Polylog-time and near-linear work approximation scheme for undirected shortest paths

Edith Cohen
AT&T Labs–Research
Florham Park, NJ 07922
edith@research.att.com

Submitted July 1994; Revised July 1998

Abstract

[1] Shortest paths computations constitute one of the most fundamental network problems. Nonetheless, known parallel shortest-paths algorithms are generally inefficient: they perform significantly more work (product of time and processors) than their sequential counterparts. This gap, known in the literature as the "transitive closure bottleneck," poses a long-standing open problem. Our main result is an $O(mn^{\epsilon_0} + s(m + n^{1+\epsilon_0}))$ work polylog-time randomized algorithm that computes paths within $(1+O(1/\operatorname{polylog} n))$ of shortest from s source nodes to all other nodes in weighted undirected networks with n nodes and m edges (for any fixed $\epsilon_0 > 0$). This work bound nearly matches the $\tilde{O}(sm)$ sequential time. In contrast, previous polylog-time algorithms required $\min\{\tilde{O}(n^3), \tilde{O}(m^2)\}$ work (even when $s = 1$), and previous near-linear work algorithms required near-$O(n)$ time. We also present faster sequential algorithms that provide good approximate distances only between "distant" vertices: We obtain an $O((m + sn)n^{\epsilon_0})$ time algorithm that computes paths of weight $(1 + O(1/\operatorname{polylog} n))\text{dist} + O(w_{\max} \operatorname{polylog} n)$, where dist is the corresponding distance and $w_{\max}$ is the maximum edge weight. Our chief instrument, which is of independent interest, are efficient constructions of sparse *hop sets*. A (d, ϵ)-hop set of a network $G = (V, E)$ is a set E^* of new weighted edges such that minimum-weight d-edge paths in $(V, E \cup E^*)$ have weight within $(1 + \epsilon)$ of the respective distances in G. We construct hop sets of size $O(n^{1+\epsilon_0})$ where $\epsilon = O(1/\operatorname{polylog} n)$ and $d = O(\operatorname{polylog} n)$.

[1]Preliminary version appeared in Proc. 26th Annual ACM Symposium on Theory of Computing, 1994

1 Introduction

Shortest paths computations in weighted networks constitute one of the most fundamental problems in combinatorial optimization. They are used extensively in practice and often arise as a subroutine in the solution of other problems. Consider a network $G = (V, E)$, where weights are associates with edges. We denote $n = |V|$ and $m = |E|$. The single-source shortest paths problem amounts to computing minimum weight paths from a designated source node to all other nodes. The all-pairs problem amounts to computing minimum weight paths between all pairs of nodes. Satisfactory sequential algorithms exist for variants of the problem: The single-source problem on graphs with nonnegative weights is solved almost optimally by Dijkstra's algorithm and improvements. The all-pairs problem on graphs with real weights is solved in $O(mn)$ time (see e.g., [6]). A standard measure of efficiency in evaluating parallel algorithms is comparing the work performed (product of time and number of processors) by the parallel algorithm to the time of the fastest-known sequential algorithm. The literature contains numerous parallel shortest paths algorithms that exhibit a wide range of work/time tradeoffs. These algorithms, however, badly fail this measure: As the target running time decreases, the work performed by the best known algorithms approaches $\tilde{O}(n^3)$,[2] even for the single-source problem and even when only approximate distances to within $(1+\epsilon)$ of shortest are required (where $\epsilon < 1$). Known polylogarithmic time shortest paths algorithms amounted to performing a transitive-closure type computation that requires $\tilde{O}(n^3)$ work. We note that a recent polylogarithmic time algorithm by Klein and Sairam [9] exhibits an improvement for sparse graphs and solves the single-source problem using $\tilde{O}(m^2)$ work. Although faster linear-work algorithms were known for classes of graphs with special structure, e.g., planar graphs, the existence of efficient parallel algorithms for shortest paths on general graphs is a long standing open problem. The inability to solve shortest paths efficiently in parallel is dubbed in the literature "the transitive closure bottleneck." The effort to obtain better parallel algorithms led to considering the relaxed problem of computing paths that are within $(1+\epsilon)$ of shortest, where typically $\epsilon = O(1/\operatorname{polylog} n)$. An additional incentive for considering the relaxed problem is that approximate shortest paths are adequate for many applications. Previously, this relaxation gave rise to some improved work-time tradeoffs, but the difficulties depicted in the discussion above applied to the relaxed problem as well. To match in efficiency the sequential algorithm for the single-source problem, we aspire for a near-linear work parallel algorithm. Under this condition, however, the best previously achievable running time was $O(n)$. (A parallel implementation of Dijkstra's algorithm runs in $O(n \log n)$ time and performs $O(m \log n)$ work.) The main contribution of this paper is overcoming the transitive closure bottleneck for approximate shortest-paths on weighted undirected graphs. We show the following

Theorem 1.1 *For any fixed $\epsilon_0 > 0$ and $d > 0$, paths from s sources within $(1 + O(1/\log^d n))$ of shortest can be computed by a randomized algorithm in polylogarithmic time using*

$$O(mn^{\epsilon_0} + s(m + n^{1+\epsilon_0}))$$

work.

We also provide tradeoffs that exhibit improved approximation and work when the time is somewhat compromised. All our results are applicable to undirected graphs with general nonnegative weights.

Another contribution is relevant to both sequential and parallel shortest paths computations. The best known sequential time bound for computing shortest paths from s sources is $\tilde{O}(sm)$. An obvious lower bound (if an explicit representation of the distances is required) is $O(m + sn)$. This

[2] We use the notation $\tilde{O}(f) = O(f \operatorname{polylog} n)$.

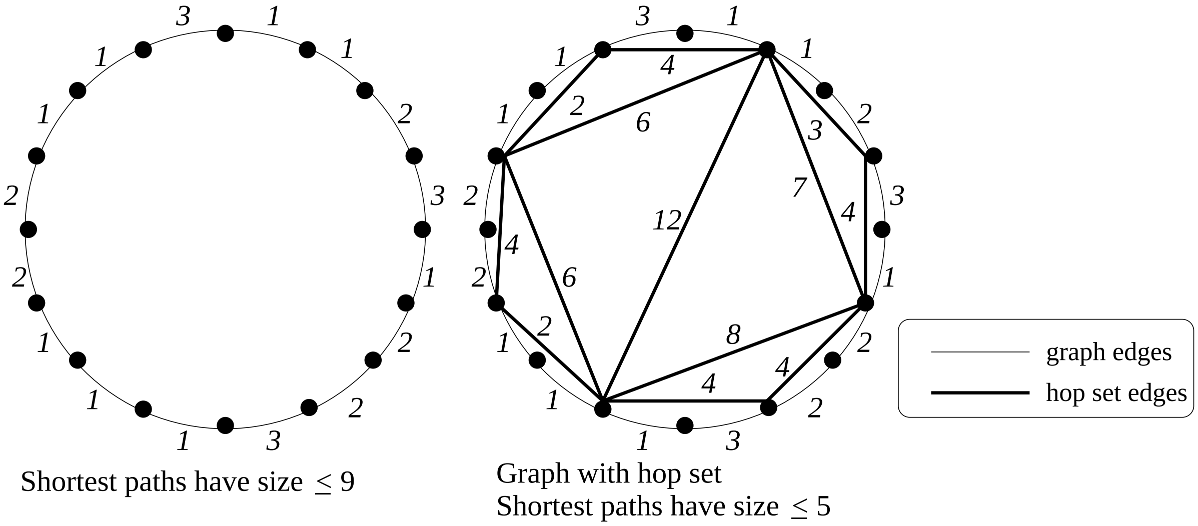

Figure 1: A hop set for a weighted circle graph

gap is significant for dense graphs (a graph is *dense* if $m = \Omega(n^{1+\epsilon})$ for some fixed $\epsilon > 0$). We use our techniques to obtain a sparse set of edges E^*, such that distances in (V, E^*) approximate long distances in E. Therefore, for distant vertex-pairs, shortest paths on E^*, that can be computed in $O(s|E^*|) \ll O(s|E|)$ time, have weights within $(1 + O(1/\operatorname{polylog} n))$ of the corresponding distances in E.

Theorem 1.2 *Let (V, E) be a weighted undirected graph. For any fixed $\epsilon_0 > 0$ and $d > 0$, paths within $(1 + O(1/\log^d n))$ of shortest between pairs of vertices that are at least $\Omega(w_{\max} \operatorname{polylog} n)$ apart, can be computed in $O((m + sn)n^{\epsilon_0})$ time, where $w_{\max}$ is the maximum edge-weight (or in polylogarithmic time using comparable expected work).*

It is worth noting that faster than $O(sm)$ algorithms were known for t-stretch paths. A path has stretch t, if its weight is at most t times the weight of the respective minimum-weight path. Awerbuch et al. [1] presented an algorithm for computing t-stretch paths between k pairs in $\tilde{O}(mn^{64/t} + kn^{32/t})$. A later algorithm by the author [5] runs in expected $\tilde{O}((m + k)n^{(2+\epsilon)/t})$ time. These bounds are meaningful, however, only when the stretch is large ($t \geq 4$). Theorem 1.2 can be viewed as a fast short-paths algorithm that produces paths within an additive term from shortest as opposed to the multiplicative factor that corresponds to the stretch.

A contribution of independent interest is our *hop sets* constructions. A (d, ϵ)-hop set of a weighted graph $G = (V, E)$ is a collection of weighted edges E^* such that for every pair of vertices, the minimum-weight d-edge path between them in $(V, E \cup E^*)$ has weight within $(1 + \epsilon)$ of the corresponding shortest path in (V, E). We refer to d as the *diameter* of the hop set and to ϵ as the *approximation quality*. The set E^* of hop edges is chosen such that it does not decrease distances. See Figure 1 for an example. The objective, typically, is to obtain sparse hop sets with small diameter and good approximation quality (preferably *exact* hop sets, where $\epsilon = 0$). One intriguing issue is the *existence* question of sparse hop sets with certain attributes. In addition, we would like to construct good hop sets efficiently. We present hop set algorithms with several tradeoffs. One tradeoff is the following:

Theorem 1.3 *For any fixed $\epsilon_0 > 0$ and integer k, $(O(\operatorname{polylog} n), O(1/\log^k n))$-hop sets of size $O(n^{1+\epsilon_0})$ can be computed sequentially in $O(mn^{\epsilon_0})$ time and in parallel by a polylogarithmic time randomized algorithm using $O(mn^{\epsilon_0})$ work.*

A related concept to hop sets is graph spanners. A t-spanner is a set of weighted edges such that distances in the spanner are not smaller, and within a factor of t of distances in the original graphs. In [5] the author provided efficient algorithms to construct sparse spanners with the additional property that small-size shortest paths on the spanner constitute factor-t approximation of original distances. The stretch factor considered is $t \geq 4$, resulting in much worse approximation quality than in hop sets. We remark that our parallel hop-set algorithms can be combined with the algorithms of [5] and yield polylogarithmic-time parallel algorithms for t-spanners and stretch-t paths, with work comparable to the sequential time bounds given in [5] (currently best known) and an additional additive term of $O(mn^{\epsilon_0})$ (for any fixed $\epsilon_0 > 0$).

