Faster Approximation Algorithms for Computing Shortest Cycles on Weighted Graphs

Guillaume Ducoffe

FASTER APPROXIMATION ALGORITHMS FOR COMPUTING SHORTEST CYCLES ON WEIGHTED GRAPHS*

GUILLAUME DUCOFFE *†‡

Abstract. Given an n-vertex m-edge graph G with non-negative edge-weights, a *shortest cycle* is one minimizing the sum of the weights on its edges. The *girth* of G is the weight of a shortest cycle. We obtain several new approximation algorithms for computing the girth of weighted graphs:

- For any graph G with polynomially bounded integer weights, we present a deterministic algorithm that computes, in $\tilde{\mathcal{O}}(n^{5/3}+m)$-time[1], a cycle of weight at most twice the girth of G. This matches the approximation factor of the best known subquadratic-time approximation algorithm for the girth of *unweighted* graphs.
- Then, we turn our algorithm into a deterministic $(2+\varepsilon)$-approximation for graphs with arbitrary non-negative edge-weights, at the price of a slightly worse running-time in $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}(1/\varepsilon)+m)$. For that, we introduce a generic method in order to obtain a polynomial-factor approximation of the girth in subquadratic time, that may be of independent interest.
- Finally, if we assume that the adjacency lists are sorted then we can get rid off the dependency in the number m of edges. Namely, we can transform our algorithms into an $\tilde{\mathcal{O}}(n^{5/3})$-time *randomized* 4-approximation for graphs with non-negative edge-weights. This can be derandomized, thereby leading to an $\tilde{\mathcal{O}}(n^{5/3})$-time deterministic 4-approximation for graphs with polynomially bounded integer weights, and an $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}(1/\varepsilon))$-time deterministic $(4+\varepsilon)$-approximation for graphs with non-negative edge-weights.

To the best of our knowledge, these are the first known *subquadratic-time* approximation algorithms for computing the girth of weighted graphs.

Key words. Girth; Weighted Graphs; Approximation Algorithms.

AMS subject classifications. 05C85, 68Q25, 68W25, 68R10

1. Introduction.

The exciting program of "Hardness in P" aims at proving (under plausible complexity theoretic conjectures) the *exact* time-complexity of fundamental, polynomial-time solvable problems in computer science. In this paper, we consider the Girth problem on edge-weighted undirected graphs, for which almost all what is known in terms of finer-grained complexity only holds for dense graphs ($m=\Omega(n^2)$). We recall that the girth of a given graph G is the minimum weight of a cycle in G — with the weight of a cycle being defined as the sum of the weights on its edges (see Sec. 2 for any undefined terminology in this introduction). For dense graphs this parameter can be computed in time $\mathcal{O}(n^3)$, and Vassilevska Williams and Williams [20] proved a bunch of combinatorial subcubic equivalences between Girth and other path and matrix problems. In particular, for every $\varepsilon>0$, there cannot exist any *combinatorial* $(4/3-\varepsilon)$-approximation algorithm for Girth that runs in truly subcubic time unless there exists a truly subcubic combinatorial algorithm for multiplying two Boolean matrices. Roditty and Tov completed this above hardness result with an $\tilde{\mathcal{O}}(n^2/\varepsilon)$-time $(4/3+\varepsilon)$-approximation algorithm [16], thereby essentially completing the picture of what can be done combinatorially in subcubic time.

However, the story does not end here for at least two reasons. A first simple but important observation is that as the graphs considered get sparser, the complexity for computing their girth falls down to $\mathcal{O}(n^2)$. In fact, when the edge-weights are

*Results of this paper were partially presented at the ICALP'19 conference [9].

*National Institute for Research and Development in Informatics, Romania
†The Research Institute of the University of Bucharest ICUB, Romania
‡Faculty of Mathematics and Computer Science, University of Bucharest, Romania

[1]The $\tilde{\mathcal{O}}(\cdot)$ notation suppresses polylogarithmic factors.

integers bounded by some constant M, there is a non-combinatorial algorithm for computing the girth of *any* n-vertex graph G in time $\tilde{\mathcal{O}}(Mn^{\omega})$ where ω stands for the exponent of square matrix multiplication over a ring [17]. It is believed that $\omega = 2$ [14], and if true, that would imply we can compute the *exact* value of the girth in quasi *quadratic* time — at least when edge-weights are bounded. So far, all the *approximation* algorithms for GIRTH on weighted graphs run in $\tilde{\Omega}(n^2)$-time [11, 16]. This leads us to the following, natural research question:

Does there exist a subquadratic approximation algorithm for GIRTH *on weighted graphs?*

In this paper, we answer to this above question in the affirmative.

1.1. Our contributions. We present new approximation algorithms for the girth of graphs with non-negative real edge-weights. These are the first algorithms to break the quadratic barrier for this problem – at the price of a slightly worse approximation factor compared to the state of the art [16] — see Sec. 1.2 on the Related work for more details.

In what follows, we ignore the cost of arithmetic operations (for graphs with bounded integer edge-weights, this can only change the final running time by a factor polylog$M$; for graphs with real edge-weights, this is essentially equivalent to consider the real RAM model).

Our first result is obtained for graphs with bounded integer edge-weights.

THEOREM 1.1. *For every $G = (V, E, w)$ with edge-weights in $\{1, \ldots, M\}$, we can compute a deterministic 2-approximation for* GIRTH *in time $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}M + m)$.*

Our starting point for Theorem 1.1 is a previous 2-approximation algorithm from Lingas and Lundell [11], that runs in *quadratic* time. Specifically, these two authors introduced an $\tilde{O}(n \log M)$-time procedure that takes as entry a specified vertex of the graph and needs to be applied to *every* vertex in order to obtain the desired 2-approximation of the girth. Inspired by the techniques used for approximate distance oracles [19] we informally modify their algorithm as follows. We only apply their procedure to the vertices in a *random* subset S: where each vertex is present with equal probability $n^{-1/3}$ (we can derandomize our approach by using known techniques from the litterature [15]). Furthermore, for the vertices not in S, we rather apply a modified version of their procedure that is restricted to a small subgraph – induced by some ball of expected size $\mathcal{O}(n^{1/3})$. A careful analysis shows this is a 2-approximation.

The reason why this above approach works is that, when we run the procedure of Lingas and Lundell at some arbitrary vertex s, it will always detect a short cycle if there is one passing *close* to s (but not necessarily passing through s itself). This nice property has been noticed and exploited for related algorithms on *unweighted* graphs [11][2]. However, we think we are the first to prove such a property in the weighted case. We note that one of the two algorithms proposed by Roditty and Tov in [16] also satisfies such a property. We did not find a way to exploit their algorithm in order to improve our approximation factor.

Our approach for graphs with bounded integer edge-weights (see Sec. 3.2) is the cornerstone of all our other results in the paper. We considerably refine this approach

[2]There is a subtle difference between our approach for weighted graphs and the one formerly applied to unweighted graphs. Indeed, we need to consider all edges in the *subgraphs* that are induced by some small balls in the graph, that might include some large-weight edges not on any shortest-path in G. For unweighted graphs [11, 17], they mostly consider edges on some shortest-paths in G between a pre-defined vertex and the other vertices in the ball.

so that it also applies to graphs with *arbitrary* non-negative edge-weights.

THEOREM 1.2. *For every $\varepsilon > 0$ and $G = (V, E, w)$ with non-negative edge-weights, we can compute a deterministic $(2+\varepsilon)$-approximation for* GIRTH *in time $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}1/\varepsilon + m)$.*

We note that in [16], Roditty and Tov introduced a nice technique – that we partly reuse in this paper – in order to transpose their results for bounded integer-weights to arbitrary weights. However, we face several new difficulties, not encountered in [16], due to the need to perform all the intermediate operations in *subquadratic* time. As a side contribution of this work, we present an intricate modification of our approach for graphs with bounded integer edge-weights that we use in order to approximate the girth of graphs with arbitrary non-negative edge-weights up to a polynomial-factor. This subquadratic-time routine could be useful to anyone improving our result for the graphs with integer-weights in order to generalize their results to the graphs with non-negative real weights.

Our algorithms are subquadratic in the size of the graph, but they may be quadratic in its order n if there are $m = \Theta(n^2)$ edges. By a folklore application of Moore bounds, any unweighted graph with $\mathcal{O}(n^{1+\frac{1}{\ell}})$ edges has girth at most 2ℓ, and so, we can always output a constant upper-bound on the girth of moderately dense graphs. It implies that the dependency on m can always be removed in the running-time of approximation algorithms for the girth of *unweighted* graphs.

