Faster Algorithms for All-Pairs Small Stretch Distances in Weighted Graphs

Telikepalli Kavitha

Indian Institute of Science, Bangalore, India
kavitha@csa.iisc.ernet.in

Abstract. Let $G = (V,E)$ be a weighted undirected graph, with non-negative edge weights. We consider the problem of efficiently computing approximate distances between all pairs of vertices in G. While many efficient algorithms are known for this problem in unweighted graphs, not many results are known for this problem in weighted graphs. Zwick [14] showed that for any fixed $\varepsilon > 0$, stretch[1] $1+\varepsilon$ distances between all pairs of vertices in a weighted directed graph on n vertices can be computed in $\tilde{O}(n^{\omega})$ time, where $\omega < 2.376$ is the exponent of matrix multiplication and n is the number of vertices. It is known that finding distances of stretch less than 2 between all pairs of vertices in G is at least as hard as Boolean matrix multiplication of two $n \times n$ matrices. It is also known that all-pairs stretch 3 distances can be computed in $\tilde{O}(n^2)$ time and all-pairs stretch $7/3$ distances can be computed in $\tilde{O}(n^{7/3})$ time. Here we consider efficient algorithms for the problem of computing all-pairs stretch $(2+\varepsilon)$ distances in G, for any $0 < \varepsilon < 1$.

We show that all pairs stretch $(2+\varepsilon)$ distances for any fixed $\varepsilon > 0$ in G can be computed in expected time $O(n^{9/4}\log n)$. This algorithm uses a fast rectangular matrix multiplication subroutine. We also present a combinatorial algorithm (that is, it does not use fast matrix multiplication) with expected running time $O(n^{9/4})$ for computing all-pairs stretch $5/2$ distances in G.

1 Introduction

The all-pairs shortest paths (APSP) problem is one of the most fundamental algorithmic graph problems. Efficient algorithms for the APSP problem are very important in several applications. The complexity of the fastest known algorithm for the APSP problem in a graph with m edges, n vertices and real non-negative edge weights is $O(mn + n^2 \log\log n)$ [12]. Thus this algorithm has a running time of $\Theta(n^3)$ when $m = \Theta(n^2)$. In fact, the best known upper bound on the worst case time complexity of this problem (in terms of n) is $O(n^3/\log n)$ [5], which is marginally subcubic. An almost cubic running time is inefficient for several applications, and this has motivated faster algorithms to compute *approximate* solutions for the APSP problem.

Let $G = (V,E)$ be an undirected graph with non-negative edge weights. A path in G between $u, v \in V$ is said to be of stretch t if its length is at most $t \cdot \delta(u,v)$ where $\delta(u,v)$ is the distance between u and v in G. In this paper we are interested in computing small stretch paths/distances between all pairs of vertices. Zwick [14] showed that for any $\varepsilon > 0$, stretch $1+\varepsilon$ distances between all pairs of vertices in a weighted *directed* graph on n vertices can be computed in time $\tilde{O}(n^{\omega}/\varepsilon \cdot \log(W/\varepsilon))$, where $\omega < 2.376$ is the exponent of matrix multiplication and W is the largest edge weight in the graph, after the edge weights are scaled so that the smallest non-zero edge weight in the graph is 1. It is also known that finding paths of stretch *less* than 2 between all pairs of vertices in an undirected graph on n vertices is at least as hard as Boolean matrix multiplication of two $n \times n$ matrices. Given an undirected weighted graph on n vertices, computing all-pairs stretch 3 distances in $\tilde{O}(n^2)$ time and all-pairs stretch $7/3$ distances in $\tilde{O}(n^{7/3})$ time is known [7] (these algorithms use only combinatorial techniques, i.e., fast matrix multiplication subroutines are not used). Researchers have been trying to explore the possible trade-off between stretch and running time for the problem of computing all-pairs t-stretch distances for $t \in [2,3)$.

[1] A path in G between $u, v \in V$ is said to be of stretch t if its length is at most t times the distance between u and v in G.

1.1 Our Main Results

In this paper we consider faster algorithms for the problem of computing all-pairs stretch t distances for $2 < t < 3$ in a weighted undirected graph G on n vertices. We first present a combinatorial algorithm $\mathsf{STRETCH}_{5/2}$ and show the following result. (For any pair of vertices u, v in G, let $\delta(u,v)$ denote the distance between u and v in G.)

Theorem 1. *Algorithm* $\mathsf{STRETCH}_{5/2}(G)$ *runs in expected time* $O(n^{9/4})$*, where* n *is the number of vertices in the input graph* G *and constructs an* $n \times n$ *table* d *such that:* $\delta(u,v) \leq d[u,v] \leq 5/2 \cdot \delta(u,v)$.

We then augment our combinatorial algorithm $\mathsf{STRETCH}_{5/2}(G)$ with a fast rectangular matrix multiplication subroutine and present the algorithm $\mathsf{STRETCH}_{2+\varepsilon}(G)$ and prove the following result.

Theorem 2. *Given any* $\varepsilon > 0$*, algorithm* $\mathsf{STRETCH}_{2+\varepsilon}(G)$ *constructs an* $n \times n$ *table* d *such that* $\delta(u,v) \leq d[u,v] \leq (2+\varepsilon)\delta(u,v)$ *in expected time* $O(n^{9/4}\log n) + \tilde{O}(n^{2.248}(\log^2 W)/\varepsilon^2)$*, where* n *is the number of vertices in the input graph* G *and* W *is the largest edge weight after scaling the edge weights so that the smallest non-zero edge weight is 1.*

Thus when all edge weights in G are polynomial in n and $\varepsilon > 0$ is a constant, $\mathsf{STRETCH}_{2+\varepsilon}(G)$ computes all-pairs stretch $2+\varepsilon$ distances in expected time $O(n^{9/4}\log n)$ since $\tilde{O}(n^{2.248})$ is $o(n^{9/4})$.

Motivation. During the last 10-15 years, many new combinatorial algorithms [2, 6, 1, 8, 7, 13, 3, 9] were designed for the all-pairs approximate shortest paths problem in order to achieve faster running times in weighted and unweighted graphs. In weighted graphs, the current fastest randomized combinatorial algorithms (from [4]) for computing all-pairs stretch t distances for $t < 3$ in G with m edges and n vertices are: computing all-pairs stretch 2 distances in expected $\tilde{O}(m\sqrt{n} + n^2)$ time and computing all-pairs stretch $7/3$ distances in expected $\tilde{O}(m^{2/3}n + n^2)$ time. These algorithms are improvements of the following deterministic algorithms: an $\tilde{O}(n^{3/2}m^{1/2})$ algorithm for stretch 2 distances and an $\tilde{O}(n^{7/3})$ algorithm for stretch $7/3$ distances by Cohen and Zwick [7]. However when $m = \Theta(n^2)$, there is no improvement in the running time and stretch 2 distances are computed in $\tilde{\Theta}(n^{5/2})$ time and stretch $7/3$ distances are computed in $\tilde{\Theta}(n^{7/3})$ time. There is an algorithm [4] with expected running time $\tilde{O}(n^2)$ for computing approximate (u,v) distances for all pairs (u,v), where the distance returned is at most $2\delta(u,v)+$ maximum weight of an edge on a u-v shortest path. However, note that we cannot claim that the *stretch* here is at most $3-\varepsilon$ for any fixed $\varepsilon > 0$. Thus there was no $o(n^{7/3})$ algorithm known for computing all-pairs stretch $(3-\varepsilon)$ distances for any constant $\varepsilon > 0$. We try to fill this gap in this paper.

Our techniques. Our algorithms construct a decreasing sequence of sets: $V = S_0 \supseteq S_1 \supseteq S_2 \supseteq S_3$. Vertices in each S_i perform Dijkstra in a specific subgraph G_{i+1} of G, where the density of G_{i+1} is inversely proportional to the cardinality of S_i. Then these sets S_i *cooperate* with each other. For instance, vertices in S_1 and S_2 exchange information about the distances they have computed. Then each $v \in S_i$ performs Dijkstra again in the subgraph G_{i+1} augmented with new "edges", corresponding to the new knowledge of distances that v has acquired. We show that now each pair (u,v) has enough data to compute a stretch $5/2$ estimate of its distance $\delta(u,v)$.

