Faster approximation algorithms for computing shortest cycles on weighted graphs
Guillaume Ducoffe

To cite this version:
Guillaume Ducoffe. Faster approximation algorithms for computing shortest cycles on weighted graphs. 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019), Jul 2019, Patras, Greece. 10.4230/LIPIcs.ICALP.2019.44. hal-02173350

HAL Id: hal-02173350
https://hal.science/hal-02173350
Submitted on 4 Jul 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License
Faster approximation algorithms for computing shortest cycles on weighted graphs

Guillaume Ducoffe
National Institute for Research and Development in Informatics, Romania
The Research Institute of the University of Bucharest ICUB, Romania
University of Bucharest, Romania
guillaume.ducoffe@ici.ro

Abstract
Given an $n$-vertex $m$-edge graph $G$ with non-negative edge-weights, a shortest cycle of $G$ is one minimizing the sum of the weights on its edges. The girth of $G$ is the weight of such a shortest cycle.

We obtain several new approximation algorithms for computing the girth of weighted graphs:

1. For any graph $G$ with polynomially bounded integer weights, we present a deterministic algorithm that computes, in $	ilde{O}(n^{5/3} + m)$-time, a cycle of weight at most twice the girth of $G$. This matches both the approximation factor and – almost – the running time of the best known subquadratic-time approximation algorithm for the girth of unweighted graphs.

2. 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 $\tilde{O}(n^{5/3}\text{polylog}(1/\varepsilon) + m)$. 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.

3. 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 $\tilde{O}(n^{5/3})$-time randomized 4-approximation for graphs with non-negative edge-weights. This can be derandomized, thereby leading to an $\tilde{O}(n^{5/3})$-time deterministic 4-approximation for graphs with polynomially bounded integer weights, and an $\tilde{O}(n^{5/3}\text{polylog}(1/\varepsilon))$-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.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Graph algorithms analysis; Theory of computation → Approximation algorithms analysis

Keywords and phrases Girth, Weighted Graphs, Approximation Algorithms.

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.44


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

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

1 The $\tilde{O}(\cdot)$ notation suppresses polylogarithmic factors.
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 $O(n^3)$, and Vassilevska Williams and Williams [19] 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 $O(n^{2/\varepsilon})$-time $(4/3 + \varepsilon)$-approximation algorithm [15], 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 $O(n^2)$. In fact, when the edge-weights are integers bounded by some constant $M$, there is a non-combinatorial algorithm for computing the girth of any $n$-vertex graph $G$ in time $\tilde{O}(Mn^\omega)$ where $\omega$ stands for the the exponent of square matrix multiplication over a ring [16]. It is widely believed that $\omega = 2$ [13], 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 [10, 15]. 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 [15] — see Sec. 1.2 on the Related work for more details. Our first result is obtained for graphs with bounded integer edge-weights.

▶ Theorem 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 $\tilde{O}(n^{5/3}\log M + m)$.

Our starting point for Theorem 1 is a previous 2-approximation algorithm from Lingas and Lundell [10], 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 [18] 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 literature [14]). 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 $O(n^{1/3})$. 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 [10]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 [15] 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 so that it also applies to graphs with arbitrary non-negative edge-weights.

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

We note that in [15], 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 [15], 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 \( O(n^{1+\frac{1}{\ell}}) \) edges has girth at most \( 2\ell \), 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, in the full version of this paper, we prove by using elementary arguments that any approximation algorithm for the girth on weighted graphs must run in \( \Omega(m) \)-time. We study what happens if we have sorted adjacency lists3.

▶ Theorem 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 \textsc{Girth} in time \( \tilde{O}(n^{5/3}\text{polylog}M) \).

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

We observe that even assuming sorted adjacency lists, it is not clear whether the algorithm of Theorem 1 can be implemented to run in time \( \tilde{O}(n^{5/3}\text{polylog}M) \). 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

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 [10, 16], they mostly consider edges on some shortest-paths in \( G \) between a pre-defined vertex and the other vertices in the ball.