However, we can prove by using elementary arguments that any approximation algorithm for the girth on weighted graphs must run in $\Omega(m)$-time. Specifically, given $G = (V, E, w)$ with positive integer weights, add a fresh new vertex $u \notin V$ and the edges $vu, v \in V$ with (unit) weight $w_{uv} = 1$. Then, the girth of this new graph G' is exactly $2 + w_{\min}$ where $w_{\min}$ denotes the minimum edge-weight in G. Since any constant-factor approximation algorithm for computing $w_{\min}$ essentially requires time $\Omega(m)$, so does any constant-factor approximation algorithm for the girth of weighted graphs.

We study what happens if we have *sorted* adjacency lists[3].

THEOREM 1.3. *Let $G = (V, E, w)$ have sorted adjacency lists.*

1. *If all edge-weights are in $\{1, \ldots, M\}$ then, we can compute a deterministic 4-approximation for* GIRTH *in time $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}M)$.*
2. *If all edge-weights are non-negative then, we can compute a* randomized *4-approximation for* GIRTH *in time $\tilde{\mathcal{O}}(n^{5/3})$. For every $\varepsilon > 0$, we can also compute a deterministic $(4+\varepsilon)$-approximation for* GIRTH *in time $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}1/\varepsilon)$.*

We observe that even assuming sorted adjacency lists, it is not clear whether the algorithm of Theorem 1.1 can be implemented to run in time $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}M)$. Indeed, this algorithm requires to build several induced subgraphs in time roughly proportional to their size, that requires a different preprocessing on the adjacency lists. We prove that we do not need to construct these induced subgraphs *entirely* in order to derive a constant-factor approximation of the girth. Similarly, for graphs with non-negative edge-weights we cannot use our polynomial-factor approximation algorithm for the girth directly, as it needs to enumerate all edges in the graph. We overcome this difficulty through the help of a classical density result for the C_4-free *unweighted* graphs [4].

1.2. Related work.

[3]Throughout this paper, we call an adjacency list *sorted* if it is sorted by edge-weight, and *ordered* if it is sorted by neighbour index. See also Sec. 2.

Approximation algorithms for the girth. Itai and Rodeh were the first to study the GIRTH problem for *unweighted* graphs [10]. Among other results, they showed how to compute an additive +1-approximation of the girth in time $\mathcal{O}(n^2)$. This was later completed by Lingas and Lundell [11], who proposed a randomized quasi 2-approximation algorithm for this problem that runs in time $\mathcal{O}(n^{3/2}\sqrt{\log n})$.

In [18], Roditty and Vassilevska Williams presented the first deterministic approximation algorithm for the girth of unweighted graphs. Specifically, they obtained a 2-approximation algorithm in time $\tilde{\mathcal{O}}(n^{5/3})$, and they conjectured that there does not exist any subquadratic $(2-\varepsilon)$-approximation for GIRTH. We obtain the same approximation factor for weighted graphs, and we almost match their running time up to polylog factors and to an additional term in $\tilde{O}(m)$. It would be interesting to know whether in our case (if the adjacency lists are sorted), this dependency on m can be removed while preserving the approximation factor 2.

Recently, new subquadratic-time approximation algorithms were proposed for GIRTH in unweighted graphs (see [8]), showing various trade-offs between the running time and the best achievable approximation factor. It is open whether one can achieve a constant-factor approximation for the girth in, say, $\tilde{\mathcal{O}}(n^{1+o(1)})$-time.

Much less is known about the girth of weighted graphs. The first known subcubic approximation was the one of Lingas and Lundell [11], that only applies to graphs with bounded integer edge-weights. Their work somewhat generalizes the algorithm of Itai and Rodeh for unweighted graphs. The approximation factor was later improved to 4/3 by Roditty and Tov, still for the graphs with bounded integer weights, and to $4/3+\varepsilon$ for the graphs with *arbitrary weights* [16]. Our algorithms in this paper are faster than these two previous algorithms, but they use the latter as a routine to be applied on several subgraphs of sublinear size. Therefore, the approximation factors that we obtain cannot outperform those obtained in [11, 16].

Since the preliminary version of this work was published, a randomized constant-factor approximation algorithm for the girth of *directed* weighted graphs was obtained in [7] (see also [12] for a previous logarithmic approximation). For undirected graphs, we obtain a better approximation ratio than in [7]. Furthermore, our algorithms are deterministic.

Approximate distances. Finally, approximation algorithms for the girth are tightly related to the computation of approximate distances in weighted graphs. In a seminal paper [19], Thorup and Zwick showed that we can compute in expected time $\mathcal{O}(mn^{1/k})$ an approximate distance oracle: that can answer any distance query in time $\mathcal{O}(k)$ with a multiplicative stretch at most $2k-1$. This has been improved in several follow-ups [2, 5, 13, 15, 21].

However, the construction of most oracles already takes (super)quadratic time for moderately dense graphs. A key observation is that we do not need to construct these oracles *entirely* if we just want to approximate the girth. This allows us to avoid a great deal of distance computations, and so, to lower the running time.

1.3. Organization of the paper. We start gathering in Section 2 some known results from the literature that we will use for our algorithm.

Then, in Section 3, we give some new insights on the algorithm of Lingas and Lundell [11] before presenting our main result for graphs with bounded integer edge-weights. Our algorithm is then generalized to graphs with arbitrary weights. Finally, we remove the dependency on the number of edges in the time complexity of our

algorithms (assuming sorted adjacency lists, and at the price of a larger approximation factor than 2).

We conclude this paper with some open perspectives (Section 6).

Results of this paper were partially presented at the ICALP'19 conference [9].

2. Preliminaries. Graphs in this study are finite, simple (hence, without any loop nor multiple edges), connected and edge-weighted. Specifically, we denote a weighted graph by a triple $G = (V, E, w)$ where $w : E \to \mathbb{R}^+$ is the edge-weight function of G. The weight of a subgraph $H \subseteq G$, denoted $w(H) := \sum_{e \in E(H)} w_e$, is the sum of the weights on its edges. The girth of G is the minimum weight of a cycle in G. The distance $dist_G(u, v)$ between any two vertices $u, v \in V$ is the minimum weight of an uv-path in G. By extension, for every $v \in V$ and $S \subseteq V$ we define $dist_G(v, S) := \min_{u \in S} dist_G(u, v)$. – We will sometimes omit the subscript if no ambiguity on the graph G can occur. – For any $v \in V$ and $r \geq 0$, we also define the ball $B_G(v, r) := \{u \in V \mid dist_G(u, v) \leq r\}$. Finally, an r-nearest set for v is any r-set $\mathcal{N}_r(v)$ such that, for any $x \in \mathcal{N}_r(v)$ and $y \notin \mathcal{N}_r(v)$, we have $dist_G(v, x) \leq dist_G(v, y)$.

We refer to [3] for any undefined graph terminology. For every $v \in V$, let $N_G(v) = \{u \in V \mid uv \in E\}$ be the (open) neighbourhood of vertex v and let $d_v = |N_G(v)|$ be its degree. Let $Q_v = \{vu \mid u \in N_G(v)\}$ be totally ordered. We call it a *sorted* adjacency list if edges incident to v are ordered by increasing weight, *i.e.*, $Q_v = (vu_1, vu_2, \ldots, vu_{d_v})$ and $w_{vu_i} \leq w_{vu_{i+1}}$ for every $i < d_v$. However, we call it an *ordered* adjacency list if, given some fixed total ordering $\prec$ over V the neighbours of v are ordered according to $\prec$ (*i.e.*, $u_i \prec u_{i+1}$ for every $i < d_v$). Throughout the rest of the paper we will assume that each vertex has access to two copies of its adjacency list: one being sorted and the other being ordered. The former can always be ensured up to an $\tilde{O}(m)$-time preprocessing.

2.1. The Hitting Set method. We gather many well-known facts in the literature, that can be found, *e.g.*, in [15, 19, 1, 6]. All these facts are combined in order to prove the following useful result for our algorithms:

PROPOSITION 2.1. *For any graph $G = (V, E, w)$ with* sorted *adjacency lists, in $\tilde{O}(n^{5/3})$-time we can compute a set $S \subseteq V$, and the open balls $B_S(v) := \{u \in V \mid dist(v, u) < dist(v, S)\}$ for every $v \in V$, such that the following two properties hold true:*

1. *$|S| = \tilde{\mathcal{O}}(n^{2/3})$;*
2. *and for every $v \in V$ we have $|B_S(v)| = \mathcal{O}(n^{1/3})$.*

It is well-known that a set S as requested by Proposition 2.1 can be constructed *randomly* as follows: every vertex in V is added in S with equal probability $n^{-1/3}$ [19]. This construction was derandomized in [15, 19, 1, 6].