The step where each vertex in the set S_i performs Dijkstra twice in a subgraph G_{i+1} bears a lot of similarity with schemes in [7] for computing all-pairs small stretch distances. The new idea here is the *cooperation* between the sets S_i - this cooperation forms a crucial step of our algorithm and that is what ensures a small stretch. In our analysis of algorithm $\mathsf{STRETCH}_{5/2}(G)$ we actually get a bound of $7/3$ on the stretch in all cases, except one where we get a stretch of $5/2$.

Improved stretch. The stretch in this algorithm can be improved to $2+\varepsilon$ by using a subroutine for witnessing a Boolean product matrix. This subroutine for witnessing a Boolean product matrix is implemented using fast rectangular matrix multiplication. If we assume that all edge weights are $O(n^c)$ for some fixed c and $\varepsilon > 0$ is any constant, then the expected running time of this step is also $O(n^{9/4})$. But we have to be more careful in constructing the set S_1 in this algorithm and so the overall cost of the algorithm becomes $O(n^{9/4}\log n)$.

Note that these algorithms can be easily modified to also report the corresponding approximate shortest path, and the time for doing so, is proportional to the number of edges in the approximate shortest path.

Related results. An active area of research in algorithms that report all-pairs small stretch distances is in designing compact data structures, to answer distance queries. Instead of storing an $n \times n$ look-up table, these algorithms use $o(n^2)$ space. More specifically, for any integer $k \geq 1$, the data structure uses $O(kn^{1+1/k})$ space and it answers any distance query with stretch $2k-1$, in $O(k)$ time [13]. It was shown in [13] that any such data structure with stretch $t < 3$ must use $\Theta(n^2)$ space on at least one input graph. Hence, in algorithms that compute all-pairs stretch $3-\varepsilon$ distances for $\varepsilon > 0$, what one seeks to optimize is the running time of the algorithm, since the space requirement is $\Theta(n^2)$.

For unweighted graphs, there are many results known for computing all-pairs small stretch paths or distances. More results, in particular, results with *additive* error are known when the graph is unweighted. Aingworth *et al.* [1] showed a simple and elegant $\tilde{O}(n^{5/2})$ algorithm for finding all distances in unweighted undirected graphs with an additive one-sided error of at most 2. Dor *et al.* [8] improved and extended this algorithm and they find all-pairs distances in unweighted undirected graphs with an additive one-sided error of at most $2(k-1)$ in $O(kn^{2-\frac{1}{k}}m^{\frac{1}{k}}\operatorname{polylog} n)$ time. The algorithms in [3, 9] compute approximate distances in unweighted graphs with both multiplicative as well as additive errors simultaneously. Note that the techniques used in unweighted graphs do not typically generalize to weighted graphs.

Organization of the paper. The preliminaries are given in Section 2. In Section 3 we present our algorithm $\mathsf{STRETCH}_{5/2}(G)$ and prove the bound on its stretch. In Section 4 we present our algorithm $\mathsf{STRETCH}_{2+\varepsilon}(G)$ and prove its correctness.

2 Preliminaries

We have the following sampling scheme: $V = S_0 \supseteq S_1 \supseteq S_2 \supseteq S_3 \supseteq S_4 = \emptyset$
where each S_i, $i = 1,2,3$ is obtained by sampling vertices in S_{i-1} with probability $n^{-1/4}$. Note that the expected size of S_1 is $n^{3/4}$, of S_2 is $\sqrt{n}$, and of S_3 is $n^{1/4}$. For each vertex $u \in V$ and for $i = 1,2,3$, define $\delta(u,S_i)$ as the distance between u and the vertex in S_i that is nearest to u. Let $s_i(u) \in S_i$ be the vertex in S_i that is nearest to u. That is, $\delta(u,S_i) = \delta(u,s_i(u)) \leq \delta(u,x)$ for all $x \in S_i$. In case there is more than one vertex in S_i with distance $\delta(u,S_i)$ to u, then break the tie arbitrarily to define $s_i(u)$. Note that since $S_4 = \emptyset$, we define $\delta(u,S_4) = \infty$.

Now we need to define certain neighborhoods around a vertex u.

Definition 1 (from [13]). *For any vertex u and for $i = 1,2,3$, define $ball_i(u)$ as:*

$$ball_i(u) = \{v \in V : \delta(u,v) < \delta(u,S_i)\}.$$

That is, $ball_i(u)$ is the set of all vertices v that are strictly closer to u than the nearest vertex in S_i is to u.

The graphs of interest to us in our algorithms are the graphs $G_i = (V,E_i)$ for $i = 1,2,3$, where

$$E_i = \{(u,v) \in E : v \in ball_i(u)\}.$$

Note that G_i, for $i = 1,2,3$, are undirected graphs. Each G_i is a subgraph of G, where each vertex $x \in V$ keeps edges to only those of its neighbors that lie in $ball_i(x)$. Note that constructing these graphs G_i is easy. In G, connect a dummy vertex s^* to all the vertices of the set S_i and assign weight zero to all these edges. Now run Dijkstra's shortest paths algorithm with source s^* in G. The distance returned between s^* and u is the distance $\delta(u,S_i)$, for any $u \in V$. The vertex $s_i(u)$ is the successor of s^* in the shortest s^*-u path in this graph. To form the edge set E_i of G_i, each u looks at its adjacency list and retains only those neighbors v where $\delta(u,v) < \delta(u,S_i)$. We have $E_1 \subseteq E_2 \subseteq E_3 \subseteq E = E_4$.

The following claims, which are simple to show, are stated in the form of Proposition 1 and Proposition 2. They will be used repeatedly in the paper and their proofs are given in the Appendix.

Proposition 1. *For $S_i \subseteq V$, ($i \in \{1,2,3\}$) the following assertions are true.*

1. *For any two vertices $u,v \in V$, if $v \in ball_i(u)$, then the subgraph $G_i = (V,E_i)$ preserves the exact distance between u and v.*
2. *The subgraph $G_i = (V,E_i \cup E(s_i(u)))$ preserves the exact distance between u and $s_i(u)$, where $E(s_i(u))$ is the set of edges incident on $s_i(u)$.*

Proposition 2. *If the set $S_i \subseteq V$ is formed by selecting each vertex independently with probability q, then the expected size of the set E_i is $O(n/q)$.*

We now define the set $bunch_i(u)$. For any vertex $u \in V$ and $i = 1,2,3$, the set $bunch_i(u) \subseteq S_i$ is defined as follows: $bunch_i(u) = \{x \in S_i |\ \delta(u,x) < \delta(u,S_{i+1})\} \cup \{s_i(u)\}$. That is, $bunch_3(u) = S_3$ since $\delta(u,S_4) = \infty$, while $bunch_2(u)$ consists of $s_2(u)$ and all the vertices in S_2 that belong to $ball_3(u)$ and $bunch_1(u)$ consists of $s_1(u)$ and all the vertices in S_1 that belong to $ball_2(u)$. The following result about the expected size of $bunch_i(u)$ and the complexity of computing the set $bunch_i(u)$ was shown in [13].

Lemma 1. *[13] Given a graph $G = (V,E)$, let the set S_{i+1} be formed by picking each vertex of a set $S_i \subseteq V$ independently with probability q. Then*
(i) the expected number of $x \in S_i$ such that $\delta(u,x) < \delta(u,S_{i+1})$ is at most $1/q$ for each u, and
(ii) the expected time to compute the sets $bunch_i(u)$, summed over all $u \in V$, is $O(m/q)$.

Another concept that we use is the notion of *overlap* of $ball_i(u)$ and $ball_i(v)$. We define this term below and Fig. 1 illustrates this.