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.
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

Approximation algorithms for the girth. Itai and Rodeh were the first to study the Girth problem for unweighted graphs [9]. Among other results, they showed how to compute an additive +1-approximation of the girth in time $O(n^2)$. This was later completed by Lingas and Lundell [10], who proposed a randomized quasi 2-approximation algorithm for this problem that runs in time $O(n^{3/2}\sqrt{\log n})$. In [17], 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 $O(n^{5/3})$, 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, this dependency on $m$ can be removed while preserving the approximation factor 2. Very recently, new subquadratic-time approximation algorithms were proposed for Girth in unweighted graphs (see [7]). It is open whether one can achieve a constant-factor approximation for the girth in, say, $\tilde{O}(n^{1+o(1)})$-time.

Much less is known about the girth of weighted graphs. The first known subcubic approximation was the one of Lingas and Lundell [10], 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 [15]. 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 [10, 15].

More recently, a breakthrough logarithmic approximation of the girth of directed weighted graphs was obtained in [11].

Approximate distances. Finally, approximation algorithms for the girth are tightly related to the computation of approximate distances in weighted graphs. In a seminal paper [18], Thorup and Zwick showed that we can compute in expected time $O(mn^{1/k})$ an approximate distance oracle: that can answer any distance query in time $O(k)$ with a multiplicative stretch at most $2k - 1$. This has been improved in several follow-ups [2, 5, 12, 14, 20]. 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 [10] before presenting our main result (Theorem 1). Our algorithm is generalized to graphs with arbitrary weights in Section 4. Finally, we remove the dependency on the number of edges in the time complexity of our algorithms in Section 5. We conclude this paper with some open perspectives (Section 6). Due to space restrictions, some of the proofs are omitted. Full proofs can be found in our technical report [8].
2 Preliminaries

We refer to [3] for any undefined graph terminology. 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 $w : E \rightarrow \mathbb{R}^+$ 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 $\text{dist}_G(u, v)$ 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 $\text{dist}_G(v, S) := \min_{u \in S} \text{dist}_G(u, v)$.

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 $B_G(v, r) := \{u \in V \mid \text{dist}_G(u, v) \leq r\}$. Finally, an $r$-nearest set for $v$ is any $r$-set $N_r(v)$ such that, for any $x \in N_r(v)$ and $y \notin N_r(v)$, we have $\text{dist}_G(x, v) \leq \text{dist}_G(y, v)$.

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 latter can always be ensured up to an $O(m)$-time preprocessing.

2.1 The Hitting Set method

We gather many well-known facts in the literature, that can be found, e.g., in [14, 18, 1, 6]. All these facts are combined in order to prove the following useful result for our algorithms:

**Proposition 4.** 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 $B_S(v) := \{u \in V \mid \text{dist}(v, u) < \text{dist}(v, S)\}$ for every $v \in V$, such that the following two properties hold true:

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

It is well-known that a set $S$ as requested by Proposition 4 can be constructed randomly as follows: every vertex in $V$ is added in $S$ with equal probability $n^{-1/3}$ [18]. This construction was derandomized in [14, 18, 1, 6]. 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 observe that all these subgraphs can be obtained almost for free. Namely:

**Lemma 5 (folklore).** For every $G = (V, E, w)$ and $U \subseteq V$ we can compute the subgraph $G[U]$ induced by $U$ in time $\tilde{O}(|U|^2)$ (assuming ordered adjacency lists).

3 Case of graphs with bounded integer weights

This section is devoted to the proof of Theorem 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 [10]. 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 [10] 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 [9], 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 [10, Lemma 2]. We revisit the analysis of the algorithm in [10] for weighted graphs, and we prove that this algorithm also satisfies such a “closeness property”.

