

New Additive Approximations for Shortest Paths and Cycles

Mingyang Deng ✉

Massachusetts Institute of Technology, Cambridge, MA, USA

Yael Kirkpatrick ✉


Massachusetts Institute of Technology, Cambridge, MA, USA

Victor Rong ✉ 

Massachusetts Institute of Technology, Cambridge, MA, USA

Virginia Vassilevska Williams ✉

Massachusetts Institute of Technology, Cambridge, MA, USA

Ziqian Zhong ✉ 

Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

This paper considers additive approximation algorithms for All-Pairs Shortest Paths (APSP) and Shortest Cycle in undirected unweighted graphs. The results are as follows:

- We obtain the first $+2$ -approximation algorithm for APSP in n -vertex graphs that improves upon Dor, Halperin and Zwick's (SICOMP'00) $\tilde{O}(n^{7/3})$ time algorithm. The new algorithm runs in $\tilde{O}(n^{2.29})$ time and is obtained via a reduction to Min-Plus product of bounded difference matrices.
- We obtain the first additive approximation scheme for Shortest Cycle, generalizing the approximation algorithms of Itai and Rodeh (SICOMP'78) and Roditty and Vassilevska W. (SODA'12). For every integer $r \geq 0$, we give an $\tilde{O}(n + n^{2+r}/m^r)$ time algorithm that returns a $+(2r + 1)$ -approximate shortest cycle in any n -vertex, m -edge graph.

2012 ACM Subject Classification Theory of computation \rightarrow Graph algorithms analysis

Keywords and phrases Fine-grained Complexity, Additive Approximation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.50

Category Track A: Algorithms, Complexity and Games

Funding *Virginia Vassilevska Williams*: Partially funded by NSF awards 2129139 and 1909429, BSF award 2020356, and a Google Research Fellowship.

1 Introduction

The all-pairs shortest paths problem (APSP) is among the most fundamental problems in algorithms. The fastest algorithms for the problem in n vertex, m edge graphs with integer edge weights and no negative cycles, run in $n^3/2^{\Theta(\sqrt{\log n})}$ time [22] and in $O(mn + n^2 \log \log n)$ time [14]. These running times are believed to be optimal, up to $n^{o(1)}$ factors (see [19, 12]).

APSP in unweighted graphs has long been known to admit faster algorithms. In undirected graphs, Seidel [16] gave an $\tilde{O}(n^\omega)$ time algorithm, where $\omega < 2.373$ [2] is the exponent of square matrix multiplication. This running time is believed to be optimal, as APSP in undirected unweighted graphs is at least as hard as Boolean Matrix Multiplication (BMM).¹

¹ In fact, later algorithms by Shoshan and Zwick [17] imply that undirected unweighted APSP is equivalent to BMM.



Due to the impracticalities of fast matrix multiplication and since it is unavoidable for the exact computation of APSP, it is natural to consider approximation algorithms, trying to get running times as close to n^2 as possible (n^2 is the size of the output). There is significant work on multiplicative approximations of APSP and distance oracles (e.g. [18, 6, 1]). However, for unweighted undirected graphs, more desirable fast *additive* approximations are also possible. In a $+C$ -approximation to APSP, one returns estimates $d'(u, v)$ for the distance $d(u, v)$ between any pair of vertices u, v , so that $d(u, v) \leq d'(u, v) \leq d(u, v) + C$. We let $+C$ -APSP denote the problem of computing a $+C$ -approximation to APSP in an undirected unweighted graph.

Following work of Aingworth et al. [1], so far the best approximation-running time tradeoff for unweighted undirected APSP is achieved by Dor, Halperin and Zwick [7]: For every even integer $k \geq 2$, there is an $\tilde{O}(n^{2+2/(3k-2)})$ time algorithm for $+k$ -APSP. (For sparser graphs, the improved running time $\tilde{O}(n^{2-2/(k+2)}m^{2/(k+2)})$ is also given.)

In particular, Dor, Halperin and Zwick provide a $+2$ -approximation algorithm that runs in $\tilde{O}(n^{7/3})$ time. The algorithm is simple and “combinatorial”, and notably is *faster* than the fastest exact algorithm for APSP by Seidel for the current bounds on ω . Aingworth et al. [1] showed that $+1$ -approximating APSP is at least as hard as BMM, and thus likely requires $n^{\omega-o(1)}$ time, so beating n^ω is only possible if the additive approximation is at least 2.

There are no known conditional lower bounds for $+2$ -approximations of APSP, and the Dor, Halperin and Zwick running time has remained unchallenged for over *three decades*.

Is $n^{7/3-o(1)}$ time necessary for $+2$ -APSP? Or, can one do better?