We will use the following two lemmata:

LEMMA 2.2 ([6]). *For every $G = (V, E, w), v \in V$ and $r \geq 0$, we can compute an r-nearest set for v in time $\mathcal{O}(r^2)$ (assuming* sorted *adjacency lists). Furthermore, all the vertices in this set can be labeled with their distance to v within the same amount of time.*

LEMMA 2.3 ([1]). *Given n sets of size r over a universe of size s, a set S of size $\mathcal{O}(s/r \log n)$ hitting all n sets in at least one element can be found deterministically in time $\mathcal{O}(s + nr)$.*

Proof of Proposition 2.1. Set $r = \lfloor n^{1/3} \rfloor$. We compute an r-nearest set $\mathcal{N}_r(v)$ for every vertex $v \in V$, that can be done in total time $\mathcal{O}(n \cdot n^{2/3}) = \mathcal{O}(n^{5/3})$ by Lemma 2.2. By

Lemma 2.3 (applied for $s = n$) we can compute in time $\mathcal{O}(n+n\cdot n^{1/3}) = \mathcal{O}(n^{4/3})$ a set S of size $\tilde{O}(n^{2/3})$ that intersects all the nearest sets $\mathcal{N}_r(v)$. Since, for every $v \in V$, we have $B_S(v) \subseteq \mathcal{N}_r(v)$ by construction, we can compute this ball in time $\mathcal{O}(r)$ simply by scanning the r-nearest set, and so, we can compute all the balls $B_S(v), v \in V$ in total time $\mathcal{O}(nr) = \mathcal{O}(n^{4/3})$.

In what follows we will not only need the balls $B_S(v)$ for every vertex v, but also the subgraphs these balls induce in G. Next, we prove that all these subgraphs can be obtained almost for free.

LEMMA 2.4 (folklore). *For every $G = (V, E, w)$ and $U \subseteq V$ we can compute the subgraph $G[U]$ induced by U in time $\tilde{\mathcal{O}}(|U|^2)$ (assuming* ordered *adjacency lists).*

Proof. For every $x, y \in U$ we search whether yx is present in the adjacency list of y. Since the adjacency lists are ordered, this can be done in time $\mathcal{O}(\log n)$ by using a dichotomic search. □

3. Case of graphs with bounded integer weights. This section is devoted to the proof of Theorem 1.1. We start presenting some new properties of a previous approximation algorithm for the girth of weighted graphs (Section 3.1) as we will need to use them in our own algorithm. Then, we prove our main result for graphs with bounded integer weights in Section 3.2.

3.1. Reporting a close short cycle. We propose a deeper analysis of an existing approximation algorithm for GIRTH on weighted graphs [11]. Roughly, this algorithm applies a same procedure to every vertex of the graph. In order to derive the approximation factor of their algorithm, the authors in [11] were considering a run that takes as entry some vertex *on a shortest cycle*. This is in contrast with the classical algorithm from Itai and Rodeh on unweighted graphs [10], that also offers provable guarantees on the length of the output assuming there is a shortest cycle passing *close* to the source (but not necessarily passing by this vertex); see [11, Lemma 2]. We revisit the analysis of the algorithm in [11] for weighted graphs, and we prove that this algorithm also satisfies such a "closeness property".

The `HBD`*-algorithm from [11].* Given $G = (V, E, w)$, $s \in V$ and $t \geq 0$, the algorithm $HBD(G, s, t)$ is a relaxed version of Dijkstra's single-source shortest-path algorithm. We are only interested in computing the ball of radius t around s, and so, we stop if there are no more unvisited vertices at a distance $\leq t$ from s. Furthermore, whenever we visit a vertex $u \in B_G(s, t)$, we only relax edges $e = \{u, v\}$ such that $dist(s, u) + w_e \leq t$. Then, a cycle is detected if we already inferred that $dist(s, v) \leq t$ (*i.e.*, using another neighbour of v than u). Overall, the algorithm stops as soon as it encounters a cycle, or all the vertices in $B_G(s, t)$ were visited. Assuming sorted adjacency lists, this algorithm runs in $\tilde{\mathcal{O}}(n)$-time [11]. Here, an easy but important observation to be made is that, if $HBD(G, s, t)$ detects a cycle, then this is also the case for $HBD(G, s, t')$, for every $t' \geq t$. We will use this observation implicitly throughout the rest of the paper.

LEMMA 3.1 ([11]). *If `HBD`(G, s, t) detects a cycle, then its weight is $\leq 2t$.*

We now complete the analysis of the `HBD`-algorithm in order to derive a generalization of [11, Lemma 2] to weighted graphs. Assuming no cycle has been detected, we first gain more insights on the structure of the ball of radius t centered at s.

LEMMA 3.2. *If `HBD`(G, s, t) does not detect a cycle then, for any $v \in B_G(s, t)$, there exists a unique sv-path of weight $\leq t$.*

Proof. Since we did not detect a cycle, we relaxed the edges of all the paths of weight $\leq t$ between s and the other vertices of G. The result now follows, since the

(a) `HBD`(G, s, t)

for all $v \in V$ **do**
  $d(v) \leftarrow \infty$
  $\pi(v) \leftarrow NIL$
$d(s) \leftarrow 0$; $Q \leftarrow \{s\}$
while $Q \neq \emptyset$ **do**
  $u \leftarrow$ `Extract-min`(Q)
  `Controlled-Relax`(u, t)

(b) `Controlled-Relax`(u, t)

$Q_u \leftarrow$ sorted adj. list
$uv \leftarrow$ `Extract-min`(Q_u)
while $d(u) + w_{uv} \leq t$ **do**
  `RelaxOrStop`(u, v)
  $uv \leftarrow$ `Extract-min`(Q_u)

(c) `RelaxOrStop`(u, v)

if $d(v) \neq \infty$ **then**
  return a cycle and stop
else
  $d(v) \leftarrow d(u) + w_{uv}$
  $Q \leftarrow Q \cup \{v\}$

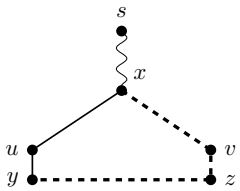
edges that we relaxed during the algorithm induce a spanning tree of $B_G(s, t)$. □

Based on Lemma 3.2, we state some bounds on the weight of the cycle detected using `HBD`. In particular, Corollary 3.4 will play a key role in the analysis of our algorithms.

COROLLARY 3.3. *Given $G = (V, E, w)$, let $s \in V$ and let C be a cycle. The minimum t_0 such that `HBD`(G, s, t_0) detects a cycle satisfies $t_0 \leq dist(s, C) + w(C)$.*

Proof. Fix $x, y \in V(C)$ and partition the cycle C into the two xy-paths P_1, P_2. By the contrapositive of Lemma 3.2, a cycle is detected if $t \geq \min\{dist(s, x), dist(s, y)\} + \max\{w(P_1), w(P_2)\}$. Indeed, in this situation there exist two different paths of length $\leq t$ between s and one of x or y. In particular, set $dist(s, x) = dist(s, C)$ and let $y \in V(C) \setminus \{x\}$ be arbitrary. We have $t_0 \leq dist(s, C) + \max\{w(P_1), w(P_2)\} \leq dist(s, C) + w(C)$. □

COROLLARY 3.4. *Given $G = (V, E, w)$, let $s \in V$ and let C be a cycle. Assume the existence of a vertex $x \in V(C)$ such that $\max_{v \in V(C)} dist_C(x, v) \geq dist_G(s, x) > 0$. Then, the minimum t_0 such that `HBD`(G, s, t_0) detects a cycle satisfies $t_0 \leq w(C)$.*



Proof. Let $B_x = \{v \in V(C) \mid dist_C(x, v) < dist_G(x, s)\}$. Since we assume that we have $\max_{v \in V(C)} dist_C(x, v) \geq dist_G(x, s)$, $B_x \neq V(C)$. Hence there exist $uy, vz \in E(C)$ distinct such that $u, v \in B_x$ but $y, z \notin B_x$. W.l.o.g. $dist_C(x, y) \leq dist_C(x, z)$. We can bipartition $E(C)$ in two edge-disjoint xy-paths P_1 and P_2, with P_1 being the xy-subpath passing by vz (and so, P_2 is the other xy-subpath passing by uy). Note that it implies $dist_C(x, y) = w(P_2) \leq w(C)/2$. Then, by Lemma 3.2 we have $t_0 \leq dist_G(s, x) + \max\{w(P_1), w(P_2)\} = dist_G(s, x) + w(P_1) \leq dist_C(x, y) + w(P_1) = w(P_2) + w(P_1) = w(C)$. □

3.2. Subquadratic-time approximation. We are now ready to present the main result in this section:

THEOREM 1.1. *For every $G = (V, E, w)$ with edge-weights in $\{1, \ldots, M\}$, we can compute a deterministic 2-approximation for* GIRTH *in time $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}M + m)$.*

Proof. We analyse the following `Subquadratic-Approx` algorithm:

The algorithm starts precomputing a set $S \subseteq V$ and the open balls $(B_S(v))_{v \in V}$ as described in Proposition 2.1. This takes time $\tilde{\mathcal{O}}(n^{5/3})$, plus an additional preprocessing time in $\tilde{O}(m)$ for sorting the adjacency lists. Then, we process the vertices in S and those in $V \setminus S$ separately: For every $s \in S$, we compute the smallest $t_s \in [3; Mn]$ such that `HBD`(G, s, t_s) detects a cycle by using a dichotomic search (procedure

(a) `Approx-Girth`(G, s, M)

1: Find the minimum $t \in [3; M \cdot |V(G)|]$ such that: $\texttt{HBD}(G, s, t)$ detects a cycle.
2: Let C_s be the shortest cycle we so computed.
3: **return** C_s.