The HBD-algorithm from [10]. 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 $dist(s, u) \leq t$ (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 $O(n)$-time [10].

\begin{align*}
(a) & \quad HBD(G, s, t) \\
1: & \quad \text{for all } v \in V \text{ do} \\
2: & \quad d(v) \leftarrow \infty; \pi(v) \leftarrow \text{NIL} \\
3: & \quad d(s) \leftarrow 0; Q \leftarrow \{s\} \\
4: & \quad \text{while } Q \neq \emptyset \text{ do} \\
5: & \quad u \leftarrow \text{Extract-min}(Q) \\
6: & \quad \text{Controlled-Relax}(u, t) \\
(b) & \quad \text{Controlled-Relax}(u, t) \\
1: & \quad Q_u \leftarrow \text{sorted adj. list} \\
2: & \quad uv \leftarrow \text{Extract-min}(Q_u) \\
3: & \quad \text{while } d(u) + w_{uv} \leq t \text{ do} \\
4: & \quad \text{RelaxOrStop}(u, v) \\
5: & \quad uv \leftarrow \text{Extract-min}(Q_u) \\
(c) & \quad \text{RelaxOrStop}(u, v) \\
1: & \quad \text{if } d(v) \neq \infty \text{ then} \\
2: & \quad \text{return a cycle and stop} \\
3: & \quad \text{else} \\
4: & \quad d(v) \leftarrow d(u) + w_{uv} \\
5: & \quad Q \leftarrow Q \cup \{v\}
\end{align*}

\textbf{Lemma 6} (10). 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 [10, 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 $s$.

\textbf{Lemma 7}. 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 $\leq t$.

Based on Lemma 7, we state some bounds on the weight of the cycle detected using HBD. In particular, Corollary 9 will play a key role in the analysis of our algorithms.

\textbf{Corollary 8}. 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 $t_0 \leq dist(s, C) + w(C)$.

\textbf{Corollary 9}. 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 $\max_{v \in V(C)} \text{dist}_G(x, v) \geq \text{dist}_G(s, x) > 0$. Then, the minimum $t_0$ such that $HBD(G, s, t_0)$ detects a cycle satisfies $t_0 \leq w(C)$. 

Proof. Let $B_x = \{v \in V(C) \mid dist_C(x,v) < dist_C(x,s)\}$. Since we assume that we have $\max_{v \in V(C)} dist_C(x,v) \geq dist_C(x,s)$, $B_x \neq V(C)$. 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 7 we have $t_0 \leq dist_C(s,x) + \max\{w(P_1), w(P_2)\} = dist_C(s,x) + w(P_1) \leq dist_C(x,y) + w(P_1) = w(P_2) + w(P_1) = w(C)$. △

3.2 Subquadratic-time approximation

Proof of Theorem 1. We analyse the following Subquadratic-Approx algorithm:

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

1: Find the minimum $t \in [3; M \cdot |V(G)|]$ such that: 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 $(BS(v))_{v \in V}$ be as in Prop. 4.
2: for all $s \in S$ do
3: \hspace{1em} $C_s \leftarrow$ Approx-Girth$(G,s,M)$
4: \hspace{1em} return $C_s$.
5: for all $v \notin S$ do
6: \hspace{2em} Let $G'_v$ be induced by $BS(v)$.
7: \hspace{2em} $C_v \leftarrow$ Approx-Girth$(G'_v,v,M)$
8: \hspace{1em} return a shortest cycle in $\{C_v \mid v \in V\}$.

The algorithm starts precomputing a set $S \subseteq V$ and the open balls $(BS(v))_{v \in V}$ as described in Proposition 4. This takes time $\tilde{O}(n^{5/3})$, plus an additional preprocessing time in $\tilde{O}(m)$ for sorting the adjacencies 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 Approx-Girth$(G,s,M)$). We store the cycle $C_s$ outputted by 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 $\tilde{O}(n \log M)$ per vertex in $S$, and so, $\tilde{O}(n|S| \log M) = \tilde{O}(n^{5/3} \log M)$ 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 $BS(v)$. By Lemma 5, this subgraph can be computed in time $\tilde{O}(|BS(v)|^2) = \tilde{O}(n^{2/3})$ – assuming a preprocessing of the graph in time $O(m)$ for ordering the adjacency lists. We apply the same procedure as for the vertices in $S$ but, we restrict ourselves to the ball $BS(v)$. That is, we call Approx-Girth$(G'_v,v,M)$, and we denote by $C_v$ the cycle outputted by this algorithm. Since we restrict ourselves to a subgraph of order $O(n^{1/3})$, this takes total time $\tilde{O}(n \cdot (n^{2/3} + n^{1/3} \log M)) = \tilde{O}(n^{5/3} \log M)$. 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 \( s \in S \), we compute a cycle \( C_s \) of weight no more than twice the weight of a shortest cycle passing by \( s \). By Corollary 8, 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 6 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 8 applied for \( \text{dist}(v, C_0) = 0 \), the smallest \( t_v \) such that \( HBD(G'_v, v, t_v) \) detects a cycle satisfies \( t_v \leq w(C_0) \). By Lemma 6, \( w(C) \leq t_v \leq 2w(C_0) \).

