockwise) such that

$$\begin{aligned}
 d_{cl}(x_i, x_j) &= \tau(\{x_i, x_{i+1 \bmod a}\}) \\
 &\quad + \tau(\{x_{i+1 \bmod a}, x_{i+2 \bmod a}\}) + \dots + \tau(\{x_{j-1 \bmod a}, x_j\}) \quad \text{and} \\
 d_{c-c}(x_i, x_j) &= \tau(\{x_i, x_{i-1 \bmod a}\}) \\
 &\quad + \tau(\{x_{i-1 \bmod a}, x_{i-2 \bmod a}\}) + \dots + \tau(\{x_{j+1 \bmod a}, x_j\}).
 \end{aligned}$$

Note that $d_{cl}(x_i, x_j) + d_{c-c}(x_i, x_j) = W$ and $d_{cl}(x_i, x_j) = d_{c-c}(x_j, x_i)$.

We provide a dynamic program that only considers “clockwise” shortest paths between x_ℓ and x_j , that is, paths of length $\text{pen}(x_\ell) + d_{cl}(x_\ell, x_j) + \text{pen}(x_j)$ that satisfy $d_{cl}(x_\ell, x_j) \leq d_{c-c}(x_\ell, x_j)$ (otherwise it is not a shortest path). Observe that all “counter-clockwise” paths will be considered in the iteration where the role of x_j and x_ℓ is swapped as $d_{cc}(x_\ell, x_j) = d_{cl}(x_\ell, x_j)$.

The dynamic program uses a table T with a entries, where the ℓ^{th} entry corresponds to x_ℓ and the value stored in the entry is the vertex x_j furthest from x_ℓ , formally,

$$x_j := \arg \max_{x \in \{x_i \mid d_{cl}(x_\ell, x_i) \leq d_{c-c}(x_\ell, x_i)\}} \{\text{dist}(x, x_\ell)\}.$$

For initialization, we start with computing $T[x_0]$ by checking in $O(a)$ time all vertices in C . Besides the table T , the dynamic program has one more variable r storing the length of a longest shortest path found so far. Initially, $r = \text{pen}(x_0) + \text{dist}(x_k, x_0) + \text{pen}(x_k)$.

Given $x_j = T[x_\ell]$ for some vertex x_ℓ the dynamic program computes the furthest vertex $x_{j'}$ from $x_{\ell+1}$ and updates r if any longest shortest path from $x_{\ell+1}$ is longer than r . Note that the furthest vertex $x_{j'}$ from $x_{\ell+1}$ is either the furthest vertex $T[x_\ell] = x_j$ from x_ℓ or some vertex x that is ignored by x_ℓ . The only possible vertices that are ignored by x_ℓ but not by $x_{\ell+1}$ are the vertices x with $d_{cl}(x_\ell, x) > d_{c-c}(x_\ell, x)$ and $d_{cl}(x_{\ell+1}, x) \leq d_{c-c}(x_{\ell+1}, x)$. Thus, we can compute the furthest vertex from $x_{\ell+1}$ in constant amortized time as follows: We can compute the furthest vertex $x_{j'}$ from $x_{\ell+1}$ by iterating over the vertices $x_{j+1 \bmod a}, x_{j+2 \bmod a}, \dots$ and check whether

$$\begin{aligned}
 &d_{c-c}(x_{\ell+1}, x_{j+1 \bmod a}) \\
 &= d_{c-c}(x_\ell, x_j) - \text{dist}(x_\ell, x_{\ell+1}) + \text{dist}(x_j \bmod a, x_{j+1 \bmod a}) \leq W/2.
 \end{aligned}$$

If this first check is met, then we compute the “pen”-distance $d_{c-c}(x_{\ell+1}, x_{k+1 \bmod a}) + \text{pen}(x_{\ell+1}) + \text{pen}(x_{k+1 \bmod a})$. If this is larger than r , then we update r with this value (a longer shortest path was found). We then continue with $x_{k+2 \bmod a}$ and so on until the first check is not met anymore.

The whole pending cycle can be checked in $O(a)$ time in this way and we can set $s = \max\{s, r\}$. □

We now analyze the running time of both reduction rules.

Lemma 5 *Reduction Rules 1 and 2 can be exhaustively applied in $O(n + m)$ time.*

Proof Notice that we can sort all vertices by their degree in linear time using bucket sort. Applying Reduction Rule 1 or Reduction Rule 2 takes constant time per deleted vertex. After applying a reduction rule, we adjust the degree of the remaining vertex (either the unique neighbor of a degree-one vertex or the high-degree vertex in a pending cycle) in constant time by moving it to the appropriate bucket. Note that applying Reduction Rule 2 can lead to a new vertex of degree one and an application of Reduction Rule 1 can lead to two maximal paths merging to either a longer maximal path or a pending cycle. Since these cases can be detected in constant time and each vertex is only removed once, the overall running time to apply Reduction Rules 1 and 2 exhaustively is in $O(n + m)$. \square

We now present the algorithm that computes the maximum $\text{dist}^{\text{pen}}(u, v)$ over all pairs of remaining vertices u, v after applying Reduction Rules 1 and 2 exhaustively. This algorithm distinguishes between three different cases: The longest shortest path has at least one endpoint in $V^{\geq 3}$ (Case 1), its two endpoints are in the same maximal path (Case 2), or its endpoints are in two different maximal paths (Case 3).