Our first result is the first *improvement* over the $+2$ -approximation algorithm of [7].

► **Theorem 1.** *There is an $O(n^{2.2867})$ time algorithm that returns a $+2$ -approximation to APSP in undirected unweighted n vertex graphs.*

We obtain the new tradeoff by reducing the $+2$ -APSP problem to Min-Plus product of (rectangular) Bounded Difference Matrices. While Min-Plus product for $n \times n$ matrices is believed to require $n^{3-o(1)}$ time (see e.g. [19]), a string of recent papers has developed faster and faster truly subcubic time algorithms for Min-Plus product for Bounded Difference Matrices [4, 5, 8, 21]. Our reduction is completely black-box, so that if there is an improved algorithm for the Min-Plus product of an integer matrix with a matrix with bounded difference columns or rows, then this improvement would immediately translate into an improved algorithm for $+2$ -APSP.

Vassilevska W. and Williams [20] showed that APSP in weighted graphs is fine-grained equivalent to the Girth problem, which asks to compute the length of the shortest cycle in a given undirected graph. The same paper [20] shows that the Girth problem in *undirected unweighted* graphs is at least as hard as triangle detection, and is known to be subcubically equivalent to undirected unweighted APSP. As triangle detection in n vertex graphs is believed to require $n^{\omega-o(1)}$ time (e.g. [19]), so is Girth, and thus faster approximation algorithms for the Girth are also well-motivated and well-studied. Multiplicative approximation schemes are well-understood (e.g. [10, 13, 15]). Meanwhile, unlike for APSP, there is *no known tradeoff* of additive approximation algorithms for the girth of undirected unweighted graphs.

There are only two known additive approximation results for Girth in undirected unweighted graphs. First, Itai and Rodeh [9] showed that a $+1$ -approximation to the girth can be obtained in $O(n^2)$ time. Then, Roditty and Vassilevska W. [15] showed that a $+3$ -approximation to the girth in n vertex, m edge graphs, can be computed in $\tilde{O}(n^3/m)$ time. *Can one generalize these two algorithms to a scheme?*

Our second result is to obtain the first additive approximation scheme for the girth.

► **Theorem 2.** *Let $G = (V, E)$ be an unweighted, undirected graph with $|V| = n, |E| = m$. Let r be an integer and denote the (unknown) girth of G by g . There is an $\tilde{O}\left(n + \frac{n^{2+r}}{m^r}\right)$ time algorithm that returns with high probability a cycle of length \hat{g} that satisfies $g \leq \hat{g} \leq g + (2r + 1)$.*

Thus, for instance, there is an $\tilde{O}(n + n^4/m^2)$ time +5-approximation algorithm. This running time is always better than the previously known additive approximations as long as $m \geq \Omega(n^{1+\varepsilon})$ and $m \leq O(n^{1.5-\varepsilon})$ for some $\varepsilon > 0$. More generally, as r grows, each $+(2r + 1)$ approximation is always faster than the approximations for smaller r for the sparsity range $m \in [\Omega(n^{1+\varepsilon}), O(n^{1+1/r-\varepsilon})]$ for any arbitrarily small $\varepsilon > 0$.

2 Additive Approximation Algorithm for APSP

We first formally define $+C$ -approximation of APSP and $(\min, +)$ matrix product.

► **Definition 3** (APSP $+C$ -approximation). *For an undirected unweighted simple graph $G = (V, E)$, output $\hat{d} : V \times V \rightarrow \mathbb{N}$ such that $d(u, v) \leq \hat{d}(u, v) \leq d(u, v) + C$, where $d(u, v)$ is the distance from u to v in graph G .*

► **Definition 4** ($(\min, +)$ matrix product). *$(\min, +)$ matrix product between matrix A and B is defined as $C = A \star B$ where $C[i, j] = \min_k \{A[i, k] + B[k, j]\}$.*

While in general, no sub-cubic algorithm has been found for $(\min, +)$ matrix product, many special cases have been addressed (e.g. [4] [21] [5]). Specifically, we consider the case between column bounded-difference and row bounded-difference matrix.

► **Definition 5** (Column bounded-difference, Row bounded-difference). *A matrix A is column bounded-difference if there exists some constant C so that $|A[i, j] - A[i + 1, j]| \leq C$ for all valid (i, j) 's. Symmetrically, a matrix A is row bounded-difference if there exists some constant C so that $|A[i, j] - A[i, j + 1]| \leq C$ for all valid (i, j) 's.*

► **Definition 6** ($(\min, +)$ matrix product between column bounded-difference matrix and row bounded-difference matrix). *Given column bounded-difference matrix A of size $n \times m$ and row bounded-difference matrix B of size $m \times n$, calculate their $(\min, +)$ matrix product $A \star B$. Call the time complexity for such a problem $MPCRBD(n, m)$.*

A similar case was previously addressed by Bringmann et al. [4] and an algorithm of runtime $O(n^{2.9217})$ is given (Theorem 1.3, [4]). By adapting the method of Chi, Duan and Xie [5], we can get the following bound.