Hop-sets can be employed as follows to compute shortest paths in parallel. *Limited shortest paths* , or *d-edge* shortest paths, are minimum weight paths that contain at most d edges. When d is small, d-edge shortest paths can be computed efficiently in parallel: By d-iteration Bellman-Ford algorithm in $O(d \log n)$ time using $O(md)$ work per source, or by a parallel weighted BFS algorithm (introduced by Klein and Sairam [8]) that produces an approximate solution using $\tilde{O}(m)$ work. Suppose we are given a (d, ϵ)-hop set, E^*, for a graph $G = (V, E)$. In order to approximate distances in G, it suffices to compute respective d-edge shortest paths in $(V, E \cup E^*)$. Hence, $(1 + \epsilon)$-approximate distances in G can be computed efficiently in $\tilde{O}(d)$ parallel time.

Many previous parallel shortest paths algorithms utilize hop sets explicitly or implicitly. We mention several parallel shortest paths algorithms and discuss their use of hop sets. Generally in these algorithms, the existence question of hop sets with these attributes is immediate and the main contribution is the construction algorithm. Ullman and Yannakakis [11] presented a parallel shortest paths algorithm for directed graphs with unit weights. Their algorithm chooses a random subset of $O(n/t)$ vertices. The hop set consists of a complete graph on this subset, and has diameter $\tilde{O}(t)$ and $\epsilon = 0$. Klein and Sairam [8] gave a parallel randomized approximation scheme for $(1 + \epsilon)$-approximate single-source shortest paths computations that runs in $\tilde{O}(\epsilon^{-2} n^{0.5})$ time and performs $\tilde{O}(mn^{0.5})$ work. Their algorithm uses the Ullman and Yannakakis idea and constructs a hop set of diameter $\tilde{O}(n^{0.5})$ and size $\tilde{O}(n)$. The author [4] presented a $\tilde{O}(t)$ time $\tilde{O}(sn^2 + n^3/t^2)$ work algorithm that is exact when edge-weights ratios are polynomially bounded and $(1 + 1/\operatorname{poly} n)$ approximate otherwise, and a $\tilde{O}(s(n^2/t + m) + n^3/t^2)$ work algorithm that produces $(1 + 1/\operatorname{polylog} n)$ approximation. The algorithms in [4] are based on the selective path doubling technique and construct in polylog time and $\tilde{O}(n\delta^2)$ work a hop set of size $\tilde{O}(n\delta^2)$ and diameter $\tilde{O}(n/\delta)$. Spencer [10] presented a $\tilde{O}(m + n^3/t^2)$ $\tilde{O}(t)$ time algorithm for single-source shortest paths on a directed graph, when the edge-weights ratios are polynomially bounded. His algorithm construct data structures that he termed "nearby lists." These data structures can also be viewed as hop sets obtained by connecting each vertex to its $O(n/t)$ nearest neighbors. Better parallel shortest paths algorithm are known for graphs with special structure. The author [3] presented a shortest paths algorithm for directed graphs with real weights and a decomposition using $O(n^\mu)$-separators. The algorithm runs in polylogarithmic time and performs $\tilde{O}(n^{3\mu} + sn^{2\mu})$ work. It produces a hop set of size $\tilde{O}(n^{2\mu})$, logarithmic diameter, and $\epsilon = 0$. Klein and Sairam [9] presented a polylogarithmic time (when weights are integral and polynomial) $\tilde{O}(sn)$ work algorithm for directed planar graphs. Their algorithm construct hop sets of size $\tilde{O}(n)$, polylogarithmic diameter, and approximation $(1 + O(1/\operatorname{polylog} n))$. We remark that the hop sets produced by the latter algorithms are interesting also from the existence perspective.

Overview

We present our main ideas an techniques through an overview. We first outline a simple sequential construction of hop sets. We then sketch further ideas needed to first improve hop-set size and

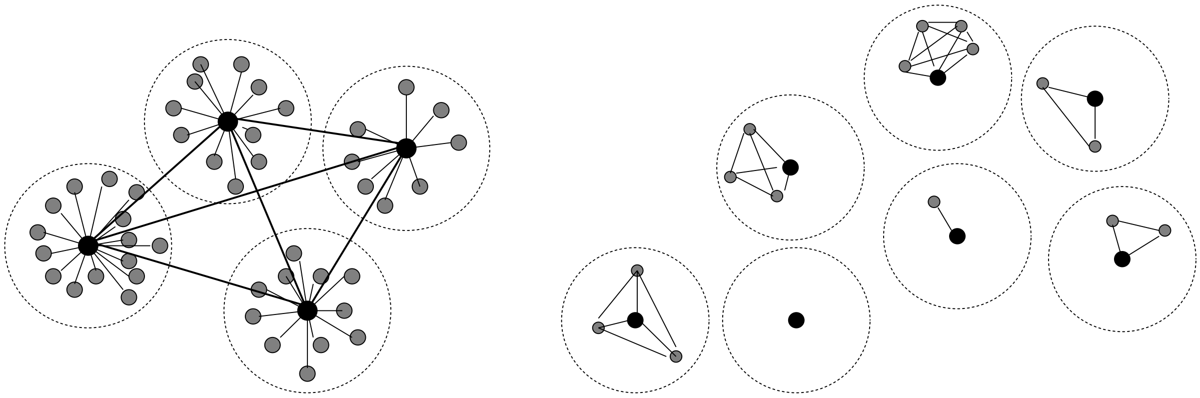

Figure 2: Hop set edges

(sequential) construction time, and then to obtain a parallelized construction.

A crucial building block for our hop set algorithms are efficient constructions of *pairwise covers* [5]. Consider a weighted graph $G = (V, E)$ and a positive number W. A W-pairwise cover of G is a collection χ of subsets of V (*clusters*). Such that (i) for every pair of vertices $\{v_1, v_2\}$ of distance at most W, there exists a cluster $X \in \chi$ such that $\{v_1, v_2\} \subset X$. (ii) The diameter of each $X \in \chi$ is at most $W \log n$, and (iii) The sum of the sizes of all cluster is $O(n)$ and the sum of edge occurrences in clusters is $O(m)$. We make use of a stronger property of the covers constructed in [5]: For every path p such that $w(p) \leq W$, there exists $X \in \chi$ such that $p \subset X$. Pairwise covers are a modification of neighborhoods covers introduced by Awerbuch and Peleg [2] and used extensively in the context of distributed computing. Awerbuch et al. [1] presented near-linear time sequential constructions of neighborhood covers. The author [5] improved trade-offs in sequential constructions and presented a randomized parallel algorithm that is imperative for our parallel hop-set and shortest paths results.

We sketch an algorithm that for a parameter $\epsilon > 0$, constructs a $(O(\epsilon^{-1} \log n), \epsilon)$-hop set of size $\tilde{O}(n^{4/3})$. The algorithm employs a subroutine that for a given $R \in \mathcal{R}_+$, computes a *restricted hop set*, that is, a set of edges that constitutes a hop set for pairs of vertices such that the distance between them is in $[R, 2R]$. A complete hop set is obtained by applying the subroutine with different values of R, and taking the union of the resulting hop sets. A reduction due to Klein and Sairam [8] allows us to assume that the ratio between the largest and smallest distances is polynomial in n. Hence, a logarithmic number of subroutine calls is sufficient. To obtain the restricted hop set, we construct a cover χ with $W = \epsilon R/(4 \log n)$. We partition χ to *big clusters* containing more that $n^{1/3}$ vertices and *small clusters* containing at most $n^{1/3}$ vertices. For each big cluster, we select a representative vertex. The restricted hop set consists of (i) a complete graph on the representative vertices of big clusters, where edges are weighted by corresponding distances, (ii) a complete graph on the vertices of each small cluster, where edges are weighted by corresponding distances, and (iii) a star graph in each big cluster, centered at the representative vertex, where the edges are assigned weight $W \log n$. See Figure 2 for an illustration. It is easy to verify that the size of the hop set is $O(n^{4/3})$, and that distances in the hop set are not smaller than distances in (V, E). We claim that for every pair $\{v_1, v_2\}$ of vertices of distance $\text{dist}_E(v_1, v_2) \in [R, 2R]$, there exists a path $p^* \subset E^* \cup E$ of size $|p^*| \leq O(\epsilon^{-1} \log n)$ and weight $w(p^*) \leq (1 + \epsilon)\text{dist}_E(v_1, v_2)$. Consider a path p between v_1 and v_2 of weight $w(p) = \text{dist}_E(v_1, v_2)$. To construct p^*, partition p to single edges and segments of weight at most W, such that there are $O(\epsilon^{-1} \log n)$ segments. Each segment

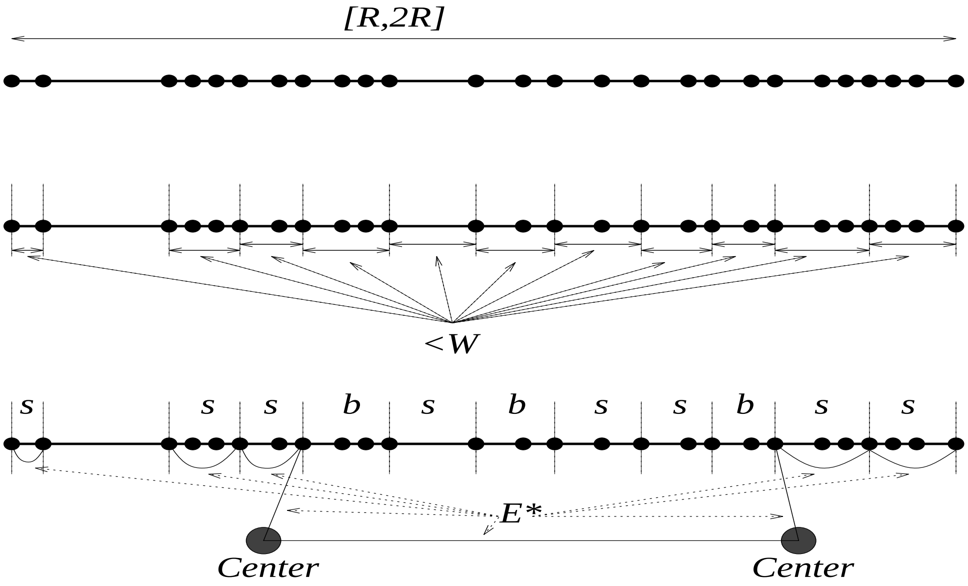

Figure 3: A path, a partition to segments, and the corresponding hop edges

of weight at most W is contained in some cluster. See Figure 3 for an illustration. Segments contained in some small cluster (marked by "s" in the illustration) have a corresponding single E^* edge. Consider the minimal subpath that includes all segments contained only in big clusters (marked by "b"). This subpath can be replaced by an E^* path of size 3, consisting of two edges between endpoints of the subpath and the respective representatives of big cluster that contain these endpoints, and one edge between the representatives. It is easy to see that the weight of the E^* path is larger by an additive term of at most $4W \log n$ than the weight of the original subpath. By combining the above, we obtain a $E \cup E^*$ path of size $O(\epsilon^{-1} \log n)$ and weight at most $\mathrm{dist}_E(v_1, v_2) + 4W \log n \leq \mathrm{dist}_E(v_1, v_2) + \epsilon R \leq \mathrm{dist}_E(v_1, v_2)(1 + \epsilon)$.