(b) `Subquadratic-Approx`(G, M)

1: Let S and $(B_S(v))_{v \in V}$ be as in Prop. 2.1.
2: **for all** $s \in S$ **do**
3: $\quad C_s \leftarrow \texttt{Approx-Girth}(G, s, M)$
4:
5: **for all** $v \notin S$ **do**
6: $\quad$ Let G'_v be induced by $B_S(v)$.
7: $\quad C_v \leftarrow \texttt{Approx-Girth}(G'_v, v, M)$
8:
9: **return** a shortest cycle in $\{C_v \mid v \in V\}$.

$\texttt{Approx-Girth}(G, s, M)$). We store the cycle C_s outputted by $\texttt{HBD}(G, s, t_s)$. Since each test we perform during the dichotomic search consists in a call to the HBD-algorithm, this takes time $\tilde{\mathcal{O}}(n \log M)$ per vertex in S, and so, $\tilde{\mathcal{O}}(n|S| \log M) = \tilde{\mathcal{O}}(n^{5/3} \log M)$ in total. We now consider the vertices $v \in V \setminus S$ sequentially. Let G'_v be the subgraph of G induced by the open ball $B_S(v)$. By Lemma 2.4, this subgraph can be computed in time $\tilde{\mathcal{O}}(|B_S(v)|^2) = \tilde{\mathcal{O}}(n^{2/3})$ – assuming a preprocessing of the graph in time $\mathcal{O}(m)$ for ordering the adjacency lists. We apply the same procedure as for the vertices in S but, we restrict ourselves to the ball $B_S(v)$. That is, we call $\texttt{Approx-Girth}(G'_v, v, M)$, and we denote by C_v the cycle outputted by this algorithm (in the case G'_v is not acyclic). Since we restrict ourselves to a subgraph of order $\mathcal{O}(n^{1/3})$, this algorithm takes total time $\tilde{\mathcal{O}}(n \cdot (n^{2/3} + n^{1/3} \log M)) = \tilde{\mathcal{O}}(n^{5/3} \log M)$.

Let $C \in \{C_v \mid v \in V\}$ be of minimum weight. We claim that $w(C)$ is a 2-approximation of the girth of G, that will end proving the theorem. In order to prove this claim, we apply the following case analysis to some arbitrary shortest cycle C_0 of G. If $V(C_0) \cap S \neq \emptyset$ then, let C_S be a shortest cycle among $\{C_s \mid s \in S\}$. We prove as a subclaim that $w(C_S)$ is at most twice the weight of a shortest cycle intersecting S. In order to prove this subclaim, it suffices to prove that for every $s \in S$ in a cycle, we compute a cycle C_s of weight no more than twice the weight of a shortest cycle passing by s. By Corollary 3.3, if t_s is the smallest t such that $HBD(G, s, t)$ detects a cycle then, a shortest cycle C passing by s must have weight $\geq t_s$. Furthermore, by Lemma 3.1 we get $w(C_s) \leq 2t_s$, thereby proving the subclaim. Thus, $w(C_S) \leq 2w(C_0)$ if $V(C_0) \cap S \neq \emptyset$. From now on we assume $V(C_0) \cap S = \emptyset$. Let $v \in V(C_0)$ be arbitrary. There are two subcases:

- If $V(C_0) \subseteq B_S(v)$ then, C_0 is also a cycle in G'_v. Moreover by Corollary 3.3 applied for $dist(v, C_0) = 0$, the smallest t_v such that $\texttt{HBD}(G'_v, v, t_v)$ detects a cycle satisfies $t_v \leq w(C_0)$. By Lemma 3.1, $w(C) \leq w(C_v) \leq 2w(C_0)$.
- Otherwise $V(C_0) \not\subseteq B_S(v)$. This implies that we have:

$$\max_{u \in V(C_0)} dist_{C_0}(u, v) \geq \max_{u \in V(C_0)} dist_G(u, v) \geq dist_G(v, S) > 0.$$

  Furthermore, let $s \in S$ minimize $dist_G(s, v)$. Then, by Corollary 3.4, the smallest t_s such that $\texttt{HBD}(G, s, t_s)$ detects a cycle satisfies $t_s \leq w(C_0)$. As a result, by Lemma 3.1:

$$w(C) \leq w(C_s) \leq 2w(C_0).$$

Summarizing, $w(C) \leq 2w(C_0)$ in all the cases. □

4. Generalization to unbounded weights. This section is devoted to the proof of Theorem 1.2. We divide it into two parts. In Section 4.1 we present a *polynomial-factor* approximation of the girth in subquadratic time. This part is new compared to [16] and the techniques used are interesting in their own right. Then, based on a clever technique from [16], we end up refining this rough estimate of the girth until we obtain a constant-factor approximation (Section 4.2).

Throughout this section, we will use the main result of Roditty and Tov as a subroutine:

THEOREM 4.1 ([16]). *For every $G = (V, E, w)$ with arbitrary non-negative edge-weights, we can compute a $(4/3+\varepsilon)$-approximation for* GIRTH *in time $\tilde{\mathcal{O}}(n^2/\varepsilon)$.*

4.1. A polynomial-factor approximation. For simplicity, we first reduce the general case of graphs with non-negative weights to the subcase of graphs with positive weights. The proof is quite similar, but simpler, to the one of Proposition 4.3.

LEMMA 4.2. *Assume there exists an $T(n,m)$-time α-approximation algorithm for* GIRTH *for graphs with positive edge-weights, where $T(n,m) = \Omega(m)$. Then, there also exists an $\mathcal{O}(T(n,m))$-time α-approximation algorithm for* GIRTH *for graphs with non-negative edge-weights.*

Proof. Let $G = (V, E, w)$ with non-negative real edge-weights, and let $E_0 = \{e \in E \mid w_e = 0\}$. If the (unweighted) graph $G_0 := (V, E_0)$ contains a cycle C, then $w(C) = 0$ is minimized and we can output C. Otherwise, G_0 is a forest, and let $V_1, V_2, \ldots, V_p$ be its connected components. We consider the following three cases:

1. First we scan all the edges in $E \setminus E_0$ in order to find, for any component V_i, the minimum weight of an edge with its two ends in V_i (if any). It takes time $\mathcal{O}(m)$. Furthermore, note that given $e \in E \setminus E_0$ with its two ends in V_i, there is a cycle of weight exactly w_e. Let C_0 be the minimum-weight cycle we so computed during this step.
2. In the same way, we can easily find a cycle C_1 of minimum weight that only intersects two components: for that, we scan all the edges in $E \setminus E_0$ in order to find, for any two distinct V_i, V_j, the at most two edges of minimum-weight with one end in V_i and the other end in V_j. It takes time $\mathcal{O}(m)$.
3. We are now left with approximating the short cycles that intersect at least three components of G_0. For that, let G' be obtained from G by contracting each component V_j into one vertex; for every distinct V_i, V_j, if there exists an edge $e \in E \cap (V_i \times V_j)$ then, we choose e minimizing w_e and we set $w'_{v_i v_j} = w_e$ in G'. We observe that if there exists a shortest cycle C of G that intersects at least three components V_j then, $C \cap V_i$ is either empty or induces a path in $G_0[V_i]$ for every i (otherwise, there would be two vertices in V_i that are connected in C by two subpaths of positive weight, and so, we could obtain a cycle of smaller weight than C by replacing any of these subpaths with any path in $G_0[V_i]$). In particular, it corresponds to C a cycle C' in G' such that $w(C') \leq w(C)$. Conversely, to any cycle C' in G, it corresponds a cycle C in G such that $w(C) \leq w(C')$ (obtained by uncontracting the connected components of G_0). Let C'_2 be a cycle of weight at most α times the girth of G', and let C_2 be a corresponding cycle in G.