We want to point out that since our reduction is black-box, the use of the following lemma is not necessary and applying the algorithm in Bringmann et al. [4] also gives a $O(n^{7/3-\Omega(1)})$ algorithm. Therefore, we defer the proof of this lemma to Appendix A.

► **Lemma 7** (Appendix A). *Let $M(n, u, n)$ be the time to multiply $n \times u$ and $u \times n$ matrices. For parameters $\alpha, \beta, \gamma > 0$, $MPCRBD(n, m) = \tilde{O}(n^2 m / \alpha^2 + n^2 \beta + M(n, \gamma, n) \alpha m / \beta + n^2 m^2 / \gamma)$. Specifically, $MPCRBD(n, n) = O(n^{2.811})$.*

We start with Dor, Halperin and Zwick's original algorithm, used as a black-box.

► **Lemma 8** (Theorem 3.1, [7]). *There is an algorithm for APSP $+2$ -approximation that runs in $\tilde{O}(|V|^{3/2}|E|^{1/2})$ time on input graph $G = (V, E)$.*

For a graph $G = (V, E)$, define $N(v) = \{u \mid (u, v) \in E\}$ as the set of adjacent vertices of v for $v \in V$. We also use the following lemma.

► **Lemma 9** (e.g. Theorem 1, [1]). *For a graph $G = (V, E)$ and a parameter s , let $V_s = \{v \mid v \in V, |N(v)| \geq s\}$, we can deterministically compute a hitting set $D \subseteq V$ where $|D| = O(|V| \log |V|/s)$ and $N(v) \cap D \neq \emptyset$ for all $v \in V_s$ in $O(|V|^2)$ time.*

Crucial to our algorithm is to consider an Euler tour of a spanning tree.

► **Definition 10** (Euler Tour). *For a spanning tree $T \subseteq E$ of a connected graph $G = (V, E)$, an Euler tour of T is a sequence of vertices $v_1, v_2, \dots, v_{2|V|-1}$, where each vertex of G appears at least once and the edges $(v_i, v_{i+1}) \in T$ for all $1 \leq i \leq 2|V| - 2$.*

Given a tree T , an Euler tour of T could be easily found by running depth-first search on T in $O(|T|)$ time.

We first give a high-level overview of our improvement. In Theorem 3.2, [7], a hitting set D_1 is computed to update the distance between all pairs of nodes $(u, v) \in V^2$. That is, for all $u, v \in V$, we update $d(u, v)$ with $\min_{t \in D_1} (d(u, t) + d(t, v))$. Notice that if we consider V, D_1, V as three dimensions in a matrix multiplication, the updating process is essentially a min-plus matrix product. Now if we arrange u, v in the order of an Euler tour, the matrices would then be (row/column) bounded difference. Thus, we can gain a speedup by aforementioned algorithms. To calculate distances between D_1 and V efficiently, we partition nodes in V by their degree.

For a graph $G = (V, E)$, let $n = |V|$, $m = |E|$. We set a parameter $l = O(\log n)$ to be determined, and define $s_i = n/2^{i-1}$ for $i \in [1, l]$. Specifically, let $s_0 = n + 1$.

By Lemma 9, we can find hitting sets D_i of size $\tilde{O}(n/s_i)$ for all vertices with degree $\geq s_i$ in G in $\tilde{O}(n^2)$ time. Let $f_i(u)$ be any element in $D_i \cap N(u)$ for $\deg u \geq s_i$, let F_i be the set of edges $(u, f_i(u))$ where $\deg u \geq s_i$ ($\deg v$ stands for the degree of vertex v), and $F = \cup_{i=0}^l F_i$. Thus $|F_i| = O(n)$, $|F| = \tilde{O}(n)$.

We define a series of graphs $G_0, G_1, G_2, \dots, G_l$. Define $G_0 = G$, and $G_i = (V, E_i)$ where $E_i = F \cup \{(u, v) \in E \mid \deg u \leq s_i \text{ or } \deg v \leq s_i\}$ for $1 \leq i \leq l$. Let $d_i(u, v)$ be the distance from u to v on graph G_i .