Proof (of Theorem 3) Let $G = (V, E)$ be the input graph with feedback edge number k and let K be a feedback edge set with $|K| = k$. We first apply Reduction Rules 1 and 2 exhaustively in $O(n + m)$ time. We next compute three different values: The length of a longest shortest path with

- at least one endpoint in $V^{\geq 3}$ (Case 1),
- both endpoints in the same maximal path (Case 2), or
- both endpoints in two different maximal paths (Case 3).

The diameter of G is then the maximum of s and the three values computed.

Case 1: We first perform Dijkstra’s algorithm from each vertex $v \in V^{\geq 3}$ and store for each vertex $u \in V \setminus \{v\}$ the distance $\text{dist}(v, u)$ and update $s = \max\{s, \text{pen}(v) + \text{pen}(u) + \text{dist}(v, u)\}$. This way we find all shortest paths that start or end in a vertex in $V^{\geq 3}$ (or a pending tree connected to such a vertex).

Case 2: This case is similar to the case of pending cycles (see Reduction Rule 2). The only adjustment is the computation of the index that is considered by $x_{\ell+1}$ but not by x_ℓ . For a maximal path $P = x_0x_1 \dots x_a$, we compute $W = \sum_{i=0}^{a-1} \text{dist}(x_i, x_{i+1})$ and check whether the distance “within” a path between two vertices x_i, x_j ($i < j$) is at most as large as $\text{dist}(x_i, x_0) + \text{dist}(x_0, x_a) + \text{dist}(x_a, x_j)$.

Case 3: We remark that this case can be solved in $O(k \cdot (m + n \log^3 n))$ time using range trees [18, Proposition 2]. In order to achieve a time of $O(kn \log n)$, we present a specialized algorithm (not using any black-box techniques). We set $V_P := \{x_1, x_2, \dots, x_{a-1}\}$ and $\overline{V}_P := V \setminus (V_P \cup \{x_0, x_a\}) = \{v_1, v_2, \dots, v_{n-a-1}\}$. In the last case we have that u is in a maximal path $P = x_0x_1 \dots x_a$ and v is outside P , that is, $u \in V_P$ and $v \in \overline{V}_P$. We present an algorithm that takes $O(n \log n)$ time for each maximal path to compute the length of a longest shortest path of the specified type. As there are $O(k)$ such maximal paths [5, Lemma 5], the overall running time is $O(k \cdot n \log n)$.

The algorithm uses a length- $|\overline{V}_P|$ array D where the i^{th} entry is the distance difference of $v_i \in \overline{V}_P$ to x_0 and x_a respectively, formally, $D[i] := \text{dist}_G(x_0, v_i) -$

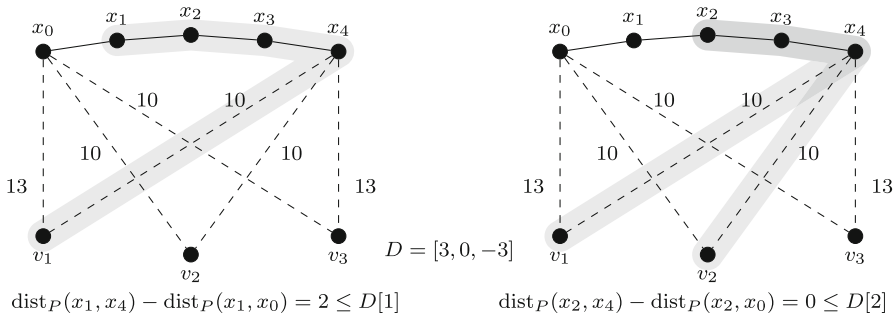


Fig. 4 Example demonstrating the monotonicities used in the proof of Theorem 3. All weights that are not displayed are 1 and all pen weights are 0. Observe that only for $i = 1$ a shortest x_1 - v_i -path goes over x_4 (see highlighted path on the left). The fact that a shortest x_1 - v_i -path goes over x_4 if and only if $\text{dist}_P(x_1, x_4) - \text{dist}_P(x_1, x_0) \leq D[i]$ can also be seen in the example: $D[2] < \text{dist}_P(x_1, x_4) - \text{dist}_P(x_1, x_0) = 3 - 1 \leq D[1]$. Exchanging x_1 with x_2 as starting point, results in more shortest x_2 - v_i -paths going over x_4 (see the highlighted paths on the right with x_2 as starting point)

$\text{dist}_G(x_a, v_i)$. Note that for some vertex x_j in P , there is a shortest x_j - v_i -path leaving P via x_a if and only if $\text{dist}_P(x_j, x_a) - \text{dist}_P(x_j, x_0) \leq D[i]$. Furthermore, D can be computed in $O(n)$ time from the distances computed in Case 1. The values $\text{dist}_P(x_j, x_a)$ and $\text{dist}_P(x_j, x_0)$ can also be computed easily in $O(n)$ time.

We use D in the following way: The algorithm sorts D in $O(n \log n)$ time in non-increasing order (for ease of notation, we still assume that the i^{th} entry of D correspond to v_i). As a result, we have that if a shortest x_j - v_i -path leaves P via x_a , then so does every shortest x_j - $v_{i'}$ -path for every $i' < i$. Furthermore, since for any $j' > j$ we have $\text{dist}_P(x_{j'}, x_a) - \text{dist}_P(x_{j'}, x_0) < \text{dist}_P(x_j, x_a) - \text{dist}_P(x_j, x_0) \leq D[i]$, we have that every shortest $x_{j'}$ - v_i -path goes via x_a . See Fig. 4 for an illustration of this monotonicity which is exploited in our subsequent algorithm.