Then, by outputting a cycle among C_0, C_1, C_2 that is of minimum weight, one obtains an α-approximation for the girth of G. □

We now obtain an approximation of the girth that only depends on the order of the graph. We stress that for weighted graphs, this is already a non-trivial task.

PROPOSITION 4.3. *For every $G = (V, E, w)$ with arbitrary positive edge-weights,*

we can compute an $\tilde{\mathcal{O}}(n^{2/3})$-approximation for Girth *in time $\tilde{\mathcal{O}}(n^{5/3}+m)$.*

Proof. Let S be as in Proposition 2.1. We show a significantly more elaborate method that uses S in order to approximate the girth. We divide this method into five main steps.

Step 1: check the small balls. For every $v \notin S$, let G'_v be the subgraph induced by the open ball $B_S(v)$. As before, we first estimate the girth of G'_v. Since this subgraph has order $\mathcal{O}(n^{1/3})$, by Theorem 4.1 we can compute a constant-factor approximation for its girth in time $\tilde{\mathcal{O}}(n^{2/3})$ (say, a 2-approximation). Overall, this step takes total time $\tilde{\mathcal{O}}(n^{5/3})$. Furthermore, after completing this step the following property (also used in Theorem 1.1) becomes true:

Claim 1. *Let C_v be a shortest cycle passing through v. If $w(C_v) < 2 \cdot dist(v,S)$ then, we computed a cycle of weight $\leq 2w(C_v)$.*

Proof. This is trivial if $v \in S$. Otherwise, $V(C_v) \subseteq B_G(v, w(C_v)/2) \subseteq B_S(v) = V(G'_v)$, and so, we outputted a cycle of weight $\leq 2w(C_v)$ for the subgraph G'_v. ◇

Step 2: partitioning into (shortest path) subtrees. Intuitively, what we try to do next is to approximate the weight of a shortest cycle passing close to S. The difference with Theorem 1.1 is that we cannot use directly the algorithm of Roditty and Tov for that. Indeed, their algorithm has some global steps (*e.g.*, the approximate computation of the girth of some sparse spanner) that we currently do not know how to do in subquadratic time. So, we need to find some new techniques. Specifically, we partition the vertex-set V into shortest-path subtrees $(T_s)_{s\in S}$ such that, for every $s \in S$ and $v \in V(T_s)$ we have $dist(v,s) = dist(v,S)$. As noted, *e.g.*, in [19], a simple way to do that is to add a dummy vertex $x_S \notin V$, edges sx_S for every $s \in S$ with weight 0, then to compute a shortest-path tree rooted at x_S in time $\tilde{\mathcal{O}}(m)$. See Fig. 3 for an example. In what follows, we show how to use this tree structure in order to compute short cycles.

Fig. 3: An example of Step 2. The two vertices in S are drawn as rectangles.

Step 3: finding short cycles in a subtree. Let $s \in S$ be fixed. Informally, we try to estimate the weight of a shortest cycle in $V(T_s)$. Note that every such a cycle has an edge that is not contained in T_s. So, we consider all the edges $e = uv$ such that $u, v \in V(T_s)$ but $e \notin E(T_s)$. Adding this edge in T_s closes a cycle. Let $C_{e,s}$ be an (unknown) shortest cycle passing by e and contained in $V(T_s)$. We output $dist(s,u) + w_e + dist(v,s)$ as a rough estimate of $w(C_{e,s})$. Indeed, the latter is a straightforward upper-bound on $w(C_{e,s})$, and this bound is reached if $s \in \{u,v\}$. Overall, this step takes total time $\mathcal{O}(m)$.

Claim 2. *Let C^*_s be a shortest cycle contained in $V(T_s)$. After Steps 1-3, we computed a cycle of weight $\leq 2w(C^*_s)$.*

Proof. Let $e = uv \in E(C^*_s) \setminus E(T_s)$. We also define C_u, C_v shortest cycles passing by u and v, respectively. By Claim 1, either we computed at Step 1 a cycle of

weight $\leq 2w(C_u) \leq 2w(C_s^*)$ (a cycle of weight $\leq 2w(C_v)) \leq 2w(C_s^*)$, resp.), or we know for sure that $w(C_s^*) \geq w(C_u) \geq 2dist(u,S) = 2dist(u,s)$ ($w(C_s^*) \geq w(C_v) \geq 2dist(v,S) = 2dist(v,s)$, resp.). In the latter case, $dist(s,u) + w_e + dist(v,s) \leq w(C_s^*)/2 + w_e + w(C_s^*)/2 \leq 2w(C_s^*)$ is a 2-approximation of $w(C_s^*)$. ⋄

Step 4: finding short cycles in two subtrees. We now want to estimate the weight of a shortest cycle in $V(T_s) \cup V(T_{s'})$, for some distinct $s, s' \in S$. We only need to consider the case where this cycle must contain two edges e, e' with an end in $V(T_s)$ and the other end in $V(T_{s'})$ (all other cases have already been considered at Step 3).

1. We scan all the edges $e = uv \in E$ such that u and v are not in a same subtree. Let $s_u, s_v \in S$ such that $u \in V(T_{s_u}), v \in V(T_{s_v})$. We set $\ell(e) = dist(s_u, u) + w_e + dist(v, s_v)$.
2. Group all these above edges with their two ends in the same two subtrees. It takes time $\mathcal{O}(m + |S|) = \mathcal{O}(m + n^{2/3})$ by using, say, a linear-time sorting algorithm.
3. Finally, for every distinct $s, s' \in S$, let $E(s, s')$ contain all the edges with one end in T_s and the other end in $T_{s'}$. If $|E(s, s')| \geq 2$ then, we pick e, e' minimizing $\ell(\cdot)$ and we output $\ell(e) + \ell(e')$. Overall, since the sets $E(s, s')$ partition the edges of G, this last phase also takes time $\mathcal{O}(m)$.

Claim 3. *Let $s, s' \in S$ be distinct and let $C^*_{s,s'}$ be a shortest cycle contained in $V(T_s) \cup V(T_{s'})$. After Steps 1-4, we computed a cycle of weight $\leq 3w(C^*_{s,s'})$.*

Proof. If either $V(C^*_{s,s'}) \subseteq V(T_s)$ or $V(C^*_{s,s'}) \subseteq V(T_{s'})$ then, we are done by Claim 2. Otherwise, let $e = uv', e' = vu' \in E(s, s') \cap E(C^*_{s,s'})$ such that $u, v \in V(T_s)$ and $u', v' \in V(T_{s'})$. Choosing the uv-path in T_s and the $u'v'$-path in $T_{s'}$, one obtains that $w(C^*_{s,s'}) \leq \ell(e) + \ell(e')$. Furthermore, let C_u be a shortest cycle passing by u. By Claim 1, either we computed at Step 1 a cycle of weight $\leq 2w(C_u) \leq 2w(C^*_{s,s'})$, or we know for sure that $w(C^*_{s,s'}) \geq w(C_u) \geq 2dist(u,S) = 2dist(u,s)$. We obtain similar results for v, u', v'. As a result, and unless we found a better estimate of the girth during Step 1, we have $\ell(e) + \ell(e') \leq 2w(C^*_{s,s'}) + w_e + w_{e'} \leq 3w(C^*_{s,s'})$. ⋄

Step 5: the general case. We end up defining a weighted graph $H_S = (S, E_S, w^S)$, where $E_S = \{ss' \mid E(s, s') \neq \emptyset\}$, and for every $ss' \in E_S$:

$$w^S_{ss'} = \min_{e \in E(s,s')} \ell(e) = \min_{uv \in E(s,s')} dist(s,u) + w_{uv} + dist(v,s').$$

Roughly, $w^S_{ss'}$ is the smallest weight of an ss'-path with one edge in $E(s, s')$. We can construct H_S simply by scanning all the sets $E(s, s')$ (computed during Step 4). Overall, since the sets $E(s, s')$ partition the edges of G, this takes total time $\mathcal{O}(m+|S|) = \mathcal{O}(m+n^{2/3})$. Furthermore, by Theorem 4.1 we can compute a constant-factor approximation of the girth of H_S in time $\tilde{\mathcal{O}}(|S|^2) = \tilde{\mathcal{O}}(n^{4/3})$. The graph H_S is *not* a subgraph of G. However, given a cycle C_H for H_S, we can compute a cycle C^*_H of G as follows. For every $s \in V(C_H)$ let $s', s'' \in V(C_H)$ be its two neighbours. By construction, there exist $e = uv \in E(s', s)$ and $e' = xy \in E(s, s'')$ such that the edges ss' and ss'' in H_S have weights $dist(s', u) + w_e + dist(v, s)$ and $dist(s, x) + w_{e'} + dist(y, s'')$, respectively. – We may assume the edges e, e' to be stored in H_S so that s', s'' will choose the same common edge with s. – Then, we replace s by the vx-path in T_s. It is important to notice that, by construction, we have $w(C^*_H) \leq w(C_H)$. In particular, we can apply this above transformation to the

(approximately shortest) cycle of H_S that has been outputted by the algorithm of Roditty and Tov (Theorem 4.1).