We then run breadth-first search on graph G_{i-1} from each vertex in set D_i to obtain the distances $d_{i-1}(t, v)$ for all pairs $(t, v) \in D_i \times V$. Define $w_i(u, v) = \min_{t \in D_i} \{d_{i-1}(t, u) + d_{i-1}(t, v)\}$. Let $g(u, v)$ to be the output of algorithm in Lemma 8 on input G_l . Let $h(u, v) = \min(\min_{i=1}^l \{w_i(u, v)\}, g(u, v))$.

We now argue that h is the desired approximation to distances, i.e. $d(u, v) \leq h(u, v) \leq d(u, v) + 2$ for all $u, v \in V$. Note that $w_i(u, v), g(u, v) \geq d(u, v)$ since their values all arise from valid paths, thus $h(u, v) \geq d(u, v)$.

Now we show that $d(u, v) + 2 \geq h(u, v)$. Consider the shortest path p from u to v in graph G . Let r be the node with maximum degree on p . If $\deg r \geq s_l$, assume that $s_a \leq \deg r < s_{a-1}$ for some $1 \leq a \leq l$. Since all nodes on p have degree $< s_{a-1}$, $d(u, v) = d_{a-1}(u, v)$. In G_{a-1} , consider $g = f_a(r)$, we have $h(u, v) \leq w_a(u, v) \leq d_{a-1}(u, g) + d_{a-1}(g, v) \leq d_{a-1}(u, r) + 1 + d_{a-1}(r, v) + 1 = d_{a-1}(u, v) + 2 \leq d(u, v) + 2$. Otherwise (if $\deg r < s_l$), p is preserved in the graph G_l and $g(u, v) \leq d(u, v) + 2$ by Lemma 8. Therefore, $h(u, v) \leq d(u, v) + 2$.

We now show how to compute w_i for a given $1 \leq i \leq l$. Without loss of generality, assume G_{i-1} is connected (otherwise we can treat each connected component separately), and let T be any spanning tree of G_{i-1} . Let $t_1, t_2, t_3, \dots, t_{2n-1}$ be an arbitrary Euler tour of T and suppose $D_i = \{x_1, x_2, \dots, x_{|D_i|}\}$. Define A to be a matrix of size $(2n-1) \times |D_i|$, where $A(i, j) = d_{i-1}(t_i, x_j)$. Let $B = A \star A^T$, then by definition, $w_i(u, v) = B[\text{pos}(u), \text{pos}(v)]$ where $t_{\text{pos}(a)} = a$ (any suffice).

Note that $|A[a, j] - A[a+1, j]| \leq 1$ since $|d_{i-1}(t_a, x_j) - d_{i-1}(t_{a+1}, x_j)| \leq d_{i-1}(t_a, t_{a+1}) = 1$, so A is column bounded-difference, A^T is row bounded-difference. The time complexity to compute $f(u, v)$ for all $u, v \in V$ is then $O(n^2 + \text{MPCRB}(n, |D_i|))$.

The total runtime of the above algorithm consists of three parts.

1. The computation of $\{D_i\}$, $\{G_i\}$, $\{F_i\}$ takes $\tilde{O}(n^2)$ time.
2. Compute w_i for all $1 \leq i \leq l$. Constructing the depth-first search tree and Euler tour takes $\tilde{O}(n^2)$ time. Since $|E_i| \leq s_i n$, the breadth-first search for each D_i in G_{i-1} takes $O(|D_i|s_i n) = \tilde{O}(n^2)$ time, so this part also takes $\tilde{O}(n^2)$ time. The $(\min, +)$ matrix products takes time $\tilde{O}(n^2 + \text{MPCRBD}(n, |D_l|))$ since $|D_l| \geq |D_i|$ for any $1 \leq i \leq l$.
3. The computation of g takes time $N^{3/2}|E_l|^{1/2} = \tilde{O}(n^2 s_l^{1/2})$.

Let s_l be the power of 2 closest to t , we obtain the following result:

► **Theorem 11.** *For a parameter $t \geq 1$, there is an algorithm for APSP +2-approximation on input graph $G = (V, E)$ that runs in $\tilde{O}(n^{2t^{1/2}} + \text{MPCRBD}(n, n/t))$ time, where $n = |V|$, $m = |E|$.*

By Lemma 7, we could obtain an upper bound for $\text{MPCRBD}(n, m)$.

► **Theorem 12.** *There is an algorithm for APSP +2-approximation on input graph $G = (V, E)$ that runs in $O(n^{2.2867})$ time where n stands for $|V|$.*

Proof. In Theorem 11, set $t = n^{0.57339}$, and apply Lemma 7 with $\alpha = n^{0.07006}$, $\beta = n^{0.28662}$, $\gamma = n^{0.56688}$. $M(n, \gamma, n) = M(n, n^{0.56688}, n) = O(n^{2.076433})$ by [11]. The claimed bound then follows from a direct computation. ◀

We also provide the following simpler bound using square MPCRBD . Plugging in Theorem 1.3 in Bringmann et al. [4] gives an algorithm of $O(n^{2.32416})$.