The algorithm handles two cases separately: One for computing the longest shortest x_j - v_i -path, $x_j \in V_P$ and $v_i \in \bar{V}_P$, that contains x_0 and one for computing longest shortest x_j - v_i -path containing x_a . As these two cases are completely symmetric, we will discuss only the latter case. For brevity, let $\text{dist}_{\max}(x_j)$ be the length of a longest shortest path starting in x_j , leaving P via x_a , and ending in some $v \in \bar{V}_P$. Formally, $\text{dist}_{\max}(x_j) = \max\{\text{dist}^{\text{pen}}(x_j, v_i) \mid v_i \in \bar{V}_P \wedge \text{dist}_G(x_j, v_i) = \text{dist}_P(x_j, x_a) + \text{dist}_G(x_a, v_i)\}$. Thus, the task is to compute $\max_{j \in [a-1]} \{\text{dist}_{\max}(x_j)\}$. To this end, the algorithm computes $\text{dist}_{\max}(x_j)$ for all j .

For the initialization, the algorithm computes the sorted array D . Moreover, it computes the largest number $i_1 \in [n-a-1]$ such that $\text{dist}_G(x_1, v_{i_1}) = \text{dist}_P(x_1, x_a) + \text{dist}_G(x_a, v_{i_1})$. If no such number exists, then set $i_1 := 0$. Furthermore, for each $i \in [i_1]$ compute $\text{dist}^{\text{pen}}(x_1, v_i) = \text{pen}(v_i) + \text{dist}_G(v_i, x_a) + \text{dist}_P(x_a, x_1) + \text{pen}(x_1)$ and store the maximum in a variable r (r will be returned at the end of the algorithm). Due to D being sorted, this initialization phase can be done in $O(i_1)$ time. Moreover, due to D being sorted, we have $r = \text{dist}_{\max}(x_1)$ as for all $i' > i_1$ every shortest x_1 - $v_{i'}$ -path leaves P via x_0 . This completes the initialization.

Next, the algorithm computes for each $j \in \{2, 3, \dots, a-1\}$ the value $\text{dist}_{\max}(x_j)$. Notice that $\text{dist}_{\max}(x_1)$ was computed in the initialization. For $j > 1$ the algorithm is

as follows: Compute the largest number $i_j \in [n - a - 1]$ such that $\text{dist}_G(x_j, v_{i_j}) = \text{dist}_P(x_j, x_a) + \text{dist}_G(x_a, v_{i_j})$. Note that due to the sorting of D we have that $i_j \geq i_{j-1}$. Hence, we find i_j in $O(i_j - i_{j-1})$ time by simply start checking D at positions $i_{j-1} + 1, i_{j-1} + 2, \dots, i_j, i_j + 1$ (note that, by definition of i_j , the last check at position $i_j + 1$ fails). For each $i \in \{i_{j-1} + 1, i_{j-1} + 2, \dots, i_j\}$ we do the following: We first compute $\text{dist}^{\text{pen}}(x_j, v_i) = \text{pen}(v_i) + \text{dist}_G(v_i, x_a) + \text{dist}_P(x_a, x_j) + \text{pen}(x_j)$ and store the maximum in a variable r' . We then update r with $\max\{r', r - \text{pen}(x_{j-1}) + \text{pen}(x_j) - \tau(\{x_{j-1}, x_j\})\}$. Observe that $r = \text{dist}_{\max}(x_j)$ as for v_i with $i \in \{i_{j-1} + 1, i_{j-1} + 2, \dots, i_j\}$ the algorithm computed $\text{dist}^{\text{pen}}(x_j, v_i)$. For all $i \in [i_{j-1}]$ we know that all $x_{j-1} - v_i$ -paths leave P via x_a . Thus, we can simply update their length by $\text{pen}(x_j) - \text{pen}(x_{j-1}) - \tau(\{x_{j-1}, x_j\})$.

Altogether, the algorithm runs in $O(k(n \log n + \sum_{i=1}^{a-1} (i_j - i_{j-1}))) = O(kn \log n)$ time. Combining this with Lemma 5 concludes the proof of Theorem 3. \square

5 Parameters for Social Networks

Here, we study parameters that we expect to be small in social networks. Recall that social networks have the “small-world” property and a power-law degree distribution [33, 35–38]. The “small-world” property directly transfers to the diameter. We capture the power-law degree distribution by the h -index as only few high-degree vertices exist in the network. Thus, we investigate parameters related to the diameter and to the h -index starting with degree-related parameters.

5.1 Degree Related Parameters

We next investigate the parameter minimum degree. Unsurprisingly, the minimum degree is not helpful for parameterized algorithms. In fact, we show that DIAMETER is 2-GP-hard with respect to the combined parameter bisection width and minimum degree. The bisection width of a graph G is the minimum number of edges to delete from G in order to partition G into two connected component whose number of vertices differ by at most one.

Proposition 2 *DIAMETER is 2-GP-hard with respect to bisection width and minimum degree.*

Proof Let $G = (V, E)$ be an arbitrary input graph for DIAMETER where $V = \{v_1, v_2, \dots, v_n\}$ and let d be the diameter of G . We construct a new graph $G' = (V', E')$ with diameter $d + 4$ as follows: Let $V' = \{s_i, t_i, u_i \mid i \in [n]\} \cup \{w_i \mid i \in [3n]\}$ and $E' = T \cup W \cup E''$, where $T = \{\{s_i, t_i\}, \{t_i, u_i\} \mid i \in [n]\}$, $W = \{u_1, w_1\} \cup \{\{w_1, w_i\} \mid i \in ([3n] \setminus \{1\})\}$, and $E'' = \{\{u_i, u_j\} \mid \{v_i, v_j\} \in E\}$.