Overall, let $C_{\min}$ be a shortest cycle computed by the algorithm above (*i.e.*, after Steps 1-5). In order to finish the proof, we need to show that $w(C_{\min})$ is an $\tilde{\mathcal{O}}(n^{2/3})$-approximation of the girth of G. By Claims 2 and 3, this is the case if there exists a shortest cycle intersecting at most two subtrees $T_s, s \in S$. From now on assume that any shortest cycle C_0 of G intersects at least three subtrees T_s. We refer to Fig. 4 for an illustration of our proof. Write $C_0 = (v_0, v_1, \ldots, v_{p-1}, v_0)$ and assume w.l.o.g. v_0, v_{p-1} are not contained into the same subtree T_s. We partition the v_i's into the maximal subpaths $P_0, P_1, \ldots, P_{q-1},\ q \leq p$ that are contained into the vertex-set of a same subtree T_s (in particular, $v_0 \in V(P_0)$ and $v_{p-1} \in V(P_{q-1})$). Furthermore for every $j \in \{0, 1, \ldots, q-1\}$ let $s_j \in S$ be such $V(P_j) \subseteq V(T_{s_j})$, and let i_j be the largest index such that $v_{i_j} \in V(P_j)$. For instance, $i_{q-1} = p-1$ by construction. Since $P_0 = P_q$ and $q \geq 3$ by the hypothesis, there exist distinct indices j_1, j_2 such that $s_{j_1} = s_{j_2+1}$ and for every $j \in \{j_1, j_1+1, \ldots, j_2\}$ the s_j's are pairwise different (indices are taken modulo q).

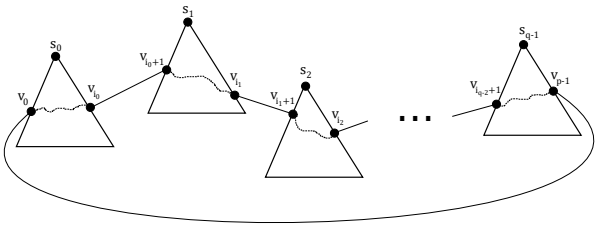


Fig. 4: Step 5 of Proposition 4.3.

Then, two cases may arise:

- Case $j_2 = j_1 + 1$. We have: $e_{j_1} := v_{i_{j_1}} v_{i_{j_1}+1},\ e_{j_2} := v_{i_{j_2}+1} v_{i_{j_2}} \in E(s_{j_1}, s_{j_2})$. Furthermore, C_0 goes by $v_{i_{j_1}}$ (by $v_{i_{j_1}+1}, v_{i_{j_2}+1}, v_{i_{j_2}}$, respectively), and so, by Claim 1, either we computed a short cycle of weight $\leq 2w(C_0)$ during Step 1, or we have:

  $$w(C_0) \geq 2 \cdot \max\{dist(s_{j_1}, v_{i_{j_1}}), dist(s_{j_1}, v_{i_{j_2}+1}), dist(s_{j_2}, v_{i_{j_1}+1}), dist(s_{j_2}, v_{i_{j_2}})\}.$$

  In the latter case, there exists a cycle of weight: $\leq dist(s_{j_1}, v_{i_{j_1}}) + w_{e_{j_1}} + dist(s_{j_2}, v_{i_{j_1}+1}) + dist(s_{j_2}, v_{i_{j_2}}) + w_{e_{j_2}} + dist(s_{j_1}, v_{i_{j_2}+1}) \leq 3w(C_0)$ that is fully contained in $V(T_{s_{j_1}}) \cup V(T_{s_{j_2}})$. By Claim 3, we so computed a cycle of weight $\leq 9w(C_0)$ at Step 4.
- From now on let us assume $j_2 \neq j_1 + 1$. For every j we have $e_j := v_{i_j} v_{i_j+1} \in E(s_j, s_{j+1})$, and so, the edge $s_j s_{j+1} \in E_S$ has weight no more than $dist(s_j, v_{i_j}) + w_{e_j} + dist(v_{i_j+1}, s_{j+1})$ in H_S (indices are taken modulo q for the s_j's and modulo p for the v_i's). Furthermore, C_0 goes by v_{i_j} (by v_{i_j+1}, respectively), and so, by Claim 1, either we computed a short cycle of weight $\leq 2w(C_0)$ during Step 1, or we have $w(C_0) \geq 2 \cdot \max\{dist(s_j, v_{i_j}), dist(s_{j+1}, v_{i_j+1}\}$ for every j.

In the latter case, $(s_{j_1}, s_{j_1+1}, \ldots, s_{j_2}, s_{j_2+1} = s_{j_1})$ is a cycle in H_S of weight:

$$\begin{aligned}
&\leq w(C_0) + \sum_{j=j_1}^{j_2} (dist(v_{i_{j-1}+1}, s_j) + dist(s_j, v_{i_j})) \\
&\leq w(C_0)(1 + (j_2 - j_1 + 1)) \\
&\leq w(C_0)|S| = \tilde{\mathcal{O}}(n^{2/3} \cdot w(C_0)).
\end{aligned}$$

Then, let C_H be a cycle of H_S such that $w(C_H) = \tilde{\mathcal{O}}(n^{2/3} \cdot w(C_0))$ (obtained by applying the algorithm of Roditty and Tov to H_S). As explained above, we can derive from C_H a cycle C_H^* of G such that $w(C_H^*) \leq w(C_H) = \tilde{\mathcal{O}}(n^{2/3} \cdot w(C_0))$.

Summarizing, we obtain an $\tilde{\mathcal{O}}(n^{2/3})$-approximation of the girth by outputting a shortest cycle computed during Steps 1,3,4,5. □

4.2. Improving the approximation factor. Then, we combine Proposition 4.3 with the approach taken in Theorem 1.1:

THEOREM 1.2. *For every $\varepsilon > 0$ and $G = (V, E, w)$ with non-negative edge-weights, we can compute a deterministic $(2+\varepsilon)$-approximation for* GIRTH *in time $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}1/\varepsilon + m)$.*

Proof. We may assume that all weights are positive by Lemma 4.2. Let g^* be the $\tilde{\mathcal{O}}(n^{2/3})$-approximation that we computed by using Proposition 4.3. There exists some (known) constant c such that the girth of G is somewhere between $g^*/(cn^{2/3}\log n)$ and g^*. Let $i_{\min}, i_{\max}$ be the smallest nonnegative integers such that $g^*/(cn^{2/3}\log n) \leq (1+\varepsilon/2)^{i_{\min}}$ and in the same way $g^* \leq (1+\varepsilon/2)^{i_{\max}}$. We have:

$$i_{\max} - i_{\min} = \mathcal{O}\left(\log_{1+\varepsilon/2}\left(\frac{g^*}{g^*/(cn^{2/3}\log n)}\right)\right) = \mathcal{O}(\log n/\log(1+\varepsilon/2)) = \mathcal{O}(\log n/\varepsilon).$$

Let S be as in Proposition 2.1. For every $v \in V \setminus S$, we compute a 2-approximation of a shortest cycle in G'_v: the subgraph of G induced by the ball $B_S(v)$[4]. By Theorem 4.1, it can be done in time $\tilde{\mathcal{O}}(n^{2/3})$ for each v, and so, this takes total time $\tilde{\mathcal{O}}(n^{5/3})$. Let $T = \{(1+\varepsilon/2)^i \mid i_{\min} \leq i \leq i_{\max}\}$. For every $s \in S$, we compute the smallest $t \in T$ such that $HBD(G, s, t)$ detects a cycle (if any). It can be done in time $\tilde{\mathcal{O}}(|S| n \log|T|) = \tilde{\mathcal{O}}(n^{5/3}\log 1/\varepsilon)$ by using a dichotomic search. Finally, let $g_{\min}$ be the value computed by the above algorithm (with a corresponding cycle). In order to conclude we prove, with a similar case analysis as for Theorem 1.1, that the girth of G is at least $g_{\min}/(2+\varepsilon)$. Specifically, let us consider a shortest cycle C_0. There are several cases:

- Assume that C_0 passes by some $s \in S$. Let i_0 be the smallest index such that $w(C_0) \leq (1+\varepsilon/2)^{i_0}$. Since we have $g^*/(cn^{2/3}\log n) \leq w(C_0) \leq g^*$, $i_0 \in \{i_{\min}, i_{\min}+1, \ldots, i_{\max}\}$. Furthermore, by Corollary 3.3 applied for $dist(s, C_0) = 0$, $HBD(G, s, (1+\varepsilon/2)^{i_0})$ detects a cycle. By Lemma 3.1, we so deduce that $g_{\min} \leq 2(1+\varepsilon/2)^{i_0} \leq 2(1+\varepsilon/2)w(C_0) \leq (2+\varepsilon)w(C_0)$.
- Otherwise, let $v \in V \setminus S$ be contained into C_0. We may assume that $V(C_0) \not\subseteq B_S(v)$, since otherwise C_0 is a cycle in G'_v, and so, we can already conclude that $g_{\min} \leq 2w(C_0)$. In particular, we have:

$$\max_{u \in V(C_0)} dist_{C_0}(u, v) \geq \max_{u \in V(C_0)} dist_G(u, v) \geq dist_G(v, S) > 0.$$

[4] In fact, this is already done in the proof of Proposition 4.3, but we restate it here for completeness of the method.