We provide some intuition for Theorem 1.2. Note that if the original edge weights are all smaller than $W = \epsilon R/(4 \log n)$, the constructed path p^* consists only of E^* edges. Hence, to obtain $(1 + \epsilon)$-approximate distances for distances larger than R, it suffices to perform shortest paths computations on the (possibly sparser) graph (V, E^*).

In order to obtain a faster construction algorithm and sparser hop sets, we use a recursive version of the algorithm. Following the non-recursive version, the algorithm constructs a cover. Clusters of size larger that $n^{1-\epsilon_0}$ are designated big clusters, and other clusters are designated small clusters. The algorithm treats big cluster as in the nonrecursive version, and generates a collection of hop edges. Instead of computing a complete graph for small clusters, however, the algorithm is applied recursively to produce sparser hop sets. The idea is that we can afford "denser" hop sets on the subgraphs the recursive calls are applied to. At the bottom level of the recursion we use a complete graph.

To construct hop sets in parallel, we consider *limited pairwise covers* defined with respect to a parameter ℓ. In limited covers, only pairs of vertices that have an ℓ-edge path of weight at most W are required to be both contained in some cluster. The author [5] presented a near-linear work and $\tilde{O}(\ell)$ expected time construction of limited covers. If limited covers are used in the hop set algorithm, the algorithm produces *limited hop sets*. Limited hop sets are such that only ℓ-edge distances in (V, E) are approximated by d-edge distances in $(V, E \cup E^*)$. Limited hop set are constructed with efficient work in time $\tilde{O}(\ell)$. A critical property is that the hop set attributes (size, diameter, and approximation quality) are independent of our choice of ℓ. Obviously, when $\ell = \omega(\text{polylog}(n))$, the limited hop set algorithm takes longer than polylogarithmic time. To obtain full (unlimited) hop sets, however, we need $\ell = n$, and therefore a direct application of the limited hop set algorithm would not work. Roughly, to obtain full hop sets in polylogarithmic time, we fix $d = \text{polylog}(n)$. We perform $O(\log n)$ iterations, where in each iteration we apply the limited hop sets algorithm with $\ell = 2d$ to the graph augmented by the hop sets produced in previous iterations.

In Section 2 we give some notation and review pairwise covers and limited parallel shortest paths computation. We also define hop sets and apply hop sets to compute shortest paths efficiently in parallel. Section 3 and 4 present a sequential hop set algorithm. Section 3 presents and analyses a restricted hop set algorithm that produces hop edges for vertex pairs within some range of distances. Section 4 presents two complete hop set algorithms. One algorithm is based on applying the algorithm of Section 3. The other uses a somewhat different scheme and yields different tradeoffs. Sections 5–8 present a parallel hop set algorithm. Both schemes used for the sequential hop set constructions can be employed to obtain parallel hop set algorithms. For brevity, however, we present and analyze only one. Section 5 presents a parallel construction of limited hop sets. The complexity analysis is given in Section 6 and the correctness in Section 7. Section 8 employs limited hop sets to construct full hop sets. Section 9 contains concluding remarks and discusses open problems.

2 Preliminaries

Notation We use $\mathcal{R}_+$ to denote the set of nonnegative real numbers. Consider a graph $G = (V, E)$ with weights $w : E \leftarrow \mathcal{R}_+$. We denote $w_{\max} = \max_{e \in E} w(e)$ and $w_{\min} = \min_{e \in E} w(e)$. For a path $p \subset E$, the size of the path $|p|$ is the number of edges and the weight of the path $w(p) = \sum_{e \in p} w(e)$, is the sum of the weights of edges in the path. A shortest path between a pair of vertices is a minimum weight path connecting them. For an integer d, a d-edge shortest paths (or limited shortest path) is a minimum weight path containing at most d edges. Note that n-edge shortest paths are shortest paths. For two vertices $\{v_1, v_2\} \subset V$, an integer $0 \le d \le n$, and a set of weighted edges $E' \subset V \times V$, we denote by $\text{dist}^d_{E'}(v_1, v_2)$ the weight of the d-edge shortest path in E', between v_1 and v_2. We refer to $\text{dist}^d_{E'}(v_1, v_2)$ as the d-edge distance in E' between v_1 and v_2. If d is omitted assume $d = n$. If E' is omitted assume $E' = E$.

Polynomial-ratio assumption Throughout the paper we assume that the edge weights have a polynomial ratio, that is $w_{\max}/w_{\min} = \text{poly}(n)$ (hence, there is polynomial-ratio between the distances of the closest and furthest pairs of vertices). The assumption is supported by a reduction due to Klein and Sairam [8] (see also [4]), which transformed the problem of computing $(1 + 1/\text{poly}(n))$-approximate shortest paths on graphs with arbitrary nonnegative weights to computing exact shortest paths on a collection of graphs with polynomial ratio. The collection of graphs contains $O(m \log n)$ edges in total and can be produced in $O(m \log n)$ time sequentially, and in logarithmic time with $O(m \log n)$ in parallel. Alternatively, if $(1 + \epsilon)$-approximate distances are obtained on the collection, we get $(1+\epsilon)(1+1/\text{poly}(n))$ approximate distances on the original graph. Since in our constructions, ϵ is typically $1/\text{polylog}(n) \gg 1/\text{poly}(n)$, the additional distortion in distances due to the reduction is negligible. Hence, our approximation algorithms obtained for graphs with polynomial-ratio can be transformed to algorithms on graphs with general nonnegative weights. Furthermore, hop sets obtained for graphs in the collection can be combined to obtain a hop set for the original graph with comparable attributes to that of hop sets obtained for similar-size graphs with polynomial ratio.

2.1 Pairwise covers

In [5], the author introduced pairwise covers of graphs. Pairwise covers are a modification of neighborhood covers that were introduced by Awerbuch and Peleg [2]. The author presented a sequential algorithm that constructs pairwise covers more efficiently than the neighborhood-covers construction of Awerbuch et al. [1], and a randomized parallel algorithm for constructing a limited version of pairwise covers. The parallel construction is an essential ingredient of our parallel shortest paths algorithm. The following definition specifies the pairwise covers produced by the sequential construction algorithm in [5].

Definition 2.1 Consider a graph $G = (V, E)$ with weights $w : E \leftarrow \mathcal{R}_+$, a scalar $\alpha \in \mathcal{R}_+$ (we use $\beta = \lceil \log_{1+\alpha} n \rceil$), and a scalar $w_r \in \mathcal{R}_+$. A *pairwise* (β, w_r)*-cover* of G is a collection of sets of vertices $X_1, \ldots, X_k$ (*clusters*) and vertices $v_1, \ldots, v_k$, where $v_i \in X_i$ is the *center* of X_i, such that:

i. $\forall \{u, v\} \subset V$ such that $\mathrm{dist}(u, v) \leq w_r$, $\exists i$ such that $\{u, v\} \subset X_i$,

ii. $\forall i$, $\forall u \in X_i$, $\mathrm{dist}(v_i, u) \leq (1 + \log_n m)\beta w_r$, and

iii. $\sum_{i=1}^{k} |X_i| \leq (1 + \alpha)n$.

Remark 2.2 The covers constructed in [5] have the following properties:

i. $\sum_{i=1}^{k} |E \cap X_i^2| \leq m(1 + \alpha)$

ii. Every path of weight at most w_r must be contained in some $X \in \chi$.

iii. For every $v \in V$, at least one cluster $X \in \chi$ contains the $w_r/2$ neighborhood of v.

Theorem 2.3 *[5] A* (β, w_r)*-cover of* G *can be computed in* $O(m(1 + \alpha))$ *time.*

Remark 2.4 Another tradeoff can be obtained by eliminating the constraint on the expansion of the edge sets in the construction of clusters in [5]. Property ii changes to: $\forall i$, $\forall u \in X_i$, $\mathrm{dist}(v_i, u) \leq \beta w_r$, the construction time is $O(m(1 + \alpha)^2)$, and $\sum_{1 \leq i \leq k} |E \cap X_i^2| = m(1 + \alpha)^2$.

Constructing pairwise covers in parallel The following pairwise covers are produced by the randomized parallel cover algorithm in [5]. The main difference from Definition 2.1 is considering ℓ-edge distances. Work-efficient parallelization introduced tradeoffs between the size of paths "covered" and running time.

Definition 2.5 [**Pairwise cover**] Consider a graph $G = (V, E)$ with weights $w : E \leftarrow \mathcal{R}_+$, integers $1 \leq \ell \leq n$ and $\beta = O(\log n)$, and scalars $w_r \in \mathcal{R}_+$ and $0 \leq \delta < 1/2$. A *pairwise* $(\ell, \beta, w_r, \delta)$*-cover* (for brevity, $(\ell, \beta, w_r, \delta)$*-cover*) of G is a collection of sets of vertices $X_1, \ldots, X_k$ (*clusters*) and vertices $v_1, \ldots, v_k$ (where for $1 \leq i \leq k$, $v_i \in X_i$ is the *center* of the cluster X_i) such that

i. $\forall \{u, v\} \subset V$, such that $\mathrm{dist}^{\ell}(u, v) \leq w_r/(1 + \delta)$, $\exists i$ such that $\{u, v\} \subset X_i$,

ii. $\forall i$, $\forall u \in X_i$, $\mathrm{dist}\{v_i, u\} \leq \beta w_r$, and

iii. $\forall v \in V$, $|\{1 \leq i \leq k | v \in X_i\}| = O(n^{1/\beta} \beta \log n)$.