- Otherwise \( V(C_0) \not\subseteq B_S(v) \). This implies that we have: \( \max_{u \in V(C_0)} \text{dist}_{C_0}(u, v) \geq \max_{w \in V(C_0)} \text{dist}_{C_0}(u, v) \geq \text{dist}_{C_0}(v, S) > 0 \). Furthermore, let \( s \in S \) minimize \( \text{dist}_{C_0}(s, v) \). Then, by Corollary 9, the smallest \( t_s \) such that \( HBD(G, s, t_s) \) detects a cycle satisfies \( t_s \leq w(C_0) \). As a result, by Lemma 6 \( w(C) \leq w(C_s) \leq 2w(C_0) \).

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 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 [15] and the techniques used are interesting in their own right. Then, based on a clever technique from [15], 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:

\( \triangleright \) Theorem 10 ([15]). For every \( G = (V, E, w) \) with arbitrary non-negative edge-weights, we can compute a \( (4/3 + \varepsilon) \)-approximation for Girth in time \( \tilde{O}(n^2/\varepsilon) \).

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. We omit the proof as it quite similar, but simpler, to the one of Proposition 12 (presented next).

\( \triangleright \) Lemma 11. Assume there exists an \( T(n, m) \)-time \( \alpha \)-approximation algorithm for Girth for graphs with positive edge-weights, where \( T(n, m) = \Omega(m) \). Then, there also exists an \( \tilde{O}(T(n, m)) \)-time \( \alpha \)-approximation algorithm for Girth for graphs with non-negative edge-weights.

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.

\( \triangleright \) Proposition 12. For every \( G = (V, E, w) \) with arbitrary positive edge-weights, we can compute an \( \tilde{O}(n^{2/3}) \)-approximation for Girth in time \( \tilde{O}(n^{5/3} + m) \).

\textbf{Proof.} Let \( S \) be as in Proposition 4. We show a significantly more elaborate method that uses \( S \) in order to approximate the girth. We divide this method into five main steps.

\textbf{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
As noted, Theorem 1 is that we cannot use directly the algorithm of Roditty and Tov for that. Indeed, in order to compute short cycles, the 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 $\text{dist}(v, s) = \text{dist}(v, S)$. As noted, e.g., in [18], 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 $O(m)$. See Fig. 3 for an example. In what follows, we show how to use this tree structure in order to compute short cycles.

**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 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 $\text{dist}(v, s) = \text{dist}(v, S)$. As noted, e.g., in [18], 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 $O(m)$. See Fig. 3 for an example. In what follows, we show how to use this tree structure in order to compute short cycles.

**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 $\text{dist}(s, u) + w_e + \text{dist}(v, s)$ 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 $O(m)$.

**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 $\ell(e) = \text{dist}(s_u, u) + w_e + \text{dist}(v, s_v)$.
2. Group all these above edges with their two ends in the same two subtrees. It takes time $O(m + \lvert S \rvert) = O(m + n^{2/3})$ 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 $\lvert E(s, s') \rvert \geq 2$ 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 $O(m)$. 

![Figure 3](image-url) An example of Step 2. The two vertices in $S$ are drawn as rectangles.
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 \( \leq 3w(C^*_{s,s'}) \).

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

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

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 \( O(m + |S|^2) = O(m + n^{2/3}) \).