Definition 2. *Let $u,v \in V$. For any $i = 1,2,3$, we say that $ball_i(u)$ and $ball_i(v)$ overlap if*

$$\delta(u,S_i) + \delta(v,S_i) > \delta(u,v).$$

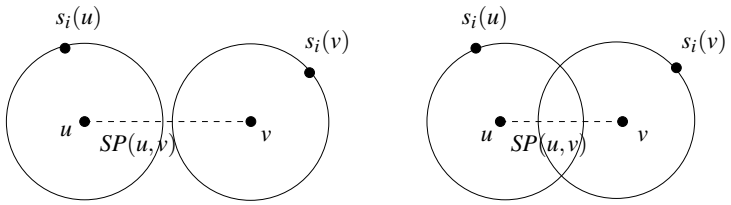

Fig. 1. In the figure on the left, $ball_i(u)$ and $ball_i(v)$ do not overlap; whereas on the right, they overlap.

Remark 1. The set S_1 can also be obtained deterministically in $O(m+n)$ time using algorithms from [1, 8]; then S_1 would dominate all vertices of degree at least $n^{1/4}$ and $|S_1|$ would be $O(n^{3/4}\log n)$. The edge set $|E_1|$ will be the set of the lightest $n^{1/4}$ edges touching each vertex (see [1, 7, 8] for the details).

Then the set S_2 will be obtained by sampling vertices in S_1 with probability $n^{-1/4}$ and the set S_3 will be obtained by sampling vertices in S_2 with probability $n^{-1/4}$. We will use this construction of $V \supseteq S_1 \supseteq S_2 \supseteq S_3$ in Section 4, since there we need $|S_1|$ to be $O(n^{3/4}\log n)$ always.

3 All-pairs stretch $5/2$ distances

Let $G = (V,E)$ be an undirected graph with a weight function $w : E \to \mathbb{Q}^+$. Our algorithm for computing small stretch distances runs Steps 1-5 given below for 2 iterations and as the algorithm evolves, distance estimates computed till then will be stored in an $n \times n$ table d. The table d is initialized as: $d[u,u] = 0$ and $d[u,v] = w(u,v)$ for all $(u,v) \in E$. Otherwise $d[u,v] = \infty$.

A basic step that we use in our algorithm is the following: a vertex v performs Dijkstra in a subgraph G' that is augmented with all *pairs* (v,x). That is, (v,x) need not be an edge, however pairs (v,x) with weight $d[v,x]$ for all $x \in V$ are added to the edge set of G', so that the source vertex v can use the distance estimates that it has acquired already, in order to find better paths to other vertices.

We first construct the sets $V \supseteq S_1 \supseteq S_2 \supseteq S_3$ using our sampling scheme, and build the graphs $G_i = (V,E_i)$, where $E_i = \cup_u ball_i(u)$ and also construct the sets $bunch_i(u)$, for $i = 1,2,3$ (see Section 2 for more details).

The Algorithm $\mathsf{STRETCH}_{5/2}(G)$

– Initialize the table d as described above.
– Each vertex $v \in S_3$ performs Dijkstra in the entire graph G and the table d gets updated accordingly.
** Run Steps 1-5 for 2 iterations and return the table d.

1. Each vertex u runs Dijkstra's single source shortest paths algorithm in the graph $G_1 = (V,E_1)$ that is augmented with pairs (u,x) for all $x \in V$ with weight $d[u,x]$.
   (Dijkstra's algorithm will update the entries in the row corresponding to u in the table d.)
   – Each u now updates entries corresponding to the rows of all vertices s, in the table d, where $s \in bunch_1(u) \cup bunch_2(u)$.
   (That is, if for some $y \in V$ we have $d[s,u] + d[u,y] < d[s,y]$ where $s \in bunch_1(u) \cup bunch_2(u)$, then we set $d[s,y] = d[s,u] + d[s,y]$.)
2. Each vertex $s_1 \in S_1$ runs Dijkstra's algorithm in the graph $G_2 = (V,E_2)$ that is augmented with all pairs (s_1,x) with weight $d[s_1,x]$.
   – Each $s_1 \in S_1$ updates entries corresponding to the rows of all vertices in S_2 in the table d.
3. Each vertex $s_2 \in S_2$ runs Dijkstra's algorithm in the graph $G_3 = (V,E_3)$ augmented with all pairs (s_2,x) with weight $d[s_2,x]$.
   – Each $s_2 \in S_2$ updates entries corresponding to the rows of all vertices in S_1 in the table d.
4. For every (u,v) store in $d[u,v]$ the minimum of $d[u,v], d[u,s] + d[s,v]$,
   where $s \in \cup_{i=1}^{3} bunch_i(u)$.
5. Make the table d symmetric: that is, store in $d[u,v]$ the minimum of $d[u,v]$ and $d[v,u]$.

Running Time Analysis. The expected size of S_i is $n^{1-i/4}$ for $i = 1,2,3$ and the expected size of E_i, the set of edges in G_i, is $O(n^{1+i/4})$ (by Lemma 2). The expected time taken for all vertices in S_i to perform Dijkstra in the graph G_{i+1} is

$$\sum_{v \in V} (n - 1 + \mathbf{E}[X_v |\ v \in S_i]) \cdot \mathbf{Pr}[v \in S_i] \tag{1}$$

where X_v is the random variable denoting the number of all edges present in the set E_{i+1} excluding those incident on v. It is easy to show that

$$\mathbf{E}[X_v |\ v \in S_i] \leq \mathbf{E}[Y |\ v \in S_i] = \mathbf{E}[Y] \leq n^{1+(i+1)/4} \tag{2}$$

where Y is the random variable denoting the number of edges in the graph $G_{i+1} - \{v\}$. Note that Y is independent of whether or not $v \in S_i$ because v is not present in $G_{i+1} - \{v\}$. Combining Equations (2) and (1), the expected time taken for all vertices in S_i to perform Dijkstra in the graph G_{i+1} is bounded by

$$\sum_{v \in V} \left(n - 1 + n^{1+(i+1)/4}\right) \cdot \mathbf{Pr}[v \in S_i] \leq 2n^{9/4}$$

since $\mathbf{Pr}[v \in S_i] = n^{1-i/4}$.

For each $i \in \{1,2,3\}$ the expected size of $bunch_i(u)$ for any $u \in V$ is $O(n^{1/4})$ (by Lemma 1(i)) and the time to compute all the sets $bunch_i(u)$ is $O(mn^{1/4})$ (by Lemma 1(ii)). Thus we have shown the following lemma.

Lemma 2. *The expected running time of the algorithm* $\mathsf{STRETCH}_{5/2}(G)$ *is* $O(n^{9/4})$.

3.1 Correctness of the algorithm $\mathsf{STRETCH}_{5/2}(G)$

Lemma 3. *For each pair* $(u,v) \in V \times V$*, we have:* $\delta(u,v) \leq d[u,v] \leq 5/2 \cdot \delta(u,v)$*, where* d *is the table returned by the algorithm* $\mathsf{STRETCH}_{5/2}(G)$ *and* $\delta(u,v)$ *is the distance between* u *and* v *in* G*.*

Proof. For every u,v, since $d[u,v]$ is the length of some path in G between u and v, we always have $\delta(u,v) \leq d[u,v]$. The hard part of the lemma is showing the upper bound on $d[u,v]$. For any pair of vertices u and v, let $SP(u,v)$ denote the shortest path between u and v in G. Let us first show the following claim.

Claim 1 *For any* $i \in \{1,2,3\}$*, if all the edges in* $SP(u,v)$ *are present in* $G_{i+1} = (V, E_{i+1})$ *and* $ball_i(u)$ *and* $ball_i(v)$ *do not overlap, then* $d[u,v] \leq 2\delta(u,v)$*.*

Proof. It is given that all the edges in $SP(u,v)$ are present in E_{i+1}. So all the edges in the path[2] $s_i(u) \rightsquigarrow u \rightsquigarrow v$ obtained by concatenating $SP(s_i(u),u)$ and $SP(u,v)$ are present in $E_{i+1} \cup E(s_i(u))$ (by Proposition 1), where $E(s_i(u))$ is the set of edges incident on $s_i(u)$. Similarly, all the edges in the path $s_i(v) \rightsquigarrow v \rightsquigarrow u$ are present in $E_{i+1} \cup E(s_i(v))$. Since every vertex $x \in S_i$ performs Dijkstra in the graph G_{i+1} augmented with $E(x)$, we have $d[s_i(u),v] \leq \delta(s_i(u),u) + \delta(u,v)$ and $d[s_i(v),u] \leq \delta(s_i(v),v) + \delta(u,v)$. Also, because $ball_i(u)$ and $ball_i(v)$ do not overlap, we have $\delta(s_i(u),u) + \delta(s_i(v),v) \leq \delta(u,v)$. Combining these inequalities, we have

$$\min\{\delta(u,s_i(u)) + d[s_i(u),v],\ \delta(v,s_i(v)) + d[s_i(v),u]\} \leq 2\delta(u,v).$$

Step 4 in our algorithm ensures that: $d[u,v] \leq \min\{\delta(u,s_i(u)) + d[s_i(u),v], \delta(v,s_i(v)) + d[s_i(v),u]\}$. Thus $d[u,v] \leq 2\delta(u,v)$. □

[2] Note that we use the symbols $x \rightsquigarrow y$ and $x \rightarrow y$ for illustrative purposes, the paths and edges here are *undirected*.