An example of this construction can be seen in Fig. 5.

We will now prove that all properties of Definition 1 hold. It is easy to verify that the reduction runs in linear time and that there are $6n$ vertices and $5n + m$ edges in G' . Notice that $\{s_i, t_i, u_i \mid i \in [n]\}$ and $\{w_i \mid i \in [3n]\}$ are both of size $3n$ and that there

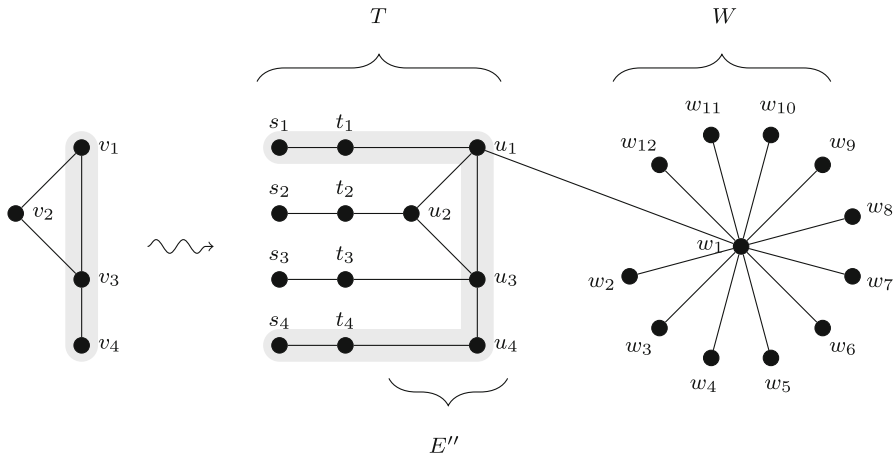


Fig. 5 Example for the construction in the proof of Theorem 2. The input graph given on the left side has diameter 2 and the constructed graph on the right side has diameter $2 + 4 = 6$

is only one edge ($\{u_1, w_1\}$) between these two sets of vertices. The bisection width of G' is therefore one and the minimum degree is also one as s_1 is only adjacent to t_1 .

It remains to show that G' has diameter $d + 4$. First, notice that the subgraph of G' induced by $\{u_i \mid i \in [n]\}$ is isomorphic to G . Note that $\text{dist}(s_i, u_i) = 2$ and thus $\text{dist}(s_i, s_j) = \text{dist}(u_i, u_j) + 4 = \text{dist}(v_i, v_j) + 4$ and therefore the diameter of G' is at least $d + 4$. Third, notice that for all vertices $x \in V' \setminus \{s_i\}$ it holds that $\text{dist}(s_i, x) > \text{dist}(t_i, x)$. Lastly, observe that for all $i \in [3n]$ and all vertices $x \in V'$ it holds that $\text{dist}(w_i, x) \leq \max\{\text{dist}(s_1, x), 4\}$. Thus the longest shortest path in G' is between two vertices s_i, s_j and is of distance $\text{dist}(u_i, u_j) + 4 = \text{dist}(v_i, v_j) + 4 \leq d + 4$. \square

We mention in passing that the constructed graph in the proof of Proposition 2 contains the original graph as an induced subgraph and if the original graph is bipartite, then so is the constructed graph. Thus, first applying the construction in the proof of Theorem 1 (see also Fig. 2) and then the construction in the proof of Proposition 2 proves that DIAMETER is GP-hard even parameterized by the sum of girth, bisection width, minimum degree, and odd cycle transversal.

5.2 Parameters Related to Both Diameter and h -index

Here, we will study combinations of two parameters where the first one is related to diameter and the second to h -index (see Fig. 1 for an overview of closely related parameters). We start with the combination maximum degree and diameter. Interestingly, although the parameter is quite large, the naive algorithm behind Observation 5 cannot be improved to a fully polynomial running time.

Theorem 4 *There is no $(d + \Delta)^{O(1)}(n + m)^{2-\epsilon}$ -time algorithm that solves DIAMETER parameterized by maximum degree Δ and diameter d unless the SETH is false.*

Proof We prove a slightly stronger statement excluding $2^{o(\sqrt{d+\Delta})} \cdot (n+m)^{2-\epsilon}$ -time algorithms for some constant c . Assume towards a contradiction that for each constant r there is a $2^{o(\sqrt{d+\Delta})} \cdot (n+m)^{2-\epsilon}$ -time algorithm that solves DIAMETER parameterized by maximum degree Δ and diameter d . Evald and Dahlgaard [21] have shown a reduction from CNF-SAT to DIAMETER where the resulting graph has maximum degree three such that for any constant $\epsilon > 0$ an $O((n+m)^{2-\epsilon})$ -time algorithm (for DIAMETER) would refute the SETH. A closer look reveals that there is some constant c such that the diameter d in their constructed graph is in $O(\log^c(n+m))$. By assumption we can solve DIAMETER parameterized by maximum degree and diameter in $2^{o(\sqrt{d+\Delta})} \cdot (n+m)^{2-\epsilon}$ time. Observe that

$$\begin{aligned} 2^{o(\sqrt{d+\Delta})} \cdot (n+m)^{2-\epsilon} &= 2^{o(\sqrt{\log^c(n+m)})} \cdot (n+m)^{2-\epsilon} \\ &= (n+m)^{o(1)} \cdot (n+m)^{2-\epsilon} \subseteq O((n+m)^{2-\epsilon'}) \text{ for some } \epsilon' > 0. \end{aligned}$$