► **Corollary 13.** *Suppose $\text{MPCRBD}(n, n) = O(n^{2+\alpha})$ for constant α , there is an algorithm for APSP +2-approximation on input graph $G = (V, E)$ that runs in $\tilde{O}(|V|^{2+\frac{\alpha}{1+2\alpha}})$ time.*

Proof. Let $n = |V|$. $\text{MPCRBD}(n, n/t) = O(t^2 \text{MPCRBD}(n/t, n/t)) = O(n^{2+\alpha}/t^\alpha)$. Set $t = n^{\alpha/(1/2+\alpha)}$, the total runtime would then be $\tilde{O}(n^{2+\frac{\alpha}{1+2\alpha}})$. ◀

3 Additive Approximation Algorithm for Girth

The first additive approximation algorithm for the girth of a graph was presented by Itai and Rodeh in 1978 [9]. In their paper they showed an $O(n^2)$ time algorithm that, given a graph of girth g , returns a cycle of length at most $g + 1$.

The next improvement, an additive approximation algorithm running in subquadratic time, was presented in 2012 by Roditty and Vassilevska W. [15]. Their algorithm provided a +3 approximation of the girth in $\tilde{O}(n^3/m)$ time.

Both algorithms use a subroutine which we will call BFS-cycle. This subroutine runs BFS from a given vertex s until it reaches some vertex v for a second time. When v is reached for the second time, BFS-cycle returns the cycle enclosed by the two paths between s and v . We will use the following result regarding this algorithm:

► **Lemma 14** (e.g. [9, 13]). *BFS-cycle(v) runs in $O(n)$ time. If a vertex v is at distance ℓ from a vertex on a simple cycle of length k , then BFS-cycle(v) reports a cycle of length at most $k + 2\ell + 1$. If k is even, BFS-cycle(v) reports a cycle of length at most $k + 2\ell$.*

Another tool used in Roditty and Vassilevska W.'s algorithm is a result in extremal graph theory proved by Bondy and Simonovits [3], regarding even cycles in a graph:

► **Theorem 15** ([3]). *Let $k \geq 2$ be an integer. If an n -node graph G has at least $100kn^{1+1/k}$ edges then G contains a $2k$ -cycle.*

As a result of this theorem, for any graph with at least $200kn^{1+1/k}$ edges, at least half of the edges are part of some $2k$ -cycle. Therefore, if we uniformly sample an edge, it is part of a $2k$ -cycle with probability $\geq \frac{1}{2}$. By running BFS-cycle from each sampled edge and taking the lowest result, we obtain a randomized algorithm with efficient runtime:

► **Lemma 16** (e.g. [23]). *If an n -node graph G has at least $200kn^{1+1/k}$ edges, then there exists an $O(n \log n)$ time algorithm that finds a cycle of length at most $2k$ with high probability.*

Using these methods, we provide an algorithm for any odd additive approximation. In the following theorem we generalize the algorithm of Roditty and Vassilevska W. [15] to provide an additive $+(2r + 1)$ approximation to the girth for any integer r .

► **Theorem 17.** *Let $G(V, E)$ be an unweighted, undirected graph with $|V| = n, |E| = m$. Let r be an integer and denote the girth of G by g . There is an $\tilde{O}\left(n + \frac{n^{2+r}}{m^r}\right)$ time algorithm that returns with high probability a cycle of length \hat{g} that satisfies $g \leq \hat{g} \leq g + 2r + 1$.*

Proof. First we consider the case where $m < 200Ln^{1+1/L}$ for $L = \left\lceil \frac{\log n}{\log \log n} \right\rceil$. This implies that $m \leq \tilde{O}(n)$. We can use the $O(n^2)$ time algorithm of Itai and Rodeh to obtain an additive $+1$ approximation of the girth of G . Notably, in this case $O(n^2) \leq \tilde{O}\left(\frac{n^{2+r}}{m^r}\right)$.

Suppose now that $m \geq 200Ln^{1+1/L}$. Then there exists an integer $k \leq L$ such that

$$200(k+1)n^{1+\frac{1}{1+k}} < m \leq 200kn^{1+\frac{1}{k}}.$$

It follows from Lemma 16 that in $O(n \log n)$ time we can find a cycle of length at most $2k + 2$ with high probability. If $g > 2k - 2r$, this cycle is an additive $+(2r + 1)$ approximation of the girth.