Claim 1 leads to the following corollary since $E_4 = E$, the edge set of G, and E obviously contains all the edges in $SP(u,v)$.

Corollary 1 *If $ball_3(u)$ and $ball_3(v)$ do not overlap, then $d[u,v] \leq 2\delta(u,v)$.*

Now let us consider the case when $ball_1(u)$ and $ball_1(v)$ overlap.

Claim 2 *If $ball_1(u)$ and $ball_1(v)$ overlap, then $d[u,v] = \delta(u,v)$.*

Proof. We are given that $ball_1(u)$ and $ball_1(v)$ overlap. So $\delta(u,v) < \delta(u,s_1(u)) + \delta(v,s_1(v))$ and we can partition the shortest path between u and v as: $SP(u,v) = u \rightsquigarrow a \rightarrow b \rightsquigarrow v$, where all the vertices in $u \rightsquigarrow a$ belong to $ball_1(u)$ and all the vertices in $b \rightsquigarrow v$ belong to $ball_1(v)$. Since the graph G_1 has the edge set $\cup_x ball_1(x)$, the only edge in $SP(u,v)$ that might possibly be missing in the graph G_1 is the edge (a,b) (refer Fig. 2). In the first iteration of the ** loop, in Step 1 (refer Algorithm $\mathsf{STRETCH}_{5/2}(G)$), the vertex b would

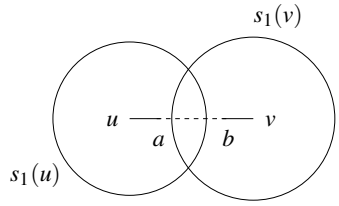

Fig. 2. $ball_1(u)$ and $ball_1(v)$ overlap.

perform Dijkstra in G_1 augmented with the edge (a,b). Since the path $a \rightsquigarrow u$ is present in G_1, in this step, the vertex b would learn of its distance to u, i.e., $d[b,u] = \delta(u,b)$. Since the table d is made symmetric in Step 5, $d[u,b] = \delta(u,b)$ at the end of the first iteration of the ** loop.

In the second iteration of the ** loop, u would augment the "edge" (u,b) with weight $d[u,b] = \delta(u,b)$ to G_1 and since all the edges in $b \rightsquigarrow v$ are present in G_1, we have the path $u \rightarrow b \rightsquigarrow v$ in the augmented G_1. Thus u determines $d[u,v] = \delta(u,v)$. This proves the statement of Claim 2. □

We shall assume henceforth that $ball_3(u)$ and $ball_3(v)$ overlap and $ball_1(u)$ and $ball_1(v)$ do not overlap (refer Corollary 1 and Claim 2). That leaves us with two further cases, as to whether $ball_2(u)$ and $ball_2(v)$ overlap or not. We shall call them Case 1 and Case 2.

Case 1: $ball_2(u)$ and $ball_2(v)$ do not overlap.

If all the edges in $SP(u,v)$ are present in $G_3 = (V,E_3)$, then it follows from Claim 1 that $d[u,v] \leq 2\delta(u,v)$. So let us assume that some of the edges of $SP(u,v)$ are *not* present in $G_3 = (V,E_3)$.

The graph G_3 has the edge set $E_3 = \cup_x ball_3(x)$. Since $ball_3(u)$ and $ball_3(v)$ overlap, the only way that some of the edges in $SP(u,v)$ are not present in E_3 is that exactly one edge in $SP(u,v)$ is missing from E_3. This edge is between the last vertex a (from the side of u) in $SP(u,v)$ that is in $ball_3(u)$ and the first vertex b in $SP(u,v)$ that is in $ball_3(v)$ (refer Fig. 3). Every other vertex and its successor in $SP(u,v)$ would either both be in $ball_3(u)$ or both be in $ball_3(v)$ and such edges have to be present in G_3.

By Step 4 we know that $d[u,v]$ is at most the minimum of $\delta(u,s_3) + \delta(s_3,v)$ distances, where $s_3 \in S_3$. Hence we have the following bound on $d[u,v]$.

$$d[u,v] \leq \delta(u,s_3(a)) + \delta(s_3(a),v) \tag{3}$$
$$\leq \delta(u,v) + 2\delta(a,s_3(a)) \tag{4}$$
$$\leq \delta(u,v) + 2w(a,b). \tag{5}$$

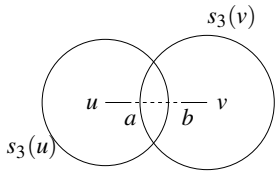

Fig. 3. $ball_3(u)$ and $ball_3(v)$ overlap but the edge (a,b) is not present in G_3.

Inequality (5) follows from Inequality (4) because the edge (a,b) is missing from $ball_3(a)$. Next, we shall show that we also have the following inequalities.

$$d[u,v] \leq \delta(u,v) + 2\delta(u,a) + 2\delta(u,s_2(u)) \text{ and} \tag{6}$$

$$d[u,v] \leq \delta(u,v) + 2\delta(v,b) + 2\delta(v,s_2(v)). \tag{7}$$

Adding Inequalities (5), (6), and (7), we get the following inequality:

$$\begin{aligned} 3d[u,v] &\leq 5\delta(u,v) + 2\delta(u,s_2(u)) + 2\delta(v,s_2(v)) \\ &\leq 7\delta(u,v) \end{aligned}$$

since $\delta(u,s_2(u)) + \delta(v,s_2(v)) \leq \delta(u,v)$ because $ball_2(u)$ and $ball_2(v)$ do not overlap (by the definition of Case 1). Thus we have $d[u,v] \leq 7/3 \cdot \delta(u,v)$.
So all that is left here is to prove Inequalities (6) and (7). If $s_2(u) \notin bunch_2(a)$, then we have $\delta(a,s_3(a)) \leq \delta(a,s_2(u)) \leq \delta(a,u) + \delta(u,s_2(u))$. Substituting this in Inequality (4), we get Inequality (6).
If $s_2(u) \in bunch_2(a)$, then in the second iteration of the ** loop, in Step 1, the vertex a updates the entry $d[s_2(u),b]$ to at most $d[s_2(u),a] + w(a,b)$ since $s_2(u) \in bunch_2(a)$. We already have $d[s_2(u),a] \leq \delta(s_2(u),u) + \delta(u,a)$ since the path $s_2(u) \rightsquigarrow u \rightsquigarrow a$ is in the augmented G_3. Thus after Step 1 in the second iteration of the ** loop, we have $d[s_2(u),b] \leq \delta(s_2(u),a) + w(a,b)$. In Step 3, $s_2(u)$ performs Dijkstra in G_3 augmented with the "edge" $(s_2(u),b)$ with weight at most $d[s_2(u),b]$. Since all the edges of $SP(b,v)$ are in G_3, we have $d[s_2(u),v] \leq \delta(s_2(u),u) + \delta(u,v)$. Since $d[u,v] \leq \delta(u,s_2(u)) + d[s_2(u),v]$, we get $d[u,v] \leq 2\delta(u,s_2(u)) + \delta(u,v)$. This implies Inequality (6). The proof of Inequality (7) is analogous to the proof of Inequality (6). This finishes Case 1.