Remark 2.6 The covers constructed in [5] are such that every path of size at most ℓ and weight at most $w_r/(1 + \delta)$ must be contained in some $X \in \chi$.

Corollary 2.7 *It follows that* $\forall e \in E$, $|\{i | 1 \leq i \leq k \wedge e \in X_i^2\}| = O(n^{1/\beta} \beta \log n)$.

Theorem 2.8 *[5] An* $(\ell, \beta, w_r, \delta)$*-cover of* G *can be computed with probability* $1 - O(1/\operatorname{poly}(n))$*, in* $O(\ell \delta^{-1} \beta^2 \log n)$ *time using* $O(n^{1/\beta} m \delta / (\ell \beta))$ *processors.*

2.2 Computing d-edge shortest paths in parallel

We utilize two existing methods for computing, in parallel, single source d-edge shortest paths:

i. A parallel version of the Bellman-Ford shortest paths algorithm runs in $O(d \log n)$ time and performs $O(dm)$ work (see, e.g. [4]).

ii. A parallel approximate weighted BFS algorithm, due to Klein and Sairam [8] (see also [7]), computes d-edge paths of weight within $(1+\epsilon)$ of the minimum weight d-edge paths, in time $O(d\epsilon^{-1} \log n)$ and work $O(m \log n)$.

2.3 Hop sets

Definition 2.9 Consider a weighted graph $G = (V, E)$. A (d, ϵ)*-Hop set* for G is a set of weighted edges E^* such that for all $\{u_1, u_2\} \subset V$,

i. $\text{dist}_{E^*}(u_1, u_2) \geq \text{dist}_E(u_1, u_2)$, and

ii. $\text{dist}^d_{E \cup E^*}(u_1, u_2) \leq (1+\epsilon)\text{dist}_E(u_1, u_2)$.

We refer to d as the *diameter* of the hop set and to $(1+\epsilon)$ as the *approximation quality* of the hop set. We also consider

i. *Restricted hop set* defined as set of weighted edges that satisfies property ii for only the subset of the pairs $V \times V$ that are within some range of distances from each other. We construct hop sets as unions of restricted hop sets.

ii. *Limited hop set* with respect to an integer r, where property ii is relaxed to:

$$\text{dist}^d_{E \cup E^*}(u_1, u_2) \leq (1+\epsilon)\text{dist}^r_E(u_1, u_2) \ .$$

Limited hop sets are needed for parallel constructions. A staightforward parallelization of the sequential construction is efficient in polylogarithmic time only for polylogartihmic values of r. Our parallel construction recursively builds hop sets with small values of r to obtain a hop set with the desired $r = n$.

We link hop sets to parallel computation of shortest paths:

Proposition 2.10 *Consider a weighted graph $G = (V, E)$ and a (d, ϵ)-hop set E^*.*

i. Paths within $(1+\epsilon)$ of shortest from a single source to all other vertices in G can be computed in $O(d \log n)$ time using $O((m + |E^*|)d)$ work.

ii. Paths within $(1+\hat{\epsilon})(1+\epsilon)$ of shortest from a single source to all other vertices can be computed in $O(d\hat{\epsilon}^{-1} \log n)$ time using $O((m + |E^*|) \log n)$ work.

Proof: It suffices to compute d-edge distances in $(V, E \cup E^*)$ (see Subsection 2.2). For the first part we apply the limited parallel Bellman-Ford algorithm. For the second part we apply the parallel weighted BFS algorithm. ∎

3 Computing restricted hop sets

In this Section we present subroutine R-HopSet that computes a restricted hop set for vertex-pairs that are within some specified small range of distances. In Section 4 we obtain a complete hop set by applying R-HopSet with different ranges of weights, and combining the restricted hop sets obtained.

R-HopSet uses the parameters $\rho \geq 9$, $0 < \epsilon_0 < 1$, and $\alpha > 0$. We denote $\beta = \lceil \log_{1+\alpha} n \rceil$. These parameters determine tradeoffs between the running time, the size of the hop set, the approximation quality, and the diameter.

Input R-HopSet is provided with

- a graph (V, E) with edge-weights $w : E \to \mathcal{R}_+$,
- a parameter $0 < \mu \leq 1$
- a scalar $R \in \mathcal{R}_+$

Output R-HopSet constructs a collection of weighted edges E^* such that:

i. If $\epsilon_0 \leq 1/2$:

$$|E^*| \leq (1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} n^{1+\mu}/2 + n\alpha^{-1}((1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} - 1)(3+\alpha)/2 \ .$$

If $\epsilon_0 > 1/2$:

$$|E^*| \leq (1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} n^{1+\mu}/2 + \alpha^{-1}((1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} - 1)n + \sum_{0 \leq i < \lceil \log_{1-\epsilon_0} \mu \rceil} (1+\alpha)^{i+2} n^{1-(1-\epsilon_0)^i (1-2\epsilon_0)}/2 \ .$$

ii. For all $\{u_1, u_2\} \subset V$, $\mathrm{dist}_{E^*}(u_1, u_2) \geq \mathrm{dist}_E(u_1, u_2)$.

iii. For every $\{u_1, u_2\} \subset V$ such that $R \leq \mathrm{dist}_E(u_1, u_2) \leq \rho\beta R$,

$$\mathrm{dist}^{d^*}_{E^* \cup E'}(u_1, u_2) \leq (1 + 21/\rho)^{\lceil \log_{(1-\epsilon_0)} \mu \rceil} \mathrm{dist}_E(u_1, u_2) \ ,$$

where $d^* = (4\beta^2\rho^2 + 6)^{\lceil \log_{(1-\epsilon_0)} \mu \rceil}$, and E' is a collection of edges such that

- for all $\{u_1, u_2\} \subset V$ such that $\mathrm{dist}(u_1, u_2) \leq R/(\beta^2\rho^2)$, we have $\mathrm{dist}^2_{E'}(u_1, u_2) \leq 9\beta \mathrm{dist}(u_1, u_2)$, and
- E' contains all edges in E of weight in $[R/(\rho\beta)^{\lceil \log_{1-\epsilon_0} \mu \rceil}, \rho\beta R]$.

The set E' contains a 2-hop spanner of short distances in E, and a collection of "long" single edges. We can choose (the spanner part of) E' to be much sparser than E, and obtain a *sparse* small-hop approximation of E, when there are not too many "long" single edges. This property allows us to obtain faster aapproximate shortest paths between distant nodes.

Theorem 3.1 R-HopSet *constructs a set E^* as above in time*

$$O(n^\mu (1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} m + \sum_{0 \leq i < \lceil \log_{1-\epsilon_0} \mu \rceil} (1+\alpha)^{i+1} n^{(1-\epsilon_0)^i \epsilon_0} m) \ .$$

In Subsection 3.1 we present R-HopSet. In Subsection 3.2 we show that R-HopSet runs within the time bound stated in Theorem 3.1. In Subsection 3.3 we prove that the size of the set E^* produced by R-HopSet is as specified in property i. In Subsection 3.4 we establish that the set E^* possesses properties ii and iii.

3.1 The R-HopSet algorithm

Algorithm 3.2 [R-HopSet$(\mu, (V, E), R)$]

i. If $\mu \geq 1$:
$E^* \leftarrow V \times V$. For all $e = (u_1, u_2) \in V \times V$, $w(e) \leftarrow \mathrm{dist}_E(u_1, u_2)$. Stop.

ii. Construct a $(\beta, R/(\beta\rho))$-cover χ of (V, E) (as in Definition 2.1).
Let $\chi_s \leftarrow \{X \in \chi | \ |X| \leq |V|^{1-\epsilon_0}\}$, $\chi_b \leftarrow \{X \in \chi | \ |X| > |V|^{1-\epsilon_0}\}$.
Let $\mu' \leftarrow \min\{1, \mu/(1-\epsilon_0)\}$.

iii. For all $X \in \chi_s$:
Place in E^* the edges returned by the recursive call R-HopSet$(\mu', (X, E \cap X^2), R/(\beta^2\rho^2))$.

iv. For each $X \in \chi_b$ and $v \in X$, place in E^* an edge from the center of X to v, weighted $3R/\rho$.

v. Place in E^* a complete graph on the centers of clusters in χ_b. For every $\{X_1, X_2\} \subset \chi_b$, where $c_1 \in X_1$ and $c_2 \in X_2$ are the corresponding centers, $e = (c_1, c_2)$ is assigned the weight $w(e) \leftarrow \mathrm{dist}_E(c_1, c_2)$.

R-HopSet performs recursive calls. For $\mu_0 \leq 1$, denote by $k(\mu_0)$ the depth of the recursion when R-HopSet is called with parameter $\mu = \mu_0$.

Proposition 3.3

$$k(\mu_0) = \lceil \log_{(1-\epsilon_0)} \mu_0 \rceil \ .$$

Proof: If $\mu_0 = 1$, R-HopSet does not perform recursive calls and hence, $k(\mu_0) = 0$. When $\mu_0 < 1$, R-HopSet performs recursive calls at step iii. In these recursive calls, the algorithm is called with parameter $\mu' = \min\{1, \mu_0/(1-\epsilon_0)\}$. Hence, the depth of the recursion is the minimum ℓ such that $\mu_0/(1-\epsilon_0)^\ell \geq 1$. ∎

Each recursive call amounts to applying R-HopSet to a subgraph of the input graph. Consider the recursion tree that corresponds to an application of R-HopSet. The tree has depth $\lceil \log_{1-\epsilon_0} \mu \rceil$. For $0 \leq i \leq \lceil \log_{1-\epsilon_0} \mu \rceil$, denote by $r(i)$ the number of subgraphs R-HopSet is applied to at the ith level of the recursion. Denote these subgraphs by (V_j^i, E_j^i) $(1 \leq j \leq r(i))$.

Proposition 3.4 *i. For all* $1 \leq j \leq r(i)$, $|V_j^i| \leq n^{(1-\epsilon_0)^i}$.

ii. $\sum_j |V_j^i| \leq (1+\alpha)^i n$.

iii. $\sum_j |E_j^i| \leq (1+\alpha)^i m$.