Since we constructed for some $\epsilon' > 0$ an $O((n+m)^{2-\epsilon'})$ -time algorithm for DIAMETER the SETH fails and thus we reached a contradiction. Finally, notice that $(d+\Delta)^{O(1)} \subset 2^{o(\sqrt{d+\Delta})}$ for any constant c . □

h-index and diameter. We next investigate the combined parameter *h-index* and diameter. The reduction by Roditty and Williams [39] produces instances with constant domination number and logarithmic vertex cover number (in the input size). Since the diameter d is linearly upper-bounded by the domination number and the *h-index* is linearly upper-bounded by the vertex cover number, any algorithm that solves DIAMETER parameterized by the combined parameter $(d+h)$ in $2^{o(d+h)} \cdot (n+m)^{2-\epsilon}$ time disproves the SETH. We will now present an algorithm for WEIGHTED DIAMETER parameterized by *h-index* and diameter that almost matches the lower bound.

Theorem 5 DIAMETER parameterized by diameter d and *h-Index* h is solvable in $O(h \cdot (n \log n + m) + n \cdot d \cdot h \cdot (d^h + h^d \log h))$ time.

Proof Let $H = \{x_1, \dots, x_h\}$ be a set of vertices such that all vertices in $V \setminus H$ have degree at most h in G . Clearly, H can be computed in linear time. We will describe a two-phase algorithm with the following basic idea: In the first phase it performs Dijkstra’s algorithm from each vertex $v \in H$, stores the distance to each other vertex and uses this to compute the “type” of each vertex, that is, a characterization by the distance to each vertex in H . In the second phase it iteratively increases a value e and verifies whether there is a vertex pair of distance at least $e + 1$. If at any point no vertex pair is found, then the diameter of G is e .

The first phase is straight forward: Execute Dijkstra’s algorithm from each vertex v in H and store the distance from v to every other vertex w in a table. Then iterate over each vertex $w \in V \setminus H$ and compute a vector of length h where the i th entry represents the distance from w to x_i . Also store the number of vertices of each type containing at least one vertex. Since the distance to any vertex is at most d , there are at most d^h different types. This first phase takes $O(h \cdot (m + n \log n))$ time.

For the second phase, we initialize e with the largest distance found so far, that is, the maximum value stored in the table and compute $G' = G - H$. Iteratively check

whether there is a pair of vertices in $V \setminus H$ of distance at least $e + 1$ as follows. We check for each vertex $v \in V \setminus H$ whether there are types such that no vertex of one of these types can be reached by a path of length at most e passing through a vertex in H . This can be done by computing the sum of the two type-vectors in $O(h)$ time and comparing the minimum entry in this sum with e . If all entries are larger than e , then no shortest path from v to some vertex w of the respective type of length at most e can contain any vertex in H . Thus we compute Dijkstra’s algorithm from v in G' up to depth e^2 and count the number of vertices of the respective types we found. If these numbers are equal to the total number of vertices of the respective types, then for all vertices w of these type it holds that $\text{dist}(v, w) \leq e$. If the respective numbers do not match, then there is a vertex pair of distance at least $e + 1$, and we can therefore increase e by one and start the process again.

There are at most d iterations in which e is increased and the check is done. In each iteration, we have to compute the sum of type vectors for each vertex and perform Dijkstra’s algorithm up to depth at most d in G' . Recall that the maximum degree in G' is h and therefore computing Dijkstra’s algorithm up to depth d takes $O(h^d \cdot d \cdot \log h)$ time. Since $\sum_{e=1}^d h^e < h^{d+1}$ for $h \geq 2$, the overall running time is in $O(h \cdot (n \log n + m) + n \cdot d \cdot h \cdot (d^h + h^d \log h))$. \square

Acyclic chromatic number and domination number: We next analyze the parameterized complexity of DIAMETER parameterized by acyclic chromatic number a and domination number d . The acyclic chromatic number upper-bounds the average degree, and therefore the standard $O(n \cdot m)$ -time algorithm runs in $O(n^2 \cdot a)$ time. We will show that this is essentially the best one can hope for as we can exclude $f(a, d) \cdot (n + m)^{2-\epsilon}$ -time algorithms assuming SETH. Our result is based on the reduction by Roditty and Williams [39] and is modified such that the acyclic chromatic number and domination number are both four in the resulting graph.

Theorem 6 *There is no $f(a, d) \cdot (n + m)^{2-\epsilon}$ -time algorithm for any computable function f that solves DIAMETER parameterized by acyclic chromatic number a and domination number d unless the SETH is false.*

Proof We provide a reduction from CNF- SAT to DIAMETER where the input instance has constant acyclic chromatic number and domination number and such that an $O((n + m)^{2-\epsilon})$ -time algorithm refutes the SETH. Since the idea is the same as in Roditty and Williams [39] we refer the reader to their work for more details. Let ϕ be a CNF- SAT instance with variable set W and clause set C . Assume without loss of generality that $|W|$ is even. We construct an instance $(G = (V, E), k)$ for DIAMETER as follows:

Arbitrarily partition W into two set W_1, W_2 of equal size. Add three sets V_1, V_2 and B of vertices to G where each vertex in V_1 (in V_2) represents one of $2^{|W_1|} = 2^{|W_2|}$ possible assignments of the variables in W_1 (in W_2) and each vertex in B represents a clause in C . Clearly $|V_1| + |V_2| = 2 \cdot 2^{|W|/2}$ and $|B| = |C|$. For each $v_i \in V_1$ and

² By “up to depth e ” we mean that we run Dijkstra’s algorithm with the addition that whenever the distance to a vertex is at least e , then we do not add it to the stack (or priority queue) and if the distance is larger then e , then we do not update its distance to the source. Similar as in the proof of Observation 5, we can show that the number of vertices and edges considered by the algorithm are at most $h^e + 1$ and h^e , respectively.