Case 2: $ball_2(u)$ and $ball_2(v)$ overlap.
This case is further split into 2 cases: CASE(I), where all the edges in $SP(u,v)$ are present in $G_3 = (V,E_3)$ but not in $G_2 = (V,E_2)$ and CASE (II), where some of the edges in $SP(u,v)$ are not present in $G_3 = (V,E_3)$. We present the entire proof of Case 2 in the Appendix.

This finishes the proof of Lemma 3. □

Lemma 3 and Lemma 2 yield Theorem 1, stated in Section 1. Note that in the proof of Lemma 3, we show a stretch of at most $7/3$ in all cases, except in CASE(II) of Case 2, where we show a stretch of $5/2$.

4 All-pairs stretch $(2+\varepsilon)$ distances

Let $\varepsilon > 0$ be any given parameter. In this section we present our algorithm $\mathsf{STRETCH}_{2+\varepsilon}(G)$ which takes as input an undirected graph $G = (V,E)$ with a weight function $w : E \to \mathbb{Q}^+$ and computes an $n \times n$ table d that stores all-pairs stretch $(2+\varepsilon)$ distances. In algorithm $\mathsf{STRETCH}_{2+\varepsilon}(G)$ we augment the algorithm

$\mathsf{STRETCH}_{5/2}$ of the previous section with some more computation so that in the new algorithm we get a stretch of at most $2+\varepsilon$. Note that we can assume that all edge weights are positive[3]. Let us now scale the edge weights, if necessary, so that the smallest edge weight is 1 and let W be the largest edge weight.

The Algorithm $\mathsf{STRETCH}_{2+\varepsilon}(G)$

1. Call algorithm $\mathsf{STRETCH}_{5/2}(G)$. An $n \times n$ table d is returned.
2. Build the sequence of matrices $M_1, M_2, \ldots, M_k$, where $k = \lceil \log_{1+\varepsilon/2}(5/2 \cdot nW) \rceil$. Each M_i is a 0-1 matrix of dimension $n \times |S_1|$ which is defined as: for each $u \in V$ and $x \in S_1$

$$M_i[u,x] = 1 \text{ iff } (1+\varepsilon/2)^{i-1} \leq d[u,x] \leq (1+\varepsilon/2)^i.$$

   The value $d[u,x]$ is looked-up from the table d returned by the $\mathsf{STRETCH}_{5/2}(G)$ algorithm in Step 1.
3. For each $(i,j) \in \{1,\ldots,k\} \times \{1,\ldots,k\}$ do:
   - compute the $n \times n$ "Boolean product witness matrix" W_{ij} corresponding to the Boolean product matrix $M_i M_j^T$. That is, for each $(u,v) \in V \times V$:

$$W_{ij}[u,v] = \begin{cases} s & \text{for some } s \text{ such that } M_i[u,s] = 1 \text{ and } M_j[s,v] = 1 \\ 0 & \text{if there is no such } s. \end{cases}$$

     That is, if $M_i M_j^T[u,v] = 1$, then the entry $W_{ij}[u,v] = s$ is a *witness* for $M_i M_j^T[u,v]$ being 1.
4. For each pair $(u,v) \in V \times V$ do:
   – for each $(i,j) \in \{1,\ldots,k\} \times \{1,\ldots,k\}$ do:
     If $W_{ij}(u,v) \neq 0$ (call it x) and $d[u,x] + d[x,v] < d[u,v]$ then set $d[u,v] = d[u,x] + d[x,v]$.
5. Return the table d.

The algorithm for computing the matrix W_{ij} (from [11]) is given in the Appendix. It can be shown (see [11] for the details) that this algorithm for computing W_{ij} has expected running time $\tilde{O}(C(n))$, where $C(n)$ is the time taken to multiply an $n \times n^\beta$ matrix with an $n^\beta \times n$ matrix.

4.1 The running time of algorithm $\mathsf{STRETCH}_{2+\varepsilon}(G)$

In the algorithm $\mathsf{STRETCH}_{2+\varepsilon}(G)$ the step whose time complexity is the most difficult to analyze is Step 3. We know that the complexity of computing the witness matrix W_{ij} for each pair (i,j) is $\tilde{O}(C(n))$, where $C(n)$ is the time taken to multiply an $n \times n^\beta$ matrix with an $n^\beta \times n$ matrix. Here we will use the following result.

Proposition 3 (Huang and Pan ([10] Section 8.2)). *Multiplying an $n \times n^\beta$ matrix with an $n^\beta \times n$ matrix for $0.294 \leq \beta \leq 1$ takes time $O(n^\alpha)$, where $\alpha = \frac{2(1-\beta)+(\beta-0.294)\omega}{0.706}$, and $\omega < 2.376$ is the best exponent of multiplying two $n \times n$ matrices.*

We want $|S_1|$ to be always around $O(n^\beta)$ for some small β now. Then we can use the above proposition to bound the running time of Step 3. Hence we will use the deterministic construction of S_1 given in Remark 1 (in Section 2) which guarantees that $|S_1|$ is $O(n^{3/4} \log n)$, which is at most $n^{0.76}$ for sufficiently large n.

[3] If there are edges with weight zero, then contract each such edge - this will reduce the number of vertices and it is simple to see that we can easily extend the all-pairs small stretch distances table for the reduced graph to the all-pairs small stretch distances table for the entire graph.

Substituting $\beta = 0.76$ in Proposition 3 yields $C(n)$ is $O(n^{2.248})$. So computing W_{ij} takes expected $\tilde{O}(n^{2.248})$ time. The entire running time of Step 3 is $\tilde{O}(k^2 \cdot n^{2.248})$, where $k = O(\log nW/\varepsilon)$, W is the largest edge weight. It is reasonable to assume that all edge weights are polynomial in n, since we always assumed that arithmetic on these values takes unit time. Then the running time of this step then is $\tilde{O}(n^{2.248}/\varepsilon^2)$, which is $O(n^{9/4})$ if ε is a constant. The call to $\mathsf{STRETCH}_{5/2}(G)$ takes expected $O(n^{9/4}\log n)$ time now since $|S_i|$ is $O(n^{1-i/4}\log n)$, for $i = 1,2,3$. Thus the expected running time of the algorithm $\mathsf{STRETCH}_{2+\varepsilon}(G)$ is $\tilde{O}(n^{2.2.48}\log W/\varepsilon^2) + O(n^{9/4}\log n)$ which is $O(n^{9/4}\log n)$ when edge weights are polynomial in n and $\varepsilon > 0$ is a constant.

4.2 Correctness of the algorithm $\mathsf{STRETCH}_{2+\varepsilon}(G)$

The following lemma shows the correctness of our algorithm.

Lemma 4. *For every $u, v \in V$, the estimate $d[u,v]$ computed by the algorithm $\mathsf{STRETCH}_{2+\varepsilon}(G)$ satisfies: $\delta(u,v) \leq d[u,v] \leq (2+\varepsilon)\delta(u,v)$.*

Proof. Since $d[u,v]$ is always the length of some path in G between u and v, we have $\delta(u,v) \leq d[u,v]$. Now we show the harder part, that is, the upper bound claimed on $d[u,v]$. Recall from Claim 2 that if $ball_1(u)$ and $ball_1(v)$ overlap, then $d[u,v] = \delta(u,v)$. So let us assume henceforth that $ball_1(u)$ and $ball_1(v)$ do not overlap. We will show the following claim.