We are left to handle the case when $g \leq 2k - 2r$. For any non-negative integer p , denote by $T_p(v) := \{u \in V : d(u, v) \leq p\}$ the vertices in the graph of distance $\leq p$ from v . Let Δ be a degree parameter; we will refer to vertices with $|T_r(v)| \leq \Delta$ as low r -degree vertices and vertices with $|T_r(v)| > \Delta$ as high r -degree vertices.

We sample a set S of $O\left(\frac{n}{\Delta} \log n\right)$ vertices uniformly at random. With high probability, any high degree vertex v satisfies $T_r(v) \cap S \neq \emptyset$. We now run BFS-cycle from each vertex of S . If the shortest cycle in the graph contains some node v such that $T_r(v)$ intersects S , Lemma 14 implies that the shortest cycle the algorithm finds is of length at most $g + 2r + 1$. The running time of this is $O\left(\frac{n^2 \log n}{\Delta}\right)$.

Now we only need to handle the case where the shortest cycle contains only vertices v for which $T_r(v) \cap S = \emptyset$, or equivalently $v \notin T_r(s) \forall s \in S$. To do so, we remove from the graph all vertices in $T_r(s)$ for any $s \in S$. With high probability the remaining vertices are all of low r -degree.

For each remaining vertex v , consider performing BFS from v up to ℓ levels for some $k - r \leq \ell < k$, while keeping track of the cumulative size of the layers. We break at the first $\ell \geq k - r$ for which $|T_\ell(v)| \leq \Delta^{(k-r)/r} |T_{\ell-(k-r)}(v)|$. Since we are assuming $g \leq 2k - 2r$, if there is a vertex $u \in T_{\ell-(k-r)}(v)$ which is part of the shortest cycle, then the BFS from v up to ℓ levels must have found a $+2(\ell - (k - r)) + 1$ approximation of this. In Lemma 18, we justify why we break before $\ell = k$. Then, we discard all vertices in $T_{\ell-(k-r)}(v)$ from further consideration. We do this for all v , and the total time complexity is $O(n\Delta^{(k-r)/r})$.

The proof that this algorithm is valid is deferred to Lemma 18. So if the shortest cycle in G is comprised of only low r -degree vertices, we will have found a $+2r + 1$ approximation of it. The full algorithm for this is described in Algorithm 1.

■ **Algorithm 1** All Low BFS-Cycle.

```

for  $s \in V$  do
   $L_0 \leftarrow \{s\}$ 
   $S_0 \leftarrow 1$ 
  for  $0 \leq i < k$  do
    for  $u \in L_i$  do
      for  $(u, v) \in E$  do
        if  $d_v \neq 0$  then
          Cycle  $\leftarrow$  cycle formed by backtracking ancestors until LCA of  $u, v$ 
          return Cycle
        end if
       $L_{i+1} \leftarrow L_{i+1} \cup \{v\}$ 
    end for
  end for
   $S_{i+1} \leftarrow S_i + |L_{i+1}|$ 
  if  $i \geq k - r$  and  $S_i \leq \Delta S_{i-k+r}$  then
    for  $0 \leq j \leq i - k + r$  do
      for  $u \in L_j$  do
        for  $(u, v) \in E$  do
           $E \leftarrow E - \{(u, v)\}$ 
        end for
      end for
    end for
  end if
end for

```

To minimize the runtime, we set $\Delta = n^{r/k}$ and obtain a running time of $\tilde{O}(n^{2-r/k})$. Since $m = \Theta(n^{1+1/k})$,

$$\tilde{O}(n^{2-r/k}) = \tilde{O}\left(\frac{n^{2+r}}{n^{(1+1/k) \cdot r}}\right) = \tilde{O}\left(\frac{n^{2+r}}{m^r}\right).$$

This gives us the desired runtime. ◀

► **Lemma 18.** *The time complexity of Algorithm 1 is $O(n\Delta^{(k-r)/r})$ and it is guaranteed to return a cycle of length $\leq 2k$ if $g \leq 2k - 2r$ and if there is a shortest cycle where all vertices have low r -degree.*