Proof: Immediate from the algorithm, definition 2.1, and Remark 2.2. ∎

3.2 Bounding the time

We bound the running time of R-HopSet. At the bottom level of the recursion ($i = \lceil \log_{1-\epsilon_0} \mu \rceil$), the algorithm performs an all pairs shortest paths computation on each subgraph, and hence runs in time $O(\sum_j |V_j^i||E_j^i|)$. Note that

$$\sum_j |V_j^i||E_j^i| \leq n^\mu \sum_j |E_j^i| \leq n^\mu (1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} m \ .$$

We bound the running time at the ith level of the recursion, for $0 \le i < \lceil \log_{1-\epsilon_0} \mu \rceil$. For $1 \le j \le r(i)$, consider an application of R-HopSet to the subgraph (V_j^i, E_j^i). Theorem 2.3 asserts that Step ii takes $O((1+\alpha)|E_j^i|)$ time. Step iv requires $O(\sum_{X \in \chi_b} |X|) = O((1+\alpha)|V_j^i|)$ time. Step v amounts to single source shortest paths computations in (V_j^i, E_j^i) from at most $|\chi_b| \le (1+\alpha)|V_j^i|/|V_j^i|^{1-\epsilon_0}$ sources. Hence, it takes

$$O((1+\alpha)|V_j^i|^{\epsilon_0}|E_j^i|) = O((1+\alpha)n^{(1-\epsilon_0)^i \epsilon_0}|E_j^i|)$$

time. It follows that level i of the recursion takes

$$O((1+\alpha)^{i+1}(m + n^{(1-\epsilon_0)^i \epsilon_0} m)) = O((1+\alpha)^{i+1} n^{(1-\epsilon_0)^i \epsilon_0} m)$$

time. We obtain that R-HopSet runs in time

$$O(n^{\mu}(1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} m + \sum_{0 \le i < \lceil \log_{1-\epsilon_0} \mu \rceil} (1+\alpha)^{i+1} n^{(1-\epsilon_0)^i \epsilon_0} m) \ .$$

3.3 Bounding the size of the hop set

At the bottom level of the recursion ($i = \lceil \log_{1-\epsilon_0} \mu \rceil$), R-HopSet computes a complete set of edges on each subgraph, and hence produces no more than $\sum_j |V_j^i|^2/2$ edges. The sum is maximized when the subgraphs are as large as possible and hence

$$\sum_j |V_j^i|^2/2 \le ((1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} n/n^{\mu}) n^{2\mu}/2 = (1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} n^{1+\mu}/2 \ .$$

For $0 \le i < \lceil \log_{1-\epsilon_0} \mu \rceil$, we bound the number of edges produced at the ith level of the recursion. For $1 \le j \le r(i)$, consider an application of R-HopSet to the subgraph (V_j^i, E_j^i). Step iv produces at most $(1+\alpha)|V_j^i|$ edges. Step v produces a complete graph on $|\chi_b| \le (1+\alpha)|V_j^i|/|V_j^i|^{1-\epsilon_0}$ vertices and hence produces at most $|\chi_b|^2/2 \le (1+\alpha)^2 |V_j^i|^{2\epsilon_0}/2$ edges.

If $\epsilon_0 \le 1/2$, we have

$$\sum_j |V_j^i|^{2\epsilon_0} \le \sum_j |V_j^i| \le (1+\alpha)^i n \ .$$

It follows that level i of the recursion produces

$$(1+\alpha)^{i+1} n + (1+\alpha)^{i+2} n/2 \le (1+\alpha)^{i+1} n(3+\alpha)/2$$

edges. We obtain that if $\epsilon_0 \le 1/2$, the number of edges produces by R-HopSet is at most

$$(1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} n^{1+\mu}/2 + n\alpha^{-1}((1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} - 1)(3+\alpha)/2 \ .$$

If $\epsilon_0 > 1/2$, we have

$$\sum_j |V_j^i|^{2\epsilon_0} \le ((1+\alpha)^i n/n^{(1-\epsilon_0)^i}) n^{2\epsilon_0(1-\epsilon_0)^i} \le (1+\alpha)^i n^{1-(1-\epsilon_0)^i(1-2\epsilon_0)} \ .$$

Hence, level i of the recursion produces

$$(1+\alpha)^{i+1} n + (1+\alpha)^{i+2} n^{1-(1-\epsilon_0)^i(1-2\epsilon_0)}/2$$

edges. We obtain that if $\epsilon_0 > 1/2$, R-HopSet produces

$$(1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} n^{1+\mu}/2 + \alpha^{-1}((1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} - 1)n + \sum_{0 \le i < \lceil \log_{1-\epsilon_0} \mu \rceil} (1+\alpha)^{i+2} n^{1-(1-\epsilon_0)^i(1-2\epsilon_0)}/2$$

edges.

3.4 Establishing the hop set properties

The following proposition asserts that property ii holds, that is, distances do not decrease as a result of augmenting the graph with E^*.

Proposition 3.5 *For all* $\{u_1, u_2\} \subset V$, $\mathrm{dist}_{E^*}(u_1, u_2) \geq \mathrm{dist}_E(u_1, u_2)$.

Proof: It suffices to show that for every edge $e = (u_1, u_2) \in E^*$, $w(e) \geq \mathrm{dist}_E(u_1, u_2)$. An edge is placed in E^* when a call to R-HopSet is applied to some subgraph (V', E') of (V, E). Consider such an edge $e = (u_1, u_2)$. If e was produced at steps i or v, its assigned weight $w(e)$ corresponds to the weight of some path between u_1 and u_2 in (V', E'), and hence, $w(e) \geq \mathrm{dist}_{E'}(u_1, u_2) \geq \mathrm{dist}_E(u_1, u_2)$. If e was produced at step iv, it follows from Definition 2.1 that $w(e) \geq \mathrm{dist}_{E'}(u_1, u_2) \geq \mathrm{dist}_E(u_1, u_2)$. ∎

It remains to show that property iii holds. Consider an application of R-HopSet. For each cluster $X \in \chi_s$, denote by $E_X \subset X^2$ the edges produced by the recursive call applied at Step iii to X. Let d' and ϵ' be as follows: For all $\{v_1, v_2\} \subset X$ such that

$$R/(\rho^2\beta^2) \leq \mathrm{dist}_{E \cap X^2}(v_1, v_2) \leq R/(\rho\beta) ,$$

$$\mathrm{dist}^{d'}_{E_X \cup E'}(v_1, v_2) \leq (1 + \epsilon')\mathrm{dist}_{E \cap X^2}(v_1, v_2) .$$

Proposition 3.6 *Let* $\{u_1, u_2\} \subset V$ *be two vertices such that* $R \leq \mathrm{dist}_E(u_1, u_2) \leq \rho\beta R$. *There exists a path* $p^* \subset E^* \cup E'$ *between* u_1 *and* u_2 *of size*

$$|p^*| \leq (2d' + 2)\beta^2\rho^2 + d' + 5$$

and weight

$$w(p^*) \leq (1 + \epsilon')(1 + 21/\rho)\mathrm{dist}_E(u_1, u_2) .$$

Proof: Consider a path $p \subset E$ such that $R \leq w(p) \leq \rho\beta R$. We explicitly construct a path p^* such that $|p^*| \leq (2d' + 2)\beta^2\rho^2 + d' + 5$ and $w(p^*) \leq (1 + \epsilon')(1 + 21/\rho)w(p)$. We partition p to segments, where each segment is either a single edge of weight more than $R/(\beta\rho)$ or a subpath of weight at most $R/(\beta\rho)$. We consider a partition created by traversing the path p and cutting maximal eligible segments (every segment is such that appending the next p edge in the traversal would cause the weight of the segment to exceed $R/(\beta\rho)$.)

Remark 2.2 asserts that every segment of weight at most $R/(\beta\rho)$ is contained in at least one cluster in χ. Consider the minimal subpath p' of p that contains all the segments of p that are of weight at most $R/(\beta\rho)$ and are not fully contained in χ_s clusters. Denote the endpoints of p' by v_1 and v_2. It follows that $\{v_1, v_2\} \subset \bigcup_{X \in \chi_b} X$. Let $X_1 \in \chi_b$ and $X_2 \in \chi_b$ be such that $v_1 \in X_1$ and $v_2 \in X_2$. Consider the path $p'^* \subset E^*$ composed of an edge between v_1 and the center of X_1, an edge e' between the centers of X_1 and X_2 (if $X_1 \neq X_2$), and an edge between the center of X_2 and v_2. The first and last edges were generated at step iv and each has weight $3R/\rho$. The edge e' is created at step v. It follows from Definition 2.1 part ii and from the triangle inequality that $w(e') \leq w(p') + 6R/\rho$. Therefore, $w(p'^*) \leq w(p') + 12R/\rho$. Note that $|p'^*| \leq 3$.

Consider segments of p of weight at most $R/(\beta^2\rho^2)$. Each such segment $\hat{p}$ is either the last segment of p or followed by an edge of weight larger than $R/(\beta\rho) - w(\hat{p})$. Hence, there are at most $1 + \lfloor w(p)(\beta\rho)/R \rfloor$ such segments. It follows that the combined weight of these segments is at most $R/(\beta^2\rho^2) + w(p)/(\beta\rho)$. Note that each such segment $\hat{p}$ has a corresponding path in E' between the endpoints of $\hat{p}$ of size at most 2 and weight at most $9\beta w(\hat{p})$. For each segment of weight at most $R/(\beta^2\rho^2)$ that is not contained in p', place corresponding E' paths in p^*. It follows that the increase in the size of p^* is at most $2 + 2\lfloor w(p)(\beta\rho)/R \rfloor$, and the sum of the weights of the new p^* edges is larger by at most $(9\beta - 1)(R/(\beta^2\rho^2) + w(p)/(\beta\rho)) \leq 9w(p)/\rho$ than the combined weight of the corresponding p segments.

Consider segments of p of weight at least $R/(\beta^2\rho^2)$ that are not contained in p'. Every two consecutive segments in the partition have weight greater than $R/(\beta\rho)$. Thus, there are at most $\lceil 2w(p)(\beta\rho)/R \rceil$ such segments. For each such a segment $\hat{p}$ do as follows: If $w(\hat{p}) \geq R/(\beta\rho)$, then $\hat{p}$ is a single edge in $e \in E$ (by definition of E', we have $e \in E'$ as well). Place e in p^*. Otherwise, $R/(\beta^2\rho^2) \leq w(\hat{p}) \leq R/(\beta\rho)$. It follows that $\hat{p}$ is contained in some cluster $X \in \chi_s$. Therefore, it follows from the assumptions that there is a path $\hat{p}^* \subset E_X \cup E'$ between the endpoints of $\hat{p}$ such that $|\hat{p}^*| \leq d'$ and $w(\hat{p}^*) \leq w(\hat{p})(1+\epsilon')$. Place this path in p^*. We obtain that the size of p^* increased by at most $d'\lceil 2w(p)(\beta\rho)/R \rceil$, and the weight of the new p^* edges is at most $(1+\epsilon')$ times the combined weight of the corresponding segments.

It is easy to verify that p^* constructed above is contained in $E^* \cup E'$ and constitutes a path between the endpoints of p. It follows that

$$|p^*| \leq d'\lceil 2w(p)(\beta\rho)/R \rceil + 3 + 2 + 2\lfloor w(p)(\beta\rho)/R \rfloor \leq (2d'+2)\beta^2\rho^2 + d' + 5$$

and

$$w(p^*) \leq w(p) + \epsilon' w(p) + 12R/\rho + 9w(p)/\rho \leq w(p)(1+\epsilon'+21/\rho) \leq w(p)(1+\epsilon')(1+21/\rho) \ .$$

∎

The following corollary establishes that property iii holds.