Claim 3 *If $ball_1(u)$ and $ball_1(v)$ do not overlap, then some vertex $s \in S_1$ satisfies $d[s,u] + d[s,v] \leq 2\delta(u,v)$.*

The above claim immediately shows a stretch of $2+\varepsilon$ of the distance estimate d computed. If $s = u$ or $s = v$ (which might happen if u or v is in S_1) then the above claim implies that $d[u,v] \leq 2\delta(u,v)$ which is a stretch of just 2 of the distance estimate computed. Hence let us assume that s is neither u nor v. So $1 \leq \delta(s,u) \leq nW$ which implies that $1 \leq d[s,u] \leq 5/2nW$. Since k has been chosen such that $(1+\varepsilon/2)^k \leq 5/2nW$, it follows that there exist i, j where $1 \leq i, j \leq k$ such that $(1+\varepsilon/2)^{i-1} \leq d[s,u] \leq (1+\varepsilon/2)^i$ and $(1+\varepsilon/2)^{j-1} \leq d[s,v] \leq (1+\varepsilon/2)^j$. Thus Boolean product witness matrix for $M_i M_j^T$ would compute the above witness $s \in S_1$ or some other $s' \in S_1$ which has to satisfy $(1+\varepsilon/2)^{i-1} \leq d[s',u] < (1+\varepsilon/2)^i$ and $(1+\varepsilon/2)^{j-1} \leq d[s',v] < (1+\varepsilon/2)^j$. This implies that

$$\begin{aligned} d[u,s'] + d[s',v] &\leq (1+\varepsilon/2)(d[u,s] + d[s,v]) \\ &\leq (2+\varepsilon)\delta(u,v). \end{aligned}$$

Step 4 ensures that $d[u,v] \leq d[u,s'] + d[s',v]$ which shows a stretch of at most $2+\varepsilon$ of the distance estimate computed. So all this is left now is to prove Claim 3.

Proof of Claim 3.

We shall call a vertex $s \in S_1$ that satisfies $d[u,s] + d[s,v] \leq 2\delta(u,v)$ our *witness*. We consider 3 cases here and show that a witness exists in each case.

Case 1: $ball_3(u)$ and $ball_3(v)$ do not overlap.

This case is very simple. We have $\delta(u,s_3(u)) + \delta(v,s_3(v)) \leq \delta(u,v)$. It is easy to check that if $\delta(u,s_3(u)) \leq \delta(v,s_3(v))$, then $s_3(u)$ is our required witness, else $s_3(v)$ is our required witness.

Case 2: $ball_2(u)$ and $ball_2(v)$ overlap.

If all the edges of the shortest path $SP(u,v)$ are present in in the graph G_2 (whose edge set is $E_2 = \cup_x ball_2(x)$), then $d[s_1(u),v] = \delta(s_1(u),v)$ and $d[s_1(v),u] = \delta(s_1(v),u)$. Since $ball_1(u)$ and $ball_1(v)$ do not

overlap, we also have $\delta(u,s_1(u)) + \delta(v,s_1(v)) \leq \delta(u,v)$. It is easy to see that if $\delta(u,s_1(u)) \leq \delta(v,s_1(v))$, then $s_1(u)$ the required witness, else it is $s_1(v)$.

So let us now assume that some of the edges in $SP(u,v)$ are not present in G_2. Let $\delta(u,s_1(u)) \leq \delta(v,s_1(v))$. We can decompose $SP(u,v)$ as $u \rightsquigarrow a \rightarrow b \rightsquigarrow v$ where $a \in ball_2(u)$ and $b \in ball_2(v)$. Observe that the only edge in $SP(u,v)$ that can be missing from $\cup_x ball_2(x)$ is edge (a,b) (refer Fig. 4). So we have $\delta(s_1(a),a) \leq w(a,b)$ since $\delta(s_2(a),a) \leq w(a,b)$ and we anyway know that $\delta(s_1(a),a) \leq \delta(s_2(a),a)$.

If $u \in ball_1(a)$, then b while doing Dijkstra in the graph G_1 augmented with b's edges finds the path $b \rightarrow a \rightsquigarrow u$ to u and this causes $d[s_1(u),b] \leq \delta(s_1(u),u) + \delta(u,b)$. So it is easy to see that $s_1(u)$ is again our required witness. So let us assume that $u \notin ball_1(a)$, as in Fig. 4.

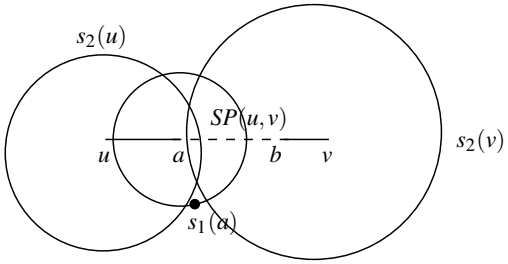

Fig. 4. $ball_2(u)$ and $ball_2(v)$ overlap.

Then $s_1(a)$ is the required witness because:

- $d[s_1(a),u] \leq \delta(s_1(a),a) + \delta(a,u) \leq w(a,b) + \delta(a,u) = \delta(u,b) \leq \delta(u,v)$.
- $d[s_1(a),v] \leq \delta(s_1(a),b) + \delta(b,v) \leq \delta(s_1(a),a) + w(a,b) + \delta(b,v) \leq \delta(u,a) + w(a,b) + \delta(b,v)$, which is $\delta(u,v)$ (in the above inequality we used $\delta(s_1(a),a) \leq \delta(a,u)$ since $u \notin ball_1(a)$).

Note that $d[s_1(a),u] \leq \delta(s_1(a),a) + \delta(a,u)$ because the $a \rightsquigarrow u$ shortest path is present in the graph G_2; also $d[s_1(a),b] \leq \delta(s_1(a),a) + w(a,b)$ and so when $s_1(a)$ does Dijkstra in G_2, it has the "edge" $(s_1(a),b)$ with weight $d[s_1(a),b]$ and the $b \rightsquigarrow v$ shortest path is anyway present in G_2. Thus $d[s_1(a),u] \leq \delta(u,v)$ and $d[s_1(a),v] \leq \delta(u,v)$.

Case 3: $ball_2(u)$ and $ball_2(v)$ do not overlap but $ball_3(u)$ and $ball_3(v)$ overlap.

Let us assume that $\delta(u,s_2(u)) \leq \delta(v,s_2(v))$. Let the shortest path $SP(u,v)$ be $u \rightsquigarrow a \rightarrow b \rightsquigarrow v$ where the edge (a,b) is missing from $\cup_x ball_3(x)$ (otherwise, $s_2(u)$ is our witness). We have three cases again.

(i) $u \notin ball_2(a)$: then we have $\delta(s_2(a),a) \leq \delta(a,u)$. This leads to $s_2(a)$ being our required witness as follows: $d[s_2(a),u] \leq \delta(s_2(a),a) + \delta(a,u) \leq \delta(u,b) \leq \delta(u,v)$ and $d[s_2(a),v] \leq \delta(s_2(a),a) + \delta(a,v) \leq \delta(u,a) + \delta(a,v) = \delta(u,v)$.

(ii) $u \in ball_2(a)$ and $s_2(u) \in bunch_3(a)$: then $d[s_2(u),b] \leq d(s_2(u),a) + w(a,b)$ since a updates the entry $d[s_2(u),b]$ because $s_2(u) \in bunch_3(a)$ (in the second iteration of Step 1 of the algorithm $\mathsf{STRETCH}_{5/2}(G)$) and it is easy to check that this leads to $s_2(u)$ being our required witness.

(iii) $u \in ball_2(a)$ and $s_2(u) \notin bunch_3(a)$: We split this further into two cases.

Case (a): The vertex $b \notin ball_2(v)$.

We will show that $s_3(a)$ is our witness here. We have

$$d[s_3(a),u] + d[s_3(a),v] \leq \delta(s_3(a),a) + \delta(a,u) + \delta(s_3(a),a) + \delta(a,v). \tag{8}$$

We also have $\delta(s_3(a),a) \leq \delta(a,s_2(u))$ (since $s_2(u) \notin bunch_3(a)$) and $\delta(s_3(a),a) \leq w(a,b)$ (since the edge (a,b) is not present in $ball_3(a)$). Substituting these bounds in Inequality (8) yields

$$d[s_3(a),u] + d[s_3(a),v] \leq \delta(u,v) + \delta(a,u) + \delta(u,s_2(u)) + w(a,b).$$

The right hand side is at most $\delta(u,v) + \delta(u,b) + \delta(v,s_2(v))$ by clubbing $\delta(a,u)$ with $w(a,b)$ and bounding $\delta(u,s_2(u))$ by $\delta(v,s_2(v))$. Now $\delta(v,s_2(v))$ is at most $\delta(b,v)$ since $b \notin ball_2(v)$. Thus we have the desired bound of $2\delta(u,v)$ on the right hand side.