Proof. Firstly, we show that for any low r -degree vertex v , we will find an $\ell \geq k - r$ such that $|T_\ell(v)| \leq \Delta^{(k-r)/r} |T_{\ell-(k-r)}|$ and $\ell < k$. Assume the contrary, that $|T_\ell(v)| > \Delta^{(k-r)/r} |T_{\ell-(k-r)}|$ for all $k - r \leq \ell < k$. Multiplying these inequalities for all ℓ in this range gives

$$\Delta^{k-r} \prod_{j=0}^{r-1} |T_j(v)| < \prod_{i=k-r}^{k-1} |T_i(v)|.$$

For any non-negative integer t , denote \bar{t} to be the unique integer such that $t \equiv \bar{t} \pmod{r}$ and $0 \leq \bar{t} < r$. Since all vertices w in this subgraph have the property that $|T_r(w)| \leq \Delta$, for any non-negative integer i and any vertex v we have

$$|T_{\bar{t}}(v)| \Delta^{(i-\bar{t})/r} \geq |T_i(v)|.$$

Now note that $\{\overline{k-r}, \overline{k-r+1}, \dots, \overline{k-1}\}$ is exactly $\{0, 1, 2, \dots, r-1\}$ as these are r consecutive integers. Thus we can bound the right-hand side of the inequality like so:

$$\begin{aligned}
\Delta^{k-r} \prod_{j=0}^{r-1} |T_j(v)| &< \prod_{i=k-r}^{k-1} |T_i(v)| \\
&\leq \prod_{i=k-r}^{k-1} \Delta^{(i-\bar{i})/r} |T_{\bar{i}}(v)| \\
&\leq \Delta^{\frac{(\sum_{i=k-r}^{k-1} i) - (\sum_{j=0}^{r-1} j)}{r}} \prod_{j=0}^{r-1} |T_j(v)| \\
&\leq \Delta^{\frac{\sum_{j=0}^{r-1} (k-r+j)-j}{r}} \prod_{j=0}^{r-1} |T_j(v)| \\
&= \Delta^{k-r} \prod_{j=0}^{r-1} |T_j(v)|.
\end{aligned}$$

So we reach a contradiction as desired and we conclude that there is some $\ell < k$ and so $\ell - (k-r) < r$. If $T_{\ell-(k-r)}(v)$ contains a vertex in the shortest cycle, we indeed obtain a $+(2r-1)$ approximation. Furthermore, we now no longer need to BFS from any vertex in $T_{\ell-(k-r)}(v)$ so we can discard all of these. So although we had to do $O(|T_\ell(v)|)$ work for BFSing from this v , we were able to discard $|T_{\ell-(k-r)}(v)|$ vertices. Thus, the amortized time complexity per vertex is $O(\Delta^{(k-r)/r})$ and so the total time complexity is $O(n\Delta^{(k-r)/r})$.

Note that we only guarantee returning a cycle of length $\leq 2k$ rather than of length $\leq g + 2r + 1$ as it is possible we find a longer cycle first, in which case we must terminate immediately to guarantee that the BFS-Cycle subroutine step is still linear only in terms of the number of vertices and not edges. This is fine as we now have a shorter cycle and we can repeat the algorithm with smaller k . To make this efficient, we can binary search, incurring only an additional $\log k$ factor. \blacktriangleleft

References

- 1 Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM J. Comput.*, 28(4):1167–1181, 1999.
- 2 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 522–539. SIAM, 2021.
- 3 John A. Bondy and Miklós Simonovits. Cycles of even length in graphs. *Journal of Combinatorial Theory*, pages 16:97–16:105, 1974.
- 4 Karl Bringmann, Fabrizio Grandoni, Barna Saha, and Virginia Vassilevska Williams. Truly subcubic algorithms for language edit distance and RNA folding via fast bounded-difference min-plus product. *SIAM J. Comput.*, 48(2):481–512, 2019.
- 5 Shucheng Chi, Ran Duan, and Tianle Xie. Faster algorithms for bounded-difference min-plus product. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1435–1447, 2022.
- 6 Edith Cohen and Uri Zwick. All-pairs small-stretch paths. *J. Algorithms*, 38(2):335–353, 2001.
- 7 Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM J. Comput.*, 29(5):1740–1759, 2000.