Furthermore, by Theorem 10 we can compute a constant-factor approximation of the girth of \( H_S \) in time \( O(|S|^2) = O(n^{4/3}) \). 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 \( \text{dist}(s',u) + w_{uv} + \text{dist}(v,s') \) and \( \text{dist}(s,x) + w_{x'v} + \text{dist}(y,s'') \), 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 10).

Overall, let \( C_{\text{min}} \) 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 \( w(C_{\text{min}}) \) is an \( O(n^{2/3}) \)-approximation of the girth of \( G \). By Claims 14 and 15, 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 \( T_s \). Write \( C_0 = (v_0, v_1, \ldots, v_p, v_0) \) 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 \( P_0, P_1, \ldots, P_{p-1}, p \leq p \) that are contained into the vertex-set of a same subtree \( T_s \) (in particular, \( v_0, v_p \in V(P_0) \) and \( v_{p-1} \in V(P_{p-1}) \)). Furthermore for every \( j \in \{0, 1, \ldots, q-1 \} \) let \( s_j \in S \) be such \( V(P_j) \subseteq V(T_s) \), 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 \( i_{j_1} = s_{j_1+1} \) and for every \( j \in \{j_1, j_1 + 1, \ldots, j_2\} \) the \( s_j \)'s are pairwise different (indices are taken modulo \( q \)). Then, two cases may arise:

- **Case \( j_2 = j_1 + 1 \).** We have: \( c_{j_1} := v_{i_{j_1}}v_{i_{j_1}+1}, c_{j_2} := v_{i_{j_1}+1}v_{i_{j_2}} \in E(s_{j_1}, s_{j_2}) \). Furthermore, \( C_0 \) goes by \( v_{i_j} \) (by \( v_{i_{j_1}+1}v_{i_{j_1}+1}, v_{i_{j_2}} \) respectively), and so, by Claim 13, either we computed a short cycle of weight \( \leq 2w(C_0) \) during Step 1, or we have \( w(C_0) \geq 2 \cdot \max\{\text{dist}(s_{j_1}, v_{i_{j_1}}), \text{dist}(s_{j_1}, v_{i_{j_2}+1}), \text{dist}(s_{j_2}, v_{i_{j_1}+1}), \text{dist}(s_{j_2}, v_{i_{j_2}})\} \). In the latter case, there exists a cycle of weight \( \leq \text{dist}(s_{j_1}, v_{i_{j_1}}) + w_{c_{j_1}} + \text{dist}(s_{j_2}, v_{i_{j_1}+1}) + \text{dist}(s_{j_2}, v_{i_{j_2}}) + w_{c_{j_2}} + \text{dist}(s_{j_1}, v_{i_{j_2}+1}) \leq 3w(C_0) \) that is fully contained in \( V(T_{s_{j_1}}) \cup V(T_{s_{j_2}}) \). By Claim 15, we so computed a cycle of weight \( \leq 9w(C_0) \) at Step 4.

- **Case \( j_2 \neq j_1 + 1 \).** 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 \( s_{j}v_{i_{j+1}} \in E_S \) has weight no more than \( \text{dist}(s_{j}, v_{i_j}) + w_{e_j} + \text{dist}(v_{i_j+1}, s_{j+1}) \) 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 13, either we computed a short cycle of weight \( \leq 2w(C_0) \) during Step 1, or we have \( w(C_0) \geq 2 \cdot \max\{\text{dist}(s_{j}, v_{i_j}), \text{dist}(s_{j+1}, v_{i_{j+1}})\} \) 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:

\[
\leq w(C_0) + \sum_{j = j_1}^{j_2} (\text{dist}(v_{i_{j_1+1}}, s_j) + \text{dist}(s_j, v_{i_j}))
\]

\[
\leq w(C_0)(1 + (j_2 - j_1 + 1))
\]

\[
\leq w(C_0)|S| = \tilde{O}(n^{2/3} \cdot w(C_0)).
\]

Then, let \(C_H\) be a cycle of \(H_S\) such that \(w(C_H) = \tilde{O}(n^{2/3} \cdot w(C_0))\) (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 \(w(C_H^*) \leq w(C_H) = \tilde{O}(n^{2/3} \cdot w(C_0))\).