Corollary 3.7 *For every pair of vertices $\{u_1, u_2\} \subset V$ such that $R \leq \mathrm{dist}_E(u_1,u_2) \leq \rho\beta R$*

$$\mathrm{dist}^{d^*}_{E^*\cup E'}(u_1,u_2) \leq (1+21/\rho)^{\lceil \log_{(1-\epsilon_0)}\mu \rceil} \mathrm{dist}_E(u_1,u_2) \ ,$$

where $d^* = (4\beta^2\rho^2+6)^{\lceil \log_{(1-\epsilon_0)}\mu \rceil}$.

Proof: The proof is by induction on the depth of the recursion $k(\mu)$. If $k(\mu) = 0$, $E^* = V \times V$ and the assigned weight to $e \in E^*$ is the corresponding distance. Hence, for every pair $\{u_1, u_2\} \subset V$, $\mathrm{dist}^1_{E^*}(u_1,u_2) = \mathrm{dist}_E(u_1,u_2)$. Consider an application of the algorithm where $k(\mu) > 0$. For each cluster $X \in \chi_s$, denote by $E_X \subset X^2$ the edges produced by the recursive call applied at Step iii to X. The induction hypothesis asserts that for $d' = (4\beta^2\rho^2+6)^{k(\mu')}$ and $\epsilon' = (1+21/\rho)^{k(\mu')} - 1$, for all $\{v_1, v_2\} \subset X$ such that

$$R/(\rho^2\beta^2) \leq \mathrm{dist}_{E\cap X^2}(v_1,v_2) \leq R/(\rho\beta) \ ,$$

$$\mathrm{dist}^{d'}_{E_X\cup E'}(v_1,v_2) \leq (1+\epsilon')\mathrm{dist}_{E\cap X^2}(v_1,v_2) \ .$$

Therefore, it follows from Proposition 3.6 that there exists a path $p^* \subset E^* \cup E'$ between u_1 and u_2 of size

$$|p^*| \leq (2d'+2)\beta^2\rho^2 + d' + 5 \leq (4\beta^2\rho^2+6)d' \leq (4\beta^2\rho^2+6)^{k(\mu')+1} = (4\beta^2\rho^2+6)^{k(\mu)}$$

and weight

$$w(p^*) \leq (1+21/\rho)^{k(\mu')}(1+21/\rho)\mathrm{dist}_E(u_1,u_2) \leq (1+21/\rho)^{k(\mu)}\mathrm{dist}_E(u_1,u_2) \ .$$

∎

4 Computing a complete hop set

In this section we present algorithm HopSet that computes a hop set for all distances. HopSet employs R-HopSet (Algorithm 3.2) as a subroutine.

4.1 Properties of HopSet

Input HopSet is provided with:

- A graph (V, E) with edge-weights $w : E \leftarrow \mathcal{R}_+$,
- Parameters:
  - $\rho \geq 6$ determines tradeoffs between approximation and diameter.
  - $\alpha > 0$ ($\beta = \lceil \log_{1+\alpha} n \rceil$) that determines tradeoffs between the diameter on one hand and the time and size of the hop set on the other hand.
  - $0 < \epsilon_0 < 1$
  - $0 < \mu \leq 1$

Output HopSet constructs a collection of weighted edges E^* such that

i. If $\epsilon_0 \leq 1/2$:

$$|E^*| \leq \lceil \log_2(nw_{\max}/w_{\min}) \rceil (1+\alpha)n +$$
$$\lceil \log_{\beta\rho}(nw_{\max}/w_{\min}) \rceil \left((1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} n^{1+\mu}/2 + n\alpha^{-1}((1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} - 1)(3+\alpha)/2 \right) .$$

If $\epsilon_0 > 1/2$:

$$|E^*| \leq \lceil \log_2(nw_{\max}/w_{\min}) \rceil (1+\alpha)n +$$
$$\lceil \log_{\beta\rho}(nw_{\max}/w_{\min}) \rceil ((1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} n^{1+\mu}/2 + \alpha^{-1}((1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} - 1)n +$$
$$\sum_{0 \leq i < \lceil \log_{1-\epsilon_0} \mu \rceil} (1+\alpha)^{i+2} n^{1-(1-\epsilon_0)^i(1-2\epsilon_0)}/2) .$$

ii. for all $\{u_1, u_2\} \subset V$, $\mathrm{dist}_{E^*}(u_1, u_2) \geq \mathrm{dist}_E(u_1, u_2)$, and

iii. for every $\{u_1, u_2\} \subset V$,

$$\mathrm{dist}^{d^*}_{E^* \cup E}(u_1, u_2) \leq (1 + 21/\rho)^{\lceil \log_{(1-\epsilon_0)} \mu \rceil} \mathrm{dist}_E(u_1, u_2) ,$$

where $d^* = (4\beta^2\rho^2 + 6)^{\lceil \log_{(1-\epsilon_0)} \mu \rceil}$.

Theorem 4.1 HopSet *runs in time:*

$$O(\lceil \log_2(nw_{\max}/w_{\min}) \rceil (1+\alpha)m +$$
$$\lceil \log_{\beta\rho}(nw_{\max}/w_{\min}) \rceil (n^{\mu}(1+\alpha)^{\lceil \log_{1-\epsilon_0} \mu \rceil} m + \sum_{0 \leq i < \lceil \log_{1-\epsilon_0} \mu \rceil} (1+\alpha)^{i+1} n^{(1-\epsilon_0)^i \epsilon_0} m)) .$$

Table 1 derives bounds for the running time of HopSet and the attributes of the output hop set, for some assignments of values to the parameters. Note that we assume $w_{\max}/w_{\min} = O(\mathrm{poly}\, n)$.

time	$\lvert E^*\rvert$	d^*	*approximation*
Assign: $\alpha = 1$ (hence, $\beta = O(\log n)$), ϵ_0 and μ are fixed constants			
$O((n^{\mu} + n^{\epsilon_0})m \log n)$	$O((n^{1+\mu} + n^{2\epsilon_0}) \log n)$	$O(\rho \log n)^{2\lceil \log_{1-\epsilon_0} \mu \rceil}$	$(1 + O(\rho^{-1}))$
Assign: $1/2 \geq \epsilon_0 = \Omega(1/\log\log n)$, $\mu = \Omega(1/\log n)$, $\alpha = 1/\lceil \log_{1-\epsilon_0} \mu \rceil \leq 1$ ($\beta = O(\epsilon_0^{-1} \log \mu^{-1} \log n)$)			
$O((n^{\epsilon_0} + n^{\mu})m \log n)$	$O((n^{1+\mu} + \epsilon_0^{-1} n \log \mu^{-1}) \log n)$	$(\rho \log n)^{O(\epsilon_0^{-1} \log \mu^{-1})}$	$(1 + 7/\rho)^{\lceil \log_{1-\epsilon_0} \mu \rceil}$
Assign: the above, $\mu = 1/\log n$, and $\epsilon_0 = 1/\log\log n$			
$O(mn^{1/\log\log n} \log n)$	$O(n \log n (\log\log n)^2)$	$(\rho \log n)^{O((\log\log n)^2)}$	$(1 + \rho^{-1})^{O((\log\log n)^2)}$
Better hop sets but worse time bounds:			
Assign: $\epsilon_0 = 1/2$, $\mu < 1/2$, $\alpha = 1/\lceil \log_2 \mu^{-1} \rceil \leq 1$ ($\beta = O(\alpha^{-1} \log n) = O(\log \mu^{-1} \log n)$)			
$O(mn^{0.5} \log n)$	$O((n^{1+\mu} + n \log \mu^{-1}) \log n)$	$(\rho \log n \log \mu^{-1})^{O(\log \mu^{-1})}$	$(1 + 7/\rho)^{\lceil \log \mu^{-1} \rceil}$
Assign: the above and $\mu = \log^{-1} n$			
$O(mn^{0.5} \log n)$	$O(n \log n \log\log n)$	$(\rho \log n)^{O(\log\log n)}$	$(1 + 7/\rho)^{\lceil \log_2 \log n \rceil}$

Table 1: Bounds for hop sets constructions by various parameter assignments

4.2 The HopSet algorithm

The complexity and correctness as claimed in Subsection 4.1 are immediate from the properties of R-HopSet (Algorithm 3.2).

Algorithm 4.2 [HopSet$(\mu, (V, E))$]

i. Compute a (9β)-spanner of diameter 2 as follows and place the spanner in E^*:
For $i = 1, \ldots, \lceil \log_{1.5}(nw_{\max}/w_{\min}) \rceil$ do:

(1) $w_i \leftarrow w_{\min} 1.5^i$

(2) Construct a (β, w_i)-cover χ of (V, E) (as in Definition 2.1).

(3) For each $X \in \chi$ and $v \in X$, place in E^* an edge from the center of X to v, weighted $3\beta w_i$

ii. For $i = 0, \ldots, \lfloor \log_{\rho\beta}(nw_{\max}/w_{\min}) \rfloor$ do:

(1) $w_i \leftarrow w_{\min} \rho^i \beta^i$.

(2) Place in E^* the edges returned by R-HopSet$(\mu, (V, E), w_i)$
(restricted hop set for distances in $[w_i, w_i \rho\beta]$)

Remark 4.3 If in Algorithm 3.2 we use the cover construction mentioned in Remark 2.4 instead, then step iv can be modified such that the edges are assigned weights R/ρ instead of $3R/\rho$. Suppose that in HopSet (Algorithm 4.2) we use the cover construction of Remark 2.4 in step i, where weights of βw_i are assigned in step i3 (hence, the result is a (3β)-spanner), and we use the altered Algorithm 3.2 as a subroutine. We obtain a collection E^*, where property iii changes to:

$$\mathrm{dist}_{E^* \cup E}^{d^*}(u_1, u_2) \leq (1 + 7/\rho)^{\lceil \log_{(1-\epsilon_0)} \mu \rceil} \mathrm{dist}_E(u_1, u_2) \ ,$$

in time

$$O(\lceil \log_2(nw_{\max}/w_{\min}) \rceil (1 + \alpha)^2 m +$$
$$\lceil \log_{\beta\rho}(nw_{\max}/w_{\min}) \rceil (n^{\mu} (1 + \alpha)^{2\lceil \log_{1-\epsilon_0} \mu \rceil} m + \sum_{0 \leq i < \lceil \log_{1-\epsilon_0} \mu \rceil} (1 + \alpha)^{2i+1} n^{(1-\epsilon_0)^i \epsilon_0} m)) \ .$$

4.3 Faster approximate shortest paths algorithm for distant pairs

The best known sequential time bound for computing shortest paths from s sources is $\tilde{O}(sm)$. Our algorithm can be modified to compute a sparse set of edges E^*, such that distances in E^* approximate long distances in E. For dense graphs, $|E^*| \ll |E|$. Therefore, shortest paths computations on E^*, which take time $O(s|E^*|) \ll O(s|E|)$, yield good approximations for distances in E between vertex-pairs that are far apart.