Let $s \in S$ be such that $dist_G(v,s) = dist_G(v,S)$. By Corollary 3.4, this implies that the smallest t_s such that $HBD(G,s,t_s)$ detects a cycle satisfies $t_s \leq w(C_0)$. Therefore, we can choose as above the smallest index i_0 such that $w(C_0) \leq (1+\varepsilon/2)^{i_0}$. As already noticed $i_0 \in \{i_{\min}, i_{\min}+1, \ldots, i_{\max}\}$, and by Corollary 3.4 we know that $HBD(G,s,(1+\varepsilon/2)^{i_0})$ detects a cycle. In this situation, we can conclude by Lemma 3.1 that $g_{\min} \leq 2(1+\varepsilon/2)^{i_0} \leq 2(1+\varepsilon/2)w(C_0) \leq (2+\varepsilon)w(C_0)$.

Overall, the above proves as claimed $g_{\min} \leq (2+\varepsilon)w(C_0)$. □

5. A subquadratic algorithm for dense graphs. A drawback of the algorithms in Theorems 1.1 and 1.2 is that their time complexity also depends on the number m of edges. It implies that for dense graphs with $m = \Theta(n^2)$ edges we do not achieve any improvement on the running time compared to [11, 16]. The main result of this section is that assuming sorted adjacency lists, the dependency on m can always be discarded (Theorem 1.3).

5.1. A weaker result. We start with a sparsification lemma:

PROPOSITION 5.1. *For every $G=(V,E,w)$ with non-negative edge-weights and sorted adjacency lists, we can compute:*

1. *a randomized 4-approximation for* GIRTH *in expected time $\tilde{\mathcal{O}}(n^{5/3})$;*
2. *and, for every $\varepsilon > 0$, a deterministic $(8+\varepsilon)$-approximation for* GIRTH *in time $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}1/\varepsilon)$.*

Roughly, we can prove Theorem 1.3 by combining this above Proposition 5.1 with a natural modification of the algorithm presented in Section 3.2.

We will need the following well-known result in graph theory:

THEOREM 5.2 ([4]). *Every* unweighted *graph with order n and $m \geq \left(\frac{1}{2}+o(1)\right)n^{3/2}$ edges contains a cycle of length four.*

Proof of Proposition 5.1. If G has $m = \tilde{\mathcal{O}}(n^{5/3})$ edges then, we can simply apply Theorem 1.2 for $\varepsilon = 2$ (the latter can be easily verified by scanning the adjacency lists until we read the end of it or we reach the desired upper-bound). From now on assume this is not the case and let H be induced by the $\left\lceil \left(\frac{1}{2}+o(1)\right)n^{3/2} \right\rceil$ edges of minimum weight in G. We claim that H can be constructed in time $\tilde{\mathcal{O}}(n^{3/2})$ by using a priority queue Q. Indeed, initially we set $E(H) = \emptyset$ and for every $v \in V$ we start inserting in Q the edge of minimum-weight that is incident to v. This way, we ensure that a minimum-weight edge of $G \setminus E(H)$ is present in Q (recall that initially, $E(H) = \emptyset$, and so, $G = G \setminus E(H)$). Then, in order to preserve this above invariant, each time a minimum-weight edge uv is extracted from Q and added in H we insert in Q the remaining edge of minimum weight in Q_u and the one in Q_v (if any). – Note that in doing so, a same edge can be added in Q twice, but this has no consequence on the algorithm. – We now apply Theorem 1.2 for $\varepsilon' = \varepsilon/4$ to H, and we so obtain a cycle C that is a $(2+\varepsilon/4)$-approximation of the girth of H. We claim that $w(C)$ is also a $(8+\varepsilon)$-approximation of the girth of G. In order to prove this claim, we need to consider two different cases:

- Assume there exists a shortest cycle C_0 of G such that $E(C_0) \subseteq E(H)$. By Theorem 1.2, $w(C) \leq (2+\varepsilon/4)w(C_0) < (8+\varepsilon)w(C_0)$.
- Otherwise, any shortest cycle C_0 of G has at least one edge that is not contained in H. Since edges are added by increasing weights, this implies that every shortest cycle contains an edge of weight at least $w_{\max}$, where $w_{\max}$ denotes the maximum-weight of an edge in H. In particular, the girth of G is at least $w_{\max}$. Furthermore, since H has enough edges by construction, by

Theorem 5.2 it contains a cycle of four vertices; the latter has weight at most $4w_{\max}$. As a result, $w(C) \leq (2+\varepsilon/4) \cdot 4w_{\max} = (8+\varepsilon)w_{\max} \leq (8+\varepsilon)w(C_0)$. The above proves the claim, and so, the deterministic version of the result. In order to obtain a randomized 4-approximation, it suffices to pick $\varepsilon \leq 2$ and to output any cycle C' of H with four vertices (then, we output any of C, C' that has minimum weight). Up to some constant multiplicative increase of the number of edges to add in H, this can be done by using a randomized algorithm of Yuster and Zwick that runs in expected linear time [22, Theorem 2.9]. Note that this is the only source of randomness in the algorithm.

We recall that any *unweighted* graph with $\mathcal{O}(n^{1+\frac{1}{\ell}})$ edges contains a cycle of length at most 2ℓ. We could use this density result instead of Theorem 5.2. In doing so, we could use a much sparser subgraph H in the proof of Proposition 5.1. However, our algorithm would still run in time $\tilde{\mathcal{O}}(n^{5/3})$ because the bottleneck is our call to the algorithm of Theorem 1.2.

5.2. The algorithm. We combine Proposition 5.1 with the approach taken in Theorem 1.2, as follows:

THEOREM 1.3. *Let $G = (V, E, w)$ have sorted adjacency lists.*

1. *If all edge-weights are in $\{1, \ldots, M\}$ then, we can compute a deterministic 4-approximation for* GIRTH *in time $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}M)$.*
2. *If all edge-weights are non-negative then, we can compute a* randomized *4-approximation for* GIRTH *in time $\tilde{\mathcal{O}}(n^{5/3})$. For every $\varepsilon > 0$, we can also compute a deterministic $(4+\varepsilon)$-approximation for* GIRTH *in time $\tilde{\mathcal{O}}(n^{5/3}\text{polylog}1/\varepsilon)$.*

Proof. It suffices to prove the result for graphs with non-negative edge-weights and $\varepsilon > 0$ arbitrary. To see that, let us consider any graph G with all edge-weights in $\{1, 2, \ldots, M\}$. If we take G as the input of our deterministic algorithm for graphs with non-negative edge-weights, setting $\varepsilon = \frac{1}{M(n+1)}$, then, the cycle C of G that is outputted by the algorithm has total integer weight strictly upper-bounded by $1/\varepsilon$. In particular, $w(C)$ is in fact a 4-approximation of the girth of G. Thus, from now on we will only consider the more general case of graphs G with non-negative edge-weights.

Let $\varepsilon > 0$ be fixed. The randomized version of the theorem was proved in Proposition 5.1. For proving the deterministic version, we start computing a c-approximation g^* of the girth of G, for some universal constant $c > 4$. By Proposition 5.1, it can be done in time $\tilde{\mathcal{O}}(n^{5/3})$. Observe that the girth of G is somewhere between g^*/c and g^*. Now, define $\eta = \varepsilon/6$. Let $i_{\min}, i_{\max}$ be the smallest nonnegative integers such that $g^*/c \leq (1+\eta)^{i_{\min}}$ and in the same way $5g^* \leq (1+\eta)^{i_{\max}}$. We have $i_{\max} - i_{\min} = \mathcal{O}(1/\log(1+\eta)) = \mathcal{O}(1/\varepsilon)$. Furthermore, let $T = \{(1+\eta)^i \mid i_{\min} \leq i \leq i_{\max}\}$.