Case (b): The vertex $b \in ball_2(v)$.
Then by our earlier analysis we know that if $u \notin ball_1(a)$ then $s_1(a)$ is our witness and if $u \in ball_1(a)$, then $s_1(u)$ is our witness because we have in this case: $d[s_1(u),v] \leq \delta(s_1(u),u) + \delta(u,v)$ (since $d[u,b] = \delta(u,b)$ here) and so $d[s_1(u),u] + d[s_1(u),v] \leq 2\delta(s_1(u),u) + \delta(u,v)$. We can bound $2\delta(s_1(u),u) \leq 2\delta(s_2(u),u) \leq \delta(u,v)$. □

This finishes the proof of Claim 3 and thus Lemma 4. It follows from Lemma 4 that the stretch of distances computed by our algorithm $\mathsf{STRETCH}_{2+\varepsilon}(G)$ is at most $2+\varepsilon$. The bound on the running time of this algorithm was already shown in Section 4.1. Thus we have shown Theorem 2 stated in Section 1.

Conclusions

In this paper we gave a combinatorial algorithm with expected running time $O(n^{9/4})$ to compute all-pairs stretch $5/2$ distances in a weighted undirected graph on n vertices. We then improved this algorithm, with the help of a subroutine for witnessing a Boolean product matrix, to compute all-pairs stretch $2+\varepsilon$ distances. These are the first algorithms to compute all-pairs stretch $3-\varepsilon$ distances for some constant $\varepsilon > 0$ in $o(n^{7/3})$ time. An open question is to further improve the running times of the algorithms given here.

References

1. Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths(without matrix multiplication). *SIAM Journal on Computing*, 28:1167–1181, 1999.
2. Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. Near-linear time construction of sparse neighborhod covers. *SIAM Journal on Computing*, 28:263–277, 1998.
3. S. Baswana, V. Goyal, and S. Sen. All-pairs nearly 2-approximate shortest paths in $O(n^2 \operatorname{polylog} n)$ time. In *22nd Annual Symposium on Theoretical Aspect of Computer Science*, pages 666–679, 2005.
4. S. Baswana and T. Kavitha. Faster algorithms for approximate distance oracles and all-pairs small stretch paths. In *47th IEEE Symposium on Foundations of Computer Science*, pages 591 – 602, 2006.
5. Timothy Chan. All-pairs shortest paths with real edge weights in $O(n^3/\log n)$ time. In *Proceedings of Workshop on Algorithms and Data Structures*, volume 3608, pages 318–324, 2005.
6. Edith Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. *SIAM Journal on Computing*, 28:210–236, 1998.
7. Edith Cohen and Uri Zwick. All-pairs small stretch paths. *Journal of Algorithms*, 38:335–353, 2001.
8. Dorit Dor, Shay Halperin, and Uri Zwick. All pairs almost shortest paths. *Siam Journal on Computing*, 29:1740–1759, 2000.
9. Michael Elkin. Computing almost shortest paths. *ACM Transactions on Algorithms (TALG)*, 1:282–323, 2005.
10. Xiaohan Huang and Victor Y. Pan. Fast rectangunlar matrix multiplication and applications. *Journal of Complexity*, 14:257 –299, 1998.
11. Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, 1995.
12. Seth Pettie. A new approach to all-pairs shortest paths on real-weighted graphs. *Theoretical Computer Science*, 312:47–74, 2004.
13. Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of Association of Computing Machinery*, 52:1–24, 2005.
14. Uri Zwick. All-pairs shortest paths using bridging sets and rectangular matrix multiplication. *Journal of Association of Computing Machinery*, 49:289–317, 2002.

Appendix

Proof of Proposition 1

Consider the shortest path $u(=v_0), v_1, \cdots, v_j(=v)$ between u and v. Since $\delta(u,S_i) > \delta(u,v)$, it follows that

$$\delta(u,S_i) > \delta(u,v_k) \quad \forall k \leq j. \tag{9}$$

All we need to show here is that all the edges of the shortest path are present in E_i. We shall prove this assertion by contradiction. Let (v_k, v_{k+1}) be the first edge (from the side of u) on the path which is absent. Since this edge is not present in E_i , it follows from the definition of E_i that there is some vertex $s \in S_i$ such that $\delta(v_k,s) < w(v_k,v_{k+1})$. This fact, when combined with Equation (9), gives us

$$\begin{aligned}
\delta(u,S_i) &> \delta(u,v_{k+1}) \\
&= \delta(u,v_k) + w(v_k,v_{k+1}) \qquad \{ \text{ since } u \rightsquigarrow v_k \rightarrow v_{k+1} \rightsquigarrow v \text{ is the shortest path} \} \\
&> \delta(u,v_k) + \delta(v_k,s) \quad \geq \quad \delta(u,s).
\end{aligned}$$

Since $s \in S_i$, this contradicts the very definition of $\delta(u,S_i)$. Hence all the edges on the shortest path between u and v are present in E_i. This proves the first part of the lemma.

If $u(=v_0), v_1, v_2, \cdots, v_\ell, s_i(u)$ is the shortest path between u and $s_i(u)$, it follows from the first part of the lemma that every edge of this path, excluding the last edge, must be present in E_i. Thus the whole path is present in $E_i \cup E(s_i(u))$ since $E(s_i(u))$ includes the last edge of the above path. This proves the second part of the lemma. □

Proof of Proposition 2

For any $v \in V$, the expected size of $ball_i(v)$ is the expected number of edges in E_i that are incident on v. Let the symbol $E_i(v)$ denote the set of edges in E_i that are incident on v. Let us calculate the expected size of $E_i(v)$ for any $v \in V$. If the vertex v belongs to the set S_i, then $E_i(v) = \emptyset$.

Let us now estimate $E_i(v)$ when $v \notin S_i$. Consider the sequence $\langle v_1, v_2, \ldots \rangle$ of neighbors of v arranged in non-decreasing order of the edge weights $w(v,v_k)$. If an edge (v,v_k) belongs to E_i, then none of the vertices $v_1, \ldots, v_k$ is present in S_i. Since each vertex is selected in S_i independently with probability q,

$$\mathbf{Pr}[(v,v_k) \in E_i] \leq (1-q)^k.$$

Using linearity of expectation, if $v \notin S_i$, then the expected number of edges in $E_i(v)$ is at most $\sum_k (1-q)^k = O(1/q)$. Hence the expected number of edges in E_i is $O(n/q)$. □

Proof of Lemma 3 (Case 2)

Case 2: $ball_2(u)$ and $ball_2(v)$ overlap.

Let us first look at some easy cases. We know that $ball_1(u)$ and $ball_1(v)$ do not overlap. Hence $\delta(s_1(u),u) + \delta(s_1(v),v) \leq \delta(u,v)$. Assume without loss of generality that $\delta(s_1(u),u) \leq \delta(s_1(v),v)$. Then the first inequality implies that $2\delta(s_1(u),u) \leq \delta(u,v)$.

Since $ball_2(u)$ and $ball_2(v)$ overlap, $SP(u,v)$, the shortest path between u and v, can be decomposed as $u \rightsquigarrow a \rightarrow b \rightsquigarrow v$ where (a,b) is an edge in $SP(u,v)$ such that $a \in ball_2(u)$ and $b \in ball_2(v)$ (refer Fig. 5).

If the vertex a in $SP(u,v)$ belongs to $ball_1(u)$ or $u \in ball_1(a)$, then we have $d[u,b] = \delta(u,b)$ at the end of the first iteration of the ** loop (the same analysis as given in the proof of Claim 2 shows this). In the

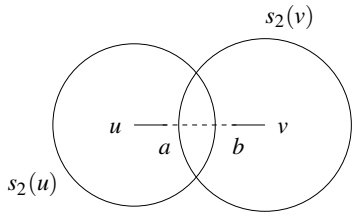

Fig. 5. $ball_2(u)$ and $ball_2(v)$ overlap.

second iteration of the ** loop, in Step 1 of the algorithm $\mathsf{STRETCH}_{5/2}(G)$ (Section 3), u would update $d[s_1(u),b] \leq d[s_1(u),u]+d[u,b]$ since $s_1(u) \in bunch_1(u)$. So $d[s_1(u),b]$ is at most $\delta(u,s_1(u))+\delta(u,b)$. The vertex $s_1(u)$ performs Dijkstra in the second iteration of the ** loop in Step 2 of this algorithm, with the "edge" $(s_1(u),b)$ with weight at most $d[s_1(u),b]$. This yields

$$d[u,v] \leq \delta(u,v)+2\delta(u,s_1(u)) \tag{10}$$
$$\leq 2\delta(u,v). \tag{11}$$

Inequality (11) uses our assumption that $2\delta(s_1(u),u) \leq \delta(u,v)$. Similarly, if the vertex $s_1(u) \in bunch_1(a)$, then again we have $d[s_1(u),b] \leq \delta(s_1(u),u)+\delta(s_1(u),b)$ because a updates $d[s_1(u),b]$ in Step 1 since $s_1(u) \in bunch_1(a)$. This again leads to Inequalities (10) and (11) above, which show a stretch 2 for the distance $d[u,v]$ computed. Hence let us assume from now that $a \notin ball_1(u))$, $u \notin ball_1(a)$ and $s_1(u) \notin bunch_1(a)$.