- 8 Yuzhou Gu, Adam Polak, Virginia Vassilevska Williams, and Yinzhan Xu. Faster monotone min-plus product, range mode, and single source replacement paths. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12–16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPICs*, pages 75:1–75:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- 9 Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. *SIAM J. Comput.*, 7(4):413–423, 1978.
- 10 Avi Kadria, Liam Roditty, Aaron Sidford, Virginia Vassilevska Williams, and Uri Zwick. Algorithmic trade-offs for girth approximation in undirected graphs. In *Proc. SODA 2022*, pages 1471–1492, 2022.
- 11 François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the Coppersmith-Winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1029–1046. SIAM, Philadelphia, PA, 2018.
- 12 Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7–10, 2018*, pages 1236–1252. SIAM, 2018.
- 13 Andrzej Lingas and Eva-Marta Lundell. Efficient approximation algorithms for shortest cycles in undirected graphs. *Inf. Process. Lett.*, 109(10):493–498, 2009.
- 14 Seth Pettie. A new approach to all-pairs shortest paths on real-weighted graphs. *Theor. Comput. Sci.*, 312(1):47–74, 2004.
- 15 Liam Roditty and Virginia Vassilevska Williams. Subquadratic time approximation algorithms for the girth. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17–19, 2012*, pages 833–845. SIAM, 2012.
- 16 Raimund Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *J. Comput. Syst. Sci.*, 51(3):400–403, 1995.
- 17 Avi Shoshan and Uri Zwick. All pairs shortest paths in undirected graphs with integer weights. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17–18 October, 1999, New York, NY, USA*, pages 605–615. IEEE Computer Society, 1999.
- 18 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005.
- 19 Virginia Vassilevska Williams. On Some Fine-Grained Questions in Algorithms and Complexity. In *Proceedings of the International Congress of Mathematicians ICM 2018*, volume 3, pages 3447–3487, 2019.
- 20 Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5):27:1–27:38, 2018.
- 21 Virginia Vassilevska Williams and Yinzhan Xu. Truly subcubic min-plus product for less structured matrices, with applications. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5–8, 2020*, pages 12–29. SIAM, 2020.
- 22 R. Ryan Williams. Faster all-pairs shortest paths via circuit complexity. *SIAM J. Comput.*, 47(5):1965–1985, 2018.
- 23 Raphael Yuster and Uri Zwick. Finding even cycles even faster. *SIAM J. Discret. Math.*, 10(2):209–222, 1997.

A Fast $(\min, +)$ Product between Column and Row Bounded-Difference Matrices

In this section, we adapt the method of Chi, Duan and Xie [5] for bounded difference min-plus matrix product for our case, thus proving Lemma 7. The main difference in our method is that we can no longer divide matrices into square blocks since differences are only bounded

50:10 New Additive Approximations for Shortest Paths and Cycles

in one direction. Instead, we divide them into rectangular blocks. Most terminologies and analysis in their work can be adapted to our case, so we only sketch the needed modifications here.

Suppose the matrices to multiply are A, B_0 with size $n \times m$ and $m \times n$. For simplicity, we transpose B_0 to be $B = B_0^T$. Our assumption is that for some constant Δ , $|A_{i,j} - A_{i+1,j}| \leq \Delta$, $|B_{i,j} - B_{i+1,j}| \leq \Delta$ for valid indexes. We want to compute $C_{i,j} = \min_k \{A_{i,k} + B_{j,k}\}$.

We divide the rows of A and B into blocks of size α . The main difference from [5] is that we can no longer divide columns into blocks. For each pair of blocks (one block of columns in A and one block of columns in B), we pick any i and j from each block and compute $C_{i,j}$. This step takes $O(n^2 m / \alpha^2)$ time.

By locality, we know for any (i', j') in these two blocks, $|C_{i',j'} - C_{i,j}| \leq 2\alpha\Delta$ (Since $C_{i',j'}$ equals to $A_{i',k} + B_{j',k}$ for some k , and $|A_{i',k} + B_{j',k} - A_{i,k} - B_{j,k}| \leq 2\alpha\Delta$ for all k 's by the property of bounded-difference within columns). We call k so that $|A_{i,k} + B_{j,k} - C_{i,j}| \leq 4\alpha\Delta$ candidates, and only these k 's could contribute to (i', j') 's in these two blocks.

For block pairs with no more than β candidates, we simply enumerate through these k 's for every (i', j') , taking $O(n^2\beta)$ time.

For block pairs with more than β candidates, we use the method in Section 2, [5]. Sample a set of columns S with size $\Omega(m \log n / \beta)$, then reduce via these columns, mapping resulting segments to γ columns. Computing the bounded min-plus matrix product after mapping would take time $\tilde{O}(M(n, \gamma, n)\alpha m / \beta)$ where $M(n, \gamma, n)$ is the complexity of multiplying $n \times \gamma$ and $\gamma \times n$ matrices. For each of the n^2 / α^2 blocks in C , each of the m^2 column pairs has probability $1/\gamma$ to collide (mapped to the same column), and subtracting each collision takes $O(\alpha^2)$ time. Thus, subtracting the contribution of all collisions would take $O(n^2 m^2 / \gamma)$ time.

The total time complexity is $\tilde{O}(n^2 m / \alpha^2 + n^2 \beta + M(n, \gamma, n)\alpha m / \beta + n^2 m^2 / \gamma)$.

Particularly when $m = n$, let $\alpha = n^{0.094513}$, $\beta = n^{0.810974}$, $\gamma = n^{1.189026}$, $M(n, \gamma, n) = O(n^{2.527435})$ [11], we can get the complexity of $O(n^{2.811})$.