The remaining of the algorithm is essentially the same as for Theorem 1.2, except that we avoid the costly computation of the induced subgraphs G'_v. Specifically, our processing of the vertices in the set S (given by Proposition 2.1) remains unchanged. However, for every $v \in V \setminus S$, we compute the smallest $t \in T$ such that $t < dist_G(v, S)$ and $HBD(G, s, t)$ detects a cycle (if any). In doing so, we can only visit the vertices in $B_S(v)$, and so, the total running time for processing v is upper-bounded by $\tilde{\mathcal{O}}(|B_S(v)|\text{polylog}M)$, that is in $\tilde{\mathcal{O}}(n^{1/3}\text{polylog}M)$. Overall, this algorithm runs in total time $\tilde{\mathcal{O}}(n^{5/3}\log T)$, that is in $\tilde{\mathcal{O}}(n^{5/3}\log 1/\varepsilon)$.

Let $C \in \{C_v \mid v \in V\}$ be of minimum weight amongst all the cycles computed by the algorithm. We claim that $w(C)$ is a $(4+\varepsilon)$-approximation of the girth of G. In order to prove this claim, let C_0 be a shortest cycle of G. There are two cases.

- Assume there exists a vertex $s \in V(C_0) \cap S$. Then, $w(C) \leq w(C_s) \leq 2(1+\eta)w(C_0) < (4+\varepsilon)w(C_0)$ (this is a similar proof as for Theorem 1.2).
- Thus, from now on we assume $V(C_0) \cap S = \emptyset$. Let $v \in V(C_0)$ be arbitrary.
  - Assume first $w(C_0) < dist_G(v,S)/(1+\eta)$. By Corollary 3.3 applied for $dist_G(v,C_0) = 0$, the smallest t_v such that $\texttt{HBD}(G,v,t_v)$ detects a cycle satisfies $t_v \leq w(C_0)$. In particular, the smallest $t'_v \in T$ such that $\texttt{HBD}(G,v,t'_v)$ detects a cycle satisfies $t'_v \leq (1+\eta)w(C_0) < dist_G(v,S)$. By Lemma 3.1, $w(C) \leq w(C_v) \leq 2(1+\eta)w(C_0) < (4+\varepsilon)w(C_0)$.
  - Otherwise $w(C_0) \geq dist_G(v,S)/(1+\eta)$. Let $s \in S$ minimize $dist_G(s,v)$. Then, by Corollary 3.3, the smallest t_s such that $\texttt{HBD}(G,s,t_s)$ detects a cycle satisfies:

    $$t_s \leq dist_G(s,v) + w(C_0) = dist_G(v,S) + w(C_0) \leq (2+\eta)w(C_0).$$

    In particular, the smallest $t'_s \in T$ such that $\texttt{HBD}(G,s,t'_s)$ detects a cycle satisfies:

    $$t'_s \leq (1+\eta)t_s \leq (2+2\eta+\eta^2)w(C_0) \leq (2+3\eta)w(C_0).$$

    As a result, by Lemma 3.1 $w(C) \leq w(C_s) \leq (4+6\eta)w(C_0) \leq (4+\varepsilon)w(C_0)$.

Summarizing, $w(C) \leq (4+\varepsilon)w(C_0)$ in all the cases. □

6. Open problems. The most pressing question is whether we can achieve a 4/3-approximation for the girth in subquadratic time. If it is not the case then, what is the best approximation factor we can get in subquadratic time? We note that in [18], Roditty and Vassilevska Williams conjectured that we cannot achieve a $(2-\varepsilon)$-approximation already for unweighted graphs[5]. If their conjecture is true then, this would imply our algorithm is essentially optimal (at least for the non-dense graphs with $\mathcal{O}(n^{2-\varepsilon})$ edges). However, for the dense graphs with sorted adjacency lists, we left open whether a better approximation-factor than 4 can be obtained in $o(n^2)$-time. Finally, another interesting question is whether a constant-approximation for the girth can be computed in quasi linear time. We recall that this is wide open even for *unweighted* graphs [8].

Acknowledgements. This work was supported by a grant of Romanian Ministry of Research and Innovation CCCDI-UEFISCDI. project no. 17PCCDI/2018.

REFERENCES

[1] D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani, *Fast estimation of diameter and shortest paths (without matrix multiplication)*, SIAM Journal on Computing (SICOMP), 28 (1999), pp. 1167–1181.
[2] S. Baswana and S. Sen, *Approximate distance oracles for unweighted graphs in expected o (n 2) time*, ACM Transactions on Algorithms (TALG), 2 (2006), pp. 557–577.
[3] J. A. Bondy and U. S. R. Murty, *Graph theory*, Graduate Texts in Mathematics, 2008.
[4] W. Brown, *On graphs that do not contain a Thomsen graph*, Canadian Mathematical Bulletin, 9 (1966), pp. 1–2.
[5] S. Chechik, *Approximate distance oracles with constant query time*, in 46th Symposium on Theory of Computing (STOC 2014), ACM, 2014, pp. 654–663.
[6] S. Chechik, D. Larkin, L. Roditty, G. Schoenebeck, R. Tarjan, and V. Vassilevska Williams, *Better approximation algorithms for the graph diameter*, in 25th Symposium on Discrete Algorithms (SODA 2014), ACM/SIAM, 2014, pp. 1041–1052.

[5] They did obtain such an algorithm for *triangle-free* graphs.

[7] S. Chechik, Y. Liu, O. Rotem, and A. Sidford, *Constant girth approximation for directed graphs in subquadratic time*, in 52nd Symposium on Theory of Computing (STOC 2020), ACM, 2020, pp. 1010–1023.

[8] S. Dahlgaard, M. Knudsen, and M. Stöckel, *New subquadratic approximation algorithms for the girth*, Tech. Report arXiv:1704.02178, arXiv, 2017.

[9] G. Ducoffe, *Faster Approximation Algorithms for Computing Shortest Cycles on Weighted Graphs*, in 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019), vol. 132 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, pp. 49:1–49:13.

[10] A. Itai and M. Rodeh, *Finding a minimum circuit in a graph*, SIAM Journal on Computing (SICOMP), 7 (1978), pp. 413–423.

[11] A. Lingas and E. Lundell, *Efficient approximation algorithms for shortest cycles in undirected graphs*, Information Processing Letters (IPL), 109 (2009), pp. 493–498.

[12] J. Pachocki, L. Roditty, A. Sidford, R. Tov, and V. Vassilevska Williams, *Approximating cycles in directed graphs: Fast algorithms for girth and roundtrip spanners*, in 29th Symposium on Discrete Algorithms (SODA 2018), ACM/SIAM, 2018, pp. 1374–1392.

[13] M. Patrascu and L. Roditty, *Distance oracles beyond the thorup–zwick bound*, SIAM Journal on Computing (SICOMP), 43 (2014), pp. 300–311.

[14] S. Robinson, *Toward an optimal algorithm for matrix multiplication*, SIAM news, 38 (2005), pp. 1–3.

[15] L. Roditty, M. Thorup, and U. Zwick, *Deterministic constructions of approximate distance oracles and spanners*, in 32nd International Colloquium on Automata, Languages, and Programming (ICALP 2005), Springer, 2005, pp. 261–272.

[16] L. Roditty and R. Tov, *Approximating the girth*, ACM Transactions on Algorithms (TALG), 9 (2013), p. 15.

[17] L. Roditty and V. Vassilevska Williams, *Minimum weight cycles and triangles: Equivalences and algorithms*, in 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2011), IEEE, 2011, pp. 180–189.

[18] ———, *Subquadratic time approximation algorithms for the girth*, in 23rd Symposium on Discrete Algorithms (SODA 2012), ACM/SIAM, 2012, pp. 833–845.

[19] M. Thorup and U. Zwick, *Approximate distance oracles*, Journal of the ACM (JACM), 52 (2005), pp. 1–24.

[20] V. Vassilevska Williams and R. Williams, *Subcubic equivalences between path, matrix, and triangle problems*, Journal of the ACM (JACM), 65 (2018), pp. 1–38.

[21] C. Wulff-Nilsen, *Approximate distance oracles with improved preprocessing time*, in 23rd Symposium on Discrete Algorithms (SODA 2012), ACM/SIAM, 2012, pp. 202–208.

[22] R. Yuster and U. Zwick, *Finding even cycles even faster*, SIAM Journal on Discrete Mathematics (SIDMA), 10 (1997), pp. 209–222.