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

### 4.2 Improving the approximation factor

**Sketch Proof of Theorem 2.** We may assume that all weights are positive by Lemma 11. Let \(g^*\) be the \(\tilde{O}(\sqrt[n]{2})\)-approximation that we computed by using Proposition 12. There exists some (known) constant \(c\) such that the girth of \(G\) is somewhere between \(g^*/(cn^{2/3} \log n)\) and \(g^*\). Then, let \(i_{\text{min}}, i_{\text{max}}\) be the smallest nonnegative integers such that \(g^*/(cn^{2/3} \log n) \leq (1 + \varepsilon/2)^{i_{\text{min}}}\) and in the same way \(g^* \leq (1 + \varepsilon/2)^{i_{\text{max}}}\). We have that:

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

Let \(S\) be as in Proposition 4. 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 10, it can be done in time \(\tilde{O}(n^{2/3})\) for each \(v\), and so, this takes total time \(\tilde{O}(n^{5/3})\). Then, let \(T = \{(1 + \varepsilon/2)^i \mid i_{\text{min}} \leq i \leq i_{\text{max}}\}\). 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 \(\tilde{O}(|S|n \log |T|) = \tilde{O}(n^{5/3} \log 1/\varepsilon)\) by using a dichotomic search. Finally, let \(g_{\text{min}}\) 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, that the girth of \(G\) is at least \(g_{\text{min}}/(2 + \varepsilon)\).

### 5 A subquadratic algorithm for dense graphs

A drawback of the algorithms in Theorems 1 and 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 [10, 15]. The main result of this section is that assuming sorted adjacency lists, the dependency on \(m\) can always be discarded (Theorem 3). Due to lack of space, we will only prove the following weaker result:

**Proposition 16.** For every \(G = (V, E, w)\) with non-negative edge-weights and sorted adjacency lists, we can compute:

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

---

4 In fact, this is already done in the proof of Proposition 12, but we restate it here for completeness of the method.
Roughly, we can prove Theorem 3 by combining this above Proposition 16 with a natural modification of the algorithm presented in Section 3.2.

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

**Theorem 17 ([4])**. Every unweighted graph with order $n$ and $m \geq (\frac{1}{2} + o(1)) n^{3/2}$ edges contains a cycle of length four.

**Proof of Proposition 16.** If $G$ has $m = \tilde{O}(n^{5/3})$ edges then, we can simply apply Theorem 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 $\left(\frac{1}{2} + o(1)\right) n^{3/2}$ edges of minimum weight in $G$.

We claim that $H$ can be constructed in time $\tilde{O}(n^{3/2})$ 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 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:

1. Assume there exists a shortest cycle $C_0$ of $G$ such that $E(C_0) \subseteq E(H)$. By Theorem 2, $w(C) \leq (2 + \varepsilon/4)w(C_0) < (8 + \varepsilon)w(C_0)$.

2. 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 $w_{\text{max}}$, where $w_{\text{max}}$ denotes the maximum-weight of an edge in $H$. In particular, the girth of $G$ is at least $w_{\text{max}}$. Furthermore, since $H$ has enough edges by construction, by Theorem 17 it contains a cycle of four vertices; the latter has weight at most $4w_{\text{max}}$. As a result, $w(C) \leq (2 + \varepsilon/4) \cdot 4w_{\text{max}} = (8 + \varepsilon)w_{\text{max}} \leq (8 + \varepsilon)w(C_0)$.

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 [21, Theorem 2.9]. Note that this is the only source of randomness in the algorithm.

We recall that any unweighted graph with $O(n^{1+\frac{1}{13}})$ edges contains a cycle of length at most $2\ell$. We could use this density result instead of Theorem 17. In doing so, we could use a much sparser subgraph $H$ in the proof of Proposition 16. However, our algorithm would still run in time $\tilde{O}(n^{5/3})$ because the bottleneck is our call to the algorithm of Theorem 2.

**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 [17], 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 \(O(n^{2-\varepsilon})\) 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 [7].

---

References


\(^5\) They did obtain such an algorithm for triangle-free graphs.