Also, if all the edges in $SP(u,v)$ are present in $G_2=(V,E_2)$, then since $ball_1(u)$ and $ball_1(v)$ do not overlap, Claim 1 applies here and so we get $d[u,v] \leq 2\delta(u,v)$. Thus we shall assume that some of the edges in $SP(u,v)$ are *not* present in $G_2=(V,E_2)$. This leaves us with two cases.

CASE(I): All the edges in $SP(u,v)$ are present in G_3 but some of these edges are not present in G_2.

Since all the edges in $SP(u,v)$ are present in $G_3=(V,E_3)$ and vertices in S_2 perform Dijkstra in G_3, we have

$$d[u,v] \leq \delta(u,v)+2\delta(u,s_2(u)) \tag{12}$$
$$d[u,v] \leq \delta(u,v)+2\delta(v,s_2(v)). \tag{13}$$

We would like to obtain upper bounds, first on $\delta(u,s_2(u))$ and then on $\delta(v,s_2(v))$. Recall that $s_1(a) \notin bunch_1(u)$. So $\delta(u,s_2(u)) \leq \delta(u,s_1(a)) \leq \delta(u,a)+\delta(a,s_1(a))$. Substituting this upper bound on $\delta(u,s_2(u))$ in Inequality (12), we get

$$d[u,v] \leq \delta(u,v)+2\delta(u,a)+2\delta(a,s_1(a)) \tag{14}$$
$$\leq \delta(u,v)+4\delta(u,a) \quad [\text{since } u \notin ball_1(a), \text{ we have } \delta(a,s_1(a)) \leq \delta(a,u)] \tag{15}$$

Recall that $SP(u,v)$, the shortest path between u and v in G can be decomposed as $u \rightsquigarrow a \rightarrow b \rightsquigarrow v$, where $a \in ball_2(u)$ and $b \in ball_2(v)$. Since all the edges of $SP(u,v)$ are not present in E_2, it means that the edge (a,b) is not present in $ball_2(x)$ for any $x \in V$. Since the edge (a,b) is not present in $\cup_x ball_2(x)$, it follows that $\delta(v,a) \geq \delta(v,S_2) = \delta(v,s_2(v))$. Substituting this in Inequality (13) we get

$$d[u,v] \leq \delta(u,v)+2\delta(v,a). \tag{16}$$

Combining Inequalities (16) and (15), we get

$$d[u,v] \leq \delta(u,v)+\min[4\delta(u,a),2\delta(v,a)]. \tag{17}$$

Note that $\delta(u,a)+\delta(a,v)=\delta(u,v)$. Thus $\min(2\delta(u,a),\delta(a,v))=\min(2\delta(u,a),\delta(u,v)-\delta(u,a))$ is at most $2/3\cdot\delta(u,v)$. Thus Inequality (17) always yields $d[u,v]\leq 7/3\cdot\delta(u,v)$. This finishes CASE(I).

CASE (II): Some of the edges in $SP(u,v)$ are not present in $G_3=(V,E_3)$.

This means that the edge (a,b) (refer Fig. 5) is missing from E_3. All the edges in the portions $u\rightsquigarrow a$ and $b\rightsquigarrow v$ in $SP(u,v)$ are present in E_2 since $ball_2(u)$ and $ball_2(v)$ overlap (Fig. 5). Thus the vertices $s_1(u)$ and $s_1(v)$ can see all the edges in $SP(u,v)$ except the edge (a,b). Since vertices in S_2 update entries in the table d in the rows corresponding to vertices in S_1 in Step 3, we have $d[s_1(u),b]\leq d[s_1(u),s_2(u)]+d[s_2(u),b]$, which is at most $\delta(s_1(u),u)+\delta(u,a)+2\delta(a,s_2(a))+w(a,b)$. In the second iteration of the ** loop, $s_1(u)$ performs Dijkstra in G_2 with an "edge" $(s_1(u),b)$ with weight at most $d[s_1(u),b]$. Thus we get

$$d[s_1(u),v]\leq\delta(s_1(u),u)+\delta(u,v)+2\delta(a,s_2(a)). \tag{18}$$

Since $s_1(u)\notin bunch_1(a)$, we have $\delta(a,s_2(a))\leq\delta(a,s_1(u))\leq\delta(a,u)+\delta(u,s_1(u))$. We also have $\delta(u,s_1(u))\leq\delta(u,a)$ since $a\notin ball_1(u))$. Using these two inequalities in Inequality (18) and using the fact that $d[u,v]\leq\delta(u,s_1(u))+d[s_1(u),v]$ gives us:

$$d[u,v]\leq\delta(u,v)+6\delta(u,a). \tag{19}$$

We also have another upper bound on $d[u,v]$. The fact that the edge (a,b) is not present in $ball_3(a)$ leads to Inequalities (3)-(5). Combining Inequality (19) and Inequality (5), we have

$$d[u,v]\leq\delta(u,v)+\min[6\delta(u,a),2w(a,b)]. \tag{20}$$

Since $\delta(u,a)+w(a,b)\leq\delta(u,v)$, we have $\min[6\delta(u,a),2\delta(u,v)-2\delta(u,a)]\leq 3\delta(u,v)/2$. Thus Inequality (20) shows that $d[u,v]\leq 5\delta(u,v)/2$ in this case. This finishes Case 2 in the proof of Lemma 3. □

Algorithm for computing Boolean Product Witness Matrix

Here we describe an algorithm from [11] to compute a witness matrix W_{ij} corresponding to the Boolean product of two 0-1 matrices M_i and M_j^T, of dimensions $n\times n^\beta$ and $n^\beta\times n$, respectively. We will need the following notation in the algorithm: for any 0-1 vector R of length n^β, let $R(M_i)$ be the $n\times n^\beta$ matrix whose the (k,ℓ)-th entry is $\ell\cdot R[\ell]M_i[k,\ell]$. Further, let $R(M_j^T)$ be the $n^\beta\times n$ matrix whose the (ℓ,k)-th entry is $R[\ell]M_j^T[\ell,k]$.

Input: Two 0-1 matrices M_i and M_j^T

Output: Witness matrix W_{ij} for the Boolean product matrix $M_iM_j^T$

1. Let $W_{ij}=-M_iM_j^T$
2. for $t=0,\ldots,\lfloor\beta\log n\rfloor$ do
   (a) $r=2^t$
   (b) repeat $\lceil 3.77\log n\rceil$ times
   - choose random $R\subseteq\{1,2,\ldots,n^\beta\}$ with $|R|=r$. Encode R as a 0-1 incidence vector of length n^β.
   - compute $R(M_i)$ and $R(M_j^T)$.
   - $Z=R(M_i)\cdot R(M_j^T)$.
   - for all $(k,\ell)\in[n]\times[n]$ do
     if $W_{ij}[k,\ell]<0$ and $Z[k,\ell]$ is a witness then $W_{ij}[k,\ell]=Z[k,\ell]$.
3. for all $(k,\ell)\in[n]\times[n]$ do
   if $W_{ij}[k,\ell]<0$ then find a witness by brute force and set $W_{ij}[k,\ell]$ to that witness.

The above algorithm is a Las Vegas algorithm for computing a Boolean product witness matrix.