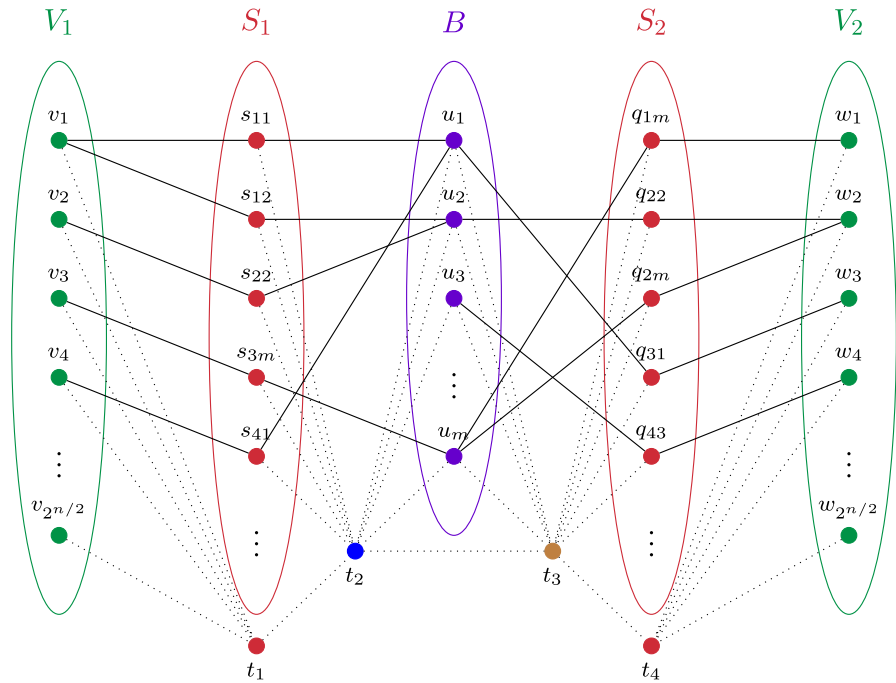


Fig. 6 A schematic illustration of the construction in the proof of Theorem 6. Note that the resulting graph has acyclic chromatic number five ($V_1 \cup V_2$, B , $S_1 \cup S_2 \cup \{t_1, t_4\}$, $\{t_2\}$ and $\{t_3\}$, also represented by colors) and a dominating number four ($\{t_1, t_2, t_3, t_4\}$) (Color figure online)

each $u_j \in B$ we add a new vertex s_{ij} and the two edges $\{v_i, s_{ij}\}$ and $\{u_j, s_{ij}\}$ to G if the respective variable assignment does *not* satisfy the respective clause. We call the set of all these newly introduced vertices S_1 . Now repeat the process for all vertices $w_i \in V_2$ and all u_j in B and call the newly introduced vertices q_{ij} and the set S_2 . Finally we add four new vertices t_1, t_2, t_3, t_4 and the following sets of edges to G : $\{\{t_1, v\} \mid v \in V_1\}$, $\{\{t_2, s\} \mid s \in S_1\}$, $\{\{t_3, q\} \mid q \in S_2\}$, $\{\{t_4, w\} \mid w \in V_2\}$, $\{\{t_2, b\}, \{t_3, b\} \mid b \in B\}$, and $\{\{t_1, t_2\}, \{t_2, t_3\}, \{t_3, t_4\}\}$. See Fig. 6 for a schematic illustration of the construction.

We will first show that ϕ is satisfiable if and only if G has diameter five and then show that the domination number and acyclic chromatic number of G are five and four, respectively. First assume that ϕ is satisfiable. Then, there exists some assignment β of the variables such that all clauses are satisfied, that is, the two assignments of β with respect to the variables in W_1 and W_2 satisfy all clauses. Let $v_1 \in V_1$ and $v_2 \in V_2$ be the vertices corresponding to β . Thus for each $b \in B$ we have $\text{dist}(v_1, b) + \text{dist}(v_2, b) \geq 5$. Observe that all paths from a vertex in V_1 to a vertex in V_2 that do not pass through a vertex in B pass through t_2 and t_3 . Since all of these paths are of length 5, it follows that $\text{dist}(v_1, v_2) = 5$. Observe that the diameter of G is at most five since each vertex is connected to some vertex in $\{t_1, t_2, t_3, t_4\}$ and these four are of pairwise distance at most three.

Assume next that the diameter of G is five. Clearly there is a shortest path between a vertex $v_i \in V_1$ and $v_j \in V_2$ of length five. Thus there is no path of

the form $v_i s_i h u_h q_j h w_j$ for any $u_h \in B$. This corresponds to the statement that the variable assignment of v_i and w_j satisfy all clauses and therefore ϕ is satisfiable.