We outline a modification of Algorithm 4.2 that produces such a sparse set. Consider a weighted graph G and a parameter $R > w_{\min}$. Consider applying to G a modified Algorithm 4.2 where step ii is performed for $i = 0, \ldots, \lfloor \log_{\rho\beta}(nw_{\max}/R) \rfloor$, and we let $w_i \leftarrow R\rho^i\beta^i$. It is easy to verify the following:

Proposition 4.4 *The set E^* produced by the modified algorithm possesses the properties specified in the output of Algorithm 4.2 (see Subsection 4.1) except property iii changes to: for every $\{u_1, u_2\} \subset V$ such that* $\mathrm{dist}_E(u_1, u_2) \geq R$*,*

$$\mathrm{dist}^{d^*}_{E^* \cup E'}(u_1, u_2) \leq (1 + 21/\rho)^{\lceil \log_{(1-\epsilon_0)} \mu \rceil} \mathrm{dist}_E(u_1, u_2) \ ,$$

where

$$E' = \{e \in E | w(e) \geq R/(\rho\beta)^{\lceil \log_{1-\epsilon_0} \mu \rceil}\} \ .$$

Corollary 4.5 *If $R > w_{\max}(\rho\beta)^{\lceil \log_{1-\epsilon_0} \mu \rceil}$,*

$$\mathrm{dist}^{d^*}_{E^*}(u_1, u_2) \leq (1 + 21/\rho)^{\lceil \log_{(1-\epsilon_0)} \mu \rceil} \mathrm{dist}_E(u_1, u_2) \ .$$

Proof: Note that $E' = \emptyset$. ∎

Using one of the tradeoffs in Table 1, we obtain the following:

Proposition 4.6 *Let $S \subset V$, where $s = |S|$ be a subset of vertices. For any fixed $\epsilon_0 \leq 1/2$ and $\mu > 0$, in $O(((n^\mu + n^{\epsilon_0})m + sn^{1+\mu}) \log n)$ time, we can compute for every vertex-pair $(u_1, u_2) \in S \times V$, a path $p_{u_1u_2}$ such that: If*

$$\mathrm{dist}_E(u_1, u_2) \geq w_{\max}(\rho \log n)^{\lceil \log_{1-\epsilon_0} \mu \rceil} \ ,$$

$$w(p_{u_1u_2}) \leq (1 + O(\rho^{-1}))\mathrm{dist}_E(u_1, u_2) \ .$$

Otherwise,

$$w(p_{u_1u_2}) \leq w_{\max}(\rho \log n)^{\lceil \log_{1-\epsilon_0} \mu \rceil} \ .$$

Remark 4.7 The key idea in avoiding $O(m)$ work per source is producing a sparse "encoding" of the graph, that is, a sparse set of edges that is almost as good as the original denser set of edges for shortest paths computations between distant pairs of vertices. Unfortunately, short distances can not be "encoded" in a similar fashion. This is demonstrated by graphs with unit weights. Note that omitting any edge results in at least doubling the distance between its endpoints. Hence, all edges are essential and a sparse encoding is not feasible.

4.4 A different hop set algorithm

We present algorithm HopSet2 that yields different tradeoffs. The correctness and complexity analysis are omitted since they are along the lines of the correctness and complexity analysis of Algorithm 3.2 and of the parallel algorithm presented in subsequent sections.

Output Assume that ϵ_0 and μ are constants, and $\alpha = 1$. The algorithm constructs a collection of weighted edges E^* such that

i.
$$|E^*| = O(n^{1+\mu}\log^{1+\lceil \log_{1-\epsilon_0}\mu\rceil} n + n^{2\epsilon_0}\log n) \ .$$

ii. for all $\{u_1, u_2\} \subset V$, $\text{dist}_{E^*}(u_1, u_2) \geq \text{dist}_E(u_1, u_2)$, and

iii. for every $\{u_1, u_2\} \subset V$
$$\text{dist}^{d^*}_{E^* \cup E}(u_1, u_2) = (1 + O(\rho^{-1})\text{dist}_E(u_1, u_2) \ ,$$
where $d^* = O(\rho\log n)^{\lceil \log_{(1-\epsilon_0)}\mu\rceil}$.

The algorithm runs in time:
$$O(mn^{\mu}\log^{1+\lceil \log_{1-\epsilon_0}\mu\rceil} n + mn^{\epsilon_0}\log n) \ .$$

Algorithm 4.8 [HopSet2$(\mu, (V, E))$]

i. If $\mu \geq 1$:
$E^* \leftarrow V \times V$. For all $e = (u_1, u_2) \in V \times V$, $w(e) \leftarrow \text{dist}_E(u_1, u_2)$. Stop.

ii. For $i = 1, \ldots, \lceil \log(nw_{\max}/w_{\min})\rceil$ do:

(1) $w_i \leftarrow w_{\min}2^i$. (compute a restricted hop set for distances in $[w_i/2, w_i]$)

(2) Construct a $(\beta, w_i/(\beta\rho))$-cover χ of (V, E) (as in Definition 2.1).
Let $\chi_s \leftarrow \{X \in \chi | |X| \leq n^{1-\epsilon_0}\}$, $\chi_b \leftarrow \{X \in \chi | |X| > n^{1-\epsilon_0}\}$.
Let $\mu' \leftarrow \min\{1, \mu/(1-\epsilon_0)\}$.

(3) For all $X \in \chi_s$:
Place in E^* the edges returned by the recursive call HopSet2$(\mu', (X, E \cap X^2))$.

(4) For each $X \in \chi_b$ and $v \in X$, place in E^* an edge from the center of X to v, weighted $3w_i/\rho$.

(5) Place in E^* a complete graph on the centers of clusters in χ_b. For every $\{X_1, X_2\} \subset \chi_b$, where $c_1 \in X_1$ and $c_2 \in X_2$ are the corresponding centers, the edge $e = (c_1, c_2)$ is assigned the weight $w(e) \leftarrow \text{dist}_E(c_1, c_2)$.

5 The Limited-HopSet algorithm

We present algorithm Limited-HopSet that inputs a weighted graph (V, E), a scalar $R \in \mathcal{R}_+$, and an integer d. Limited-HopSet produces a restricted limited hop set E^* such that d-edge distances in (V, E) of weight at most R, are approximated by d^*-edge distances in $(V, E^* \cup E)$. Limited-HopSet is a parallel version of Algorithm 4.8 that applies only for d-edge distances. Note that d^* is independent of our choice of d. A work-efficient parallel implementation of Limited-HopSet requires time linear in d. Therefore, Limited-HopSet can not be applied directly to obtain a hop set for n-edge distances in polylog time. The latter is achieved in a subsequent section by algorithm Full-HopSet that uses Limited-HopSet as a subroutine. Roughly, Full-HopSet sets $d \leftarrow 2d^*$ and calls Limited-HopSet iteratively. In each iteration, Limited-HopSet is applied to the input graph augmented with the hop sets produced in previous iterations.

Limited-HopSet uses the following parameters:

i. $\rho \gg 4$ that determines tradeoffs between accuracy and time.

ii. β (we set $\beta = \lceil \log n \rceil$ and hence $n^{1/\beta} = O(1)$) used for pairwise covers (see Definition 2.5).

iii. δ (we set $\delta = 0.1$) used for pairwise covers (see Definition 2.5).

iv. $\epsilon_0 < 1$, $\epsilon_0 = \Omega(1/\log\log n)$.

v. $\hat{\epsilon} > 0$ determines the accuracy in the limited single-source shortest-paths computations performed by the algorithm. (see Subsection 2.2).

Input Limited-HopSet is provided with:

- A graph (V, E) with edge-weights $w : E \leftarrow \mathcal{R}_+$.
- A parameter $\mu \leq 1$, $\mu = \Omega(1/\log\log n)$.
- A scalar $R \in \mathcal{R}_+$.
- An integer d.

Output Limited-HopSet constructs a collection of weighted edges E^* such that:

i. $|E^*| = O((C\lceil\log(R/w_{\min})\rceil \log^2 n)^{\lceil \log_{1-\epsilon_0} \mu \rceil} n^{1+\mu} + n^{2\epsilon_0}\lceil\log(R/w_{\min})\rceil \log^4 n)$, for some fixed constant C.

ii. For all $\{u_1, u_2\} \subset V$, $\mathrm{dist}_{E^*}(u_1, u_2) \geq \mathrm{dist}_E(u_1, u_2)$.

iii. For every $\{u_1, u_2\} \subset V$ such that $\mathrm{dist}^d_E(u_1, u_2) \leq R$, we have

$$\mathrm{dist}^{d^*}_{E^* \cup E}(u_1, u_2) \leq (1 + \hat{\epsilon})(1 + 4/\rho)^{\lceil \log_{(1-\epsilon_0)} \mu \rceil} \mathrm{dist}^d_E(u_1, u_2) \ ,$$

where $d^* = (4\beta\rho(1 + \delta) + 3)^{\lceil \log_{(1-\epsilon_0)} \mu \rceil}$.

Algorithm 5.1 [Limited-HopSet$(\mu, (V, E), R, d)$]

i. If $\mu \geq 1$:
$E^* \leftarrow V \times V$. The edge-weights assigned are corresponding $(1 + \hat{\epsilon})$-approximate d-edge distances in (V, E). Stop.

ii. For $i = 1, \ldots, \lceil \log(R/w_{\min}) \rceil$ do in parallel:

(1) $w_i \leftarrow w_{\min} 2^i$. (shortcut distances in $[w_i/2, w_i]$)

(2) Construct a $(d, \beta, w_i/(2\beta\rho), \delta)$-cover χ of (V, E). (As in Definition 2.5).
Let $\chi_s \leftarrow \{X \in \chi | |X| \leq |V|^{1-\epsilon_0}\}$, $\chi_b \leftarrow \{X \in \chi | |X| > |V|^{1-\epsilon_0}\}$.
Let $\mu' \leftarrow \min\{1, \mu/(1 - \epsilon_0)\}$.

(3) For all $X \in \chi_s$:
Place in E^* the edges returned by the recursive call
Limited-HopSet$(\mu', (X, E \cap X^2), w_i/(2\beta\rho), d)$.