The domination number of G is four since $\{t_1, t_2, t_3, t_4\}$ is a dominating set. The acyclic chromatic number of G is at most five as $V_1 \cup V_2, B, S_1 \cup S_2 \cup \{t_1, t_4\}, \{t_2\}$ and $\{t_3\}$ each induce an independent set and each combination of them not including $S_1 \cup S_2 \cup \{t_1, t_4\}$ only induce independent sets or stars. Lastly, note that $S_1 \cup S_2 \cup \{t_1, t_4\}$ and $\{t_2\}$ or $\{t_3\}$ only induces a star and an independent set, $S_1 \cup S_2 \cup \{t_1, t_4\}$ and $V_1 \cup V_2$ induces two trees of depth 2 (where t_1 and t_4 are the roots and S_1 and S_2 are the leaves), and $S_1 \cup S_2 \cup \{t_1, t_4\}$ and B induce a disjoint union of stars and isolated vertices as each vertex in $S_1 \cup S_2 \cup \{t_1, t_4\}$ has maximum degree one in $G[B \cup S_1 \cup S_2 \cup \{t_1, t_4\}]$.

Now assume that we have an $O(f(k) \cdot (n + m)^{2-\epsilon})$ -time algorithm for DIAMETER parameterized by domination number and acyclic chromatic number. Since the constructed graph has $O(2^{|W|/2} \cdot |C|)$ vertices and edges, this would imply an algorithm with running time

$$\begin{aligned} &O(f(9) \cdot (2^{|W|/2} \cdot |C|)^{2-\epsilon}) \\ &= O(2^{(|W|/2)(2-\epsilon)} \cdot |C|^{2-\epsilon}) \\ &= O(2^{|W|(1-\epsilon/2)} \cdot |C|^{2-\epsilon}) \\ &= 2^{|W|(1-\epsilon')} \cdot (|C| + |W|)^{O(1)} \text{ for some } \epsilon' > 0. \end{aligned}$$

Hence, such an algorithm for DIAMETER would refute the SETH. □

6 Conclusion

We have resolved the complexity status of DIAMETER for most of the parameters in the complexity landscape shown in Fig. 1. However, several open questions remain. For example, is there an $f(k)n^2$ -time algorithm with respect to the parameter diameter? Moreover, our algorithms working with parameter combinations have mostly impractical running times which, assuming SETH, cannot be improved by much. So the question arises, whether there are parameters k_1, \dots, k_ℓ that allow for practically relevant running times like $\prod_{i=1}^\ell k_i \cdot (n + m)$ or even $(n + m) \cdot \sum_{i=1}^\ell k_i$? The list of parameters displayed in Fig. 1 is by no means exhaustive. Hence, the question arises which other parameters are small in typical scenarios? For example, what is a good parameter capturing the special community structures of social networks [26]?

Acknowledgements This work was partially supported by DFG project FPTinP, NI 369/16.

Funding Open Access funding enabled and organized by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted

by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Abboud, A., Williams, V.V., Wang, J.R.: Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In: Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '16), pp. 377–391. SIAM (2016)
2. Aingworth, D., Chekuri, C., Indyk, P., Motwani, R.: Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM J. Comput.* **28**(4), 1167–1181 (1999)
3. Backurs, A., Roditty, L., Segal, G., Williams, V.V., Wein, N.: Toward tight approximation bounds for graph diameter and eccentricities. *SIAM J. Comput.* **50**(4), 1155–1199 (2021)
4. Bentert, M., Fluschnik, T., Nichterlein, A., Niedermeier, R.: Parameterized aspects of triangle enumeration. *J. Comput. Syst. Sci.* **103**, 61–77 (2019)
5. Bentert, M., Dittmann, A., Kellerhals, L., Nichterlein, A., Niedermeier, R.: An adaptive version of Brandes' algorithm for betweenness centrality. *J. Gr. Algorithms Appl.* **24**(3), 483–522 (2020)
6. Borassi, M., Crescenzi, P., Habib, M., Kusters, W.A., Marino, A., Takes, F.W.: Fast diameter and radius BFS-based computation in (weakly connected) real-world graphs: with an application to the six degrees of separation games. *Theoret. Comput. Sci.* **586**, 59–80 (2015)
7. Borassi, M., Crescenzi, P., Trevisan, L.: An axiomatic and an average-case analysis of algorithms and heuristics for metric properties of graphs. In: Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 17), pp. 920–939. SIAM (2017)
8. Brandstädt, A., Van Bang L., Spinrad, J.P.: Graph classes: a survey, volume 3 of SIAM Monographs on Discrete Mathematics and Applications. SIAM (1999)
9. Bretscher, A., Corneil, D.G., Habib, M., Paul, C.: A simple linear time LexBFS cograph recognition algorithm. *SIAM J. Discret. Math.* **22**(4), 1277–1296 (2008)
10. Bringmann, K., Husfeldt, T., Magnusson, M.: Multivariate analysis of orthogonal range searching and graph distances. *Algorithmica* **82**(8), 2292–2315 (2020)
11. Cairo, M., Grossi, R., Rizzi, R.: New bounds for approximating extremal distances in undirected graphs. In: Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '16), pp. 363–376. SIAM (2016)
12. Cao, Y.: Linear recognition of almost interval graphs. In: Krauthgamer, R. (ed) Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '16), pp. 1096–1115. SIAM (2016)
13. Chan, T.M., Williams, R.R.: Deterministic APSP, orthogonal vectors, and more: quickly derandomizing Razborov-Smolensky. *ACM Trans. Algorithms* **17**(1), 2:1–2:14 (2021)
14. Chen, D.Z., Lee, D.T., Sridhar, R., Sekharan, C.N.: Solving the all-pair shortest path query problem on interval and circular-arc graphs. *Networks* **31**(4), 249–258 (1998)
15. Corneil, D.G., Perl, Y., Stewart, L.K.: A linear recognition algorithm for cographs. *SIAM J. Comput.* **14**(4), 926–934 (1985)
16. Coudert, D., Ducoffe, G., Popa, A.: Fully polynomial FPT algorithms for some classes of bounded clique-width graphs. *ACM Trans. Algorithms* **15**(3), 33:1–33:57 (2019)
17. Dalirrooyfard, M., Li, R., Williams, V.V.: Hardness of approximate diameter: now for undirected graphs. In: Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS '21), pp. 1021–1032. IEEE (2021)
18. Ducoffe, G.: A new application of orthogonal range searching for computing giant graph diameters. In: Proceedings of the 2nd Symposium on Simplicity in Algorithms (SOSA '19), pp. 12:1–12:7. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019)
19. Ducoffe, G.: Optimal centrality computations within bounded clique-width graphs. In: Proceedings of the 16th International Symposium on Parameterized and Exact Computation (IPEC '21), volume 214 of LIPIcs, pp. 16:1–16:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021)
20. Ducoffe, G., Habib, M., Viennot, L.: Diameter computation on H-minor free graphs and graphs of bounded (distance) VC-dimension. In: Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA '20), pp. 1905–1922. SIAM (2020)
21. Evald, J., Dahlgaard, S.: Tight hardness results for distance and centrality problems in constant degree graphs. *Computing Research Repository*, abs/1609.08403 (2016)

22. Fluschnik, T., Komusiewicz, C., Mertzios, G.B., Nichterlein, A., Niedermeier, R., Talmon, N.: When can graph hyperbolicity be computed in linear time? *Algorithmica* **81**(5), 2016–2045 (2019)
23. Fomin, F.V., Lokshtanov, D., Saurabh, S., Pilipczuk, M., Wrochna, M.: Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. *ACM Trans. Algorithms* **14**(3), 34:1–34:45 (2018)
24. Gawrychowski, P., Kaplan, H., Mozes, S., Sharir, M., Weimann, O.: Voronoi diagrams on planar graphs, and computing the diameter in deterministic $\tilde{O}(n^{5/3})$ time. *SIAM J. Comput.* **50**(2), 509–554 (2021)
25. Giannopoulou, A.C., Mertzios, G.B., Niedermeier, R.: Polynomial fixed-parameter algorithms: a case study for longest path on interval graphs. *Theoret. Comput. Sci.* **689**, 67–95 (2017)
26. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. *Proc. Natl. Acad. Sci.* **99**(12), 7821–7826 (2002)
27. Guo, J., Hüffner, F., Niedermeier, R.: A structural view on parameterizing problems: distance from triviality. In: *Proceedings of the 1st International Workshop on Parameterized and Exact Computation (IWPEC 04)*, volume 3162 of LNCS, pp. 162–173. Springer (2004)
28. Impagliazzo, R., Paturi, R.: On the complexity of k-sat. *J. Comput. Syst. Sci.* **62**(2), 367–375 (2001)
29. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.* **63**(4), 512–530 (2001)
30. Johnson, D.B.: Efficient algorithms for shortest paths in sparse networks. *J. ACM* **24**(1), 1–13 (1977)
31. Koana, T., Korenwein, V., Nichterlein, A., Niedermeier, R., Zschoche, P.: Data reduction for maximum matching on real-world graphs: theory and experiments. *ACM J. Exp. Algorithmics* **24**, 1–30 (2021)
32. Kratsch, S., Nelles, F.: Efficient and adaptive parameterized algorithms on modular decompositions. In: *Proceedings of the 26th Annual European Symposium on Algorithms (ESA' 18)*, volume 112 of LIPIcs, pp. 55:1–55:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2018)
33. Leskovec, J., Horvitz, E.: Planetary-scale views on a large instant-messaging network. In: *Proceedings of the 17th International World Wide Web Conference (WWW '08)*, pp. 915–924. ACM (2008). ISBN 978-1-60558-085-2
34. Mertzios, G.B., Nichterlein, A., Niedermeier, R.: The power of linear-time data reduction for maximum matching. *Algorithmica* **82**(12), 3521–3565 (2020)
35. Milgram, S.: The small world problem. *Psychol. Today* **1**, 61–67 (1967)
36. Newman, M.E.J.: The structure and function of complex networks. *SIAM Rev.* **45**(2), 167–256 (2003)
37. Newman, M.E.J.: *Networks: An Introduction*. Oxford University Press, Oxford (2010)
38. Newman, M.E.J., Park, J.: Why social networks are different from other types of networks. *Phys. Rev. E* **68**(3), 036122 (2003)
39. Roditty, L., Williams, V.V.: Fast approximation algorithms for the diameter and radius of sparse graphs. In: *Proceedings of the 45th Symposium on Theory of Computing Conference (STOC '13)*, pp. 515–524. ACM (2013)
40. Seidel, R.: On the all-pairs-shortest-path problem in unweighted undirected graphs. *J. Comput. Syst. Sci.* **51**(3), 400–403 (1995)
41. Sorge, M., Weller, M.: The graph parameter hierarchy. *Manuscript* (2013)
42. Sprague, A.P., Takaoka, T.: $O(1)$ query time algorithm for all pairs shortest distances on interval graphs. *Int. J. Found. Comput. Sci.* **10**(4), 465–472 (1999)
43. Weimann, O., Yuster, R.: Approximating the diameter of planar graphs in near linear time. *ACM Trans. Algorithms* **12**(1), 12:1–12:13 (2016)