(4) For each $X \in \chi_b$ and $v \in X$, place in E^* an edge from the center of X to v, weighted $w_i/(2\rho)$.

(5) We place in E^* a complete graph on the centers of clusters in χ_b, where the edges are weighted as follows:
For each cluster $X \in \chi_b$, consider the graph formed by contracting X. Perform a $(1+\hat{\epsilon})$-approximate d-edge single source shortest path computation in it rooted at the contracted X. For each cluster $X' \in \chi_b$, denote by $a(X,X')$ the minimum weight of a computed path between the contracted X and a vertex in X'. The edge in E^* between the centers of X and X' is assigned weight $a(X,X') + w_i/\rho$.

We utilize either one of two algorithms for the d-edge single-source shortest-paths computations performed at Step i and Step ii5 (see Subsection 2.2). (i) The limited parallel Bellman-Ford algorithm that computes exact d-edge shortest paths, and (ii) the parallel (approximate) weighted BFS algorithm that computes d-edge paths that are within $(1+\hat{\epsilon})$ of the shortest d-edge paths.

Correctness In Section 7 we establish that Algorithm 5.1 is correct, that is, the set E^* produced by the algorithm possesses the specified properties.

Complexity In Section 6 we prove that Limited-HopSet can be implemented with the following bounds on the resources. If the weighted BFS algorithm is utilized, Limited-HopSet performs

$$O(mn^{\epsilon_0}\log^3 n\lceil\log(R/w_{\min})\rceil + (D\lceil\log(R/w_{\min})\rceil\log^2 n)^{\lceil\log_{1-\epsilon_0}\mu\rceil} mn^{\mu}\log n)$$

work and runs in

$$O(\epsilon_0^{-1}\hat{\epsilon}^{-1}d\log^3 n)$$

time. If the limited Bellman-Ford algorithm is utilized, Limited-HopSet performs

$$O(dmn^{\epsilon_0}\log^2 n\lceil\log(R/w_{\min})\rceil + (D\lceil\log(R/w_{\min})\rceil\log^2 n)^{\lceil\log_{1-\epsilon_0}\mu\rceil} dmn^{\mu})$$

work and runs in time

$$O(d\epsilon_0^{-1}\log^3 n)\ .$$

6 Complexity of Limited-HopSet

In this Section we bound the resources (time and work) used by Algorithm 5.1. We analyze the bounds when either the limited Bellman-Ford algorithm or the parallel (approximate) weighted BFS algorithm (see Subsection 2.2) are utilized for the d-edge single source shortest paths computations performed at Step i and Step ii5.

For $\mu_0 \le 1$, denote by $k(\mu_0)$ the depth of the recursion when Algorithm 5.1 is called with parameter $\mu = \mu_0$.

Proposition 6.1

$$k(\mu_0) = \lceil\log_{(1-\epsilon_0)}\mu_0\rceil\ .$$

Proof: Identical to the argument in the proof of Proposition 3.3. ∎

Consider the recursion tree of a run of the algorithm. The root of the recursion tree (top level) is a single instance applied to the input graph. Each node corresponds to an instance of the algorithm applied to a subgraph of the input graph. The children of a node correspond to the subgraphs the recursive calls are applied to. Each level of the tree consists of a collection of instances. For $0 \le i \le \lceil\log_{1-\epsilon_0}\mu\rceil$, denote by (V^i_j, E^i_j) $(1 \le j \le r(i))$ the collection of subgraphs constituting the instances at the ith level of the recursion.

Proposition 6.2 *i. For all* $1 \le j \le r(i)$, $|V_j^i| \le n^{(1-\epsilon_0)^i}$.

ii. $\sum_j |V_j^i| = O(\lceil \log(R/w_{\min}) \rceil \log^2 n)^i n$.

iii. $\sum_j |E_j^i| = O(\lceil \log(R/w_{\min}) \rceil \log^2 n)^i m$.

Proof: Immediate from the algorithm, Definition 2.5, and Corollary 2.7. ∎

6.1 Bounding the work

We bound the work performed by Limited-HopSet when either the limited Bellman-Ford algorithm or the parallel weighted BFS algorithm (see Subsection 2.2) are utilized.

Proposition 6.3 *Suppose that the parallel weighted BFS algorithm is utilized by Algorithm 5.1. There exist a constant D such that the work performed is*

$$O(mn^{\epsilon_0} \log^3 n \lceil \log(R/w_{\min}) \rceil + (D \lceil \log(R/w_{\min}) \rceil \log^2 n)^{\lceil \log_{1-\epsilon_0} \mu \rceil} mn^{\mu} \log n) \ .$$

Proof: The work performed at the bottom level of the recursion tree ($i = \lceil \log_{1-\epsilon_0} \mu \rceil$) is dominated by d-edge shortest paths computations in each of the subgraphs and hence is bounded by

$$O(\sum_{1 \le j \le r(i)} |V_j^i||E_j^i| \log n) = O(n^{(1-\epsilon_0)^i} \sum_{1 \le j \le r(i)} |E_j^i| \log n) = O(\lceil \log(R/w_{\min}) \rceil \log^2 n)^i n^{\mu} m \log n \ .$$

We bound the work at level i of the recursion, where $0 \le i < \lceil \log_{1-\epsilon_0} \mu \rceil$. For $1 \le j \le r(i)$, consider a subgraph (V_j^i, E_j^i). Theorem 2.8 asserts that Step ii2 performs $O(|E_j^i|(1-\epsilon_0)^{2i} \log^2 n)$ work. Step ii4 performs $O(|V_j^i|(1-\epsilon_0)^{2i} \log^2 n)$ work. Step ii5 performs

$$O(|V_j^i|^{\epsilon_0} |E_j^i| (1-\epsilon_0)^{3i} \log^3 n)$$

work, since $|\chi_b| = O(|V_j^i|^{\epsilon_0} \log^2 |V_j^i|)$. Therefore, each iteration of step ii performs

$$O(|V_j^i|^{\epsilon_0} |E_j^i| (1-\epsilon_0)^{3i} \log^3 n)$$

work. There are $\lceil \log(R/w_{\min}) \rceil$ iteration. It follows that level i of the recursion performs

$$\sum_{1 \le j \le r(i)} O(|V_j^i|^{\epsilon_0} |E_j^i| (1-\epsilon_0)^{3i} \log^3 n \lceil \log(R/w_{\min}) \rceil) =$$

$$O(n^{(1-\epsilon_0)^i \epsilon_0} (1-\epsilon_0)^{3i} \log^3 n \lceil \log(R/w_{\min}) \rceil \sum_{1 \le j \le r(i)} |E_j^i|) =$$

$$O(\lceil \log(R/w_{\min}) \rceil \log^2 n)^{i+1} mn^{(1-\epsilon_0)^i \epsilon_0} (1-\epsilon_0)^{3i} \log n$$

work. Using the assumption $\epsilon_0 = \Omega(1/\log\log n)$, we obtain that the algorithm performs

$$O(mn^{\epsilon_0} \log^3 n \lceil \log(R/w_{\min}) \rceil + O(\lceil \log(R/w_{\min}) \rceil \log^2 n)^{\lceil \log_{1-\epsilon_0} \mu \rceil} mn^{\mu} \log n)$$

work. ∎

Proposition 6.4 *Suppose that the parallel limited Bellman-Ford algorithm is utilized by Algorithm 5.1. There exist a constant D such that the work performed is*

$$O(dmn^{\epsilon_0} \log^2 n \lceil \log(R/w_{\min}) \rceil + (D \lceil \log(R/w_{\min}) \rceil \log^2 n)^{\lceil \log_{1-\epsilon_0} \mu \rceil} dmn^{\mu}) \ .$$

Proof: The time bounds are dominated by the single source shortest paths computations. The parallel Bellman-Ford algorithm performs a factor of $O(d/\log n)$ more work than the parallel weighted BFS algorithm. ∎

6.2 Bounding the time

We bound the number of parallel steps needed when either the limited Bellman-Ford or the parallel (approximate) weighted BFS algorithms (see Subsection 2.2) are utilized.

Proposition 6.5 *When the parallel weighted BFS algorithm is used, the running time is bounded by*

$$O(\epsilon_0^{-1}\hat{\epsilon}^{-1}d\log^3 n) \ .$$

Proof: Note that all recursive calls at a certain level of the recursion can be performed in parallel. Hence, to bound the time of level i, it suffices to bound the time of a single call on a subgraph. At the bottom level of the recursion ($i = \lceil \log_{1-\epsilon_0}\mu \rceil$), the time is dominated by performing d-edge single-source shortest-paths computations on graphs of size $O(n^\mu)$. Hence, it takes $O(d\hat{\epsilon}^{-1}\mu\log n)$ time. Consider level i of the recursion. Consider an application of the algorithm to a subgraph (V_j^i, E_j^i). The iterations at Step ii can be performed in parallel, and hence, it suffices to bound the time of a single iteration. Theorem 2.8 asserts that Step ii2 takes

$$O(d\hat{\epsilon}^{-1}\log^3|V_j^i|) = O(d\hat{\epsilon}^{-1}(1-\epsilon_0)^{3i}\log^3 n)$$

time. Steps ii3 and ii4 take $O(\log|V_j^i|) = O((1-\epsilon_0)^i\log n)$ time. Step ii5 amounts to performing (in parallel) d-edge shortest paths computations on (V_j^i, E_j^i) and hence take

$$O(d\hat{\epsilon}^{-1}\log|V_j^i|) = O(d\hat{\epsilon}^{-1}(1-\epsilon_0)^i\log n)$$

time. The proof follows. ∎

A very similar analysis yields the following:

Proposition 6.6 *When the limited Bellman-Ford algorithm is used, the running time is bounded by $O(d\epsilon_0^{-1}\log^3 n)$.*

7 Correctness of Limited-HopSet

In this section we show that the set E^* constructed by Algorithm 5.1 possesses the claimed properties.

7.1 Bounding the size of the hop set

We bound the size of the set E^* produced by Algorithm 5.1:

Proposition 7.1 *There exists a constant C such that*

$$|E^*| = O((C\lceil\log(R/w_{\min})\rceil\log^2 n)^{\lceil\log_{1-\epsilon_0}\mu\rceil}n^{1+\mu} + n^{2\epsilon_0}\lceil\log(R/w_{\min})\rceil\log^4 n) \ .$$

Proof: We bound the number of edges produced at each level of the recursion. Consider the bottom level of the recursion ($i = \lceil\log_{1-\epsilon_0}\mu\rceil$). For each subgraph (V_j^i, E_j^i), step i produces $|V_j^i|(|V_j^i| - 1)/2 \le |V_j^i|^2$ edges. Hence, the number of edges produced at the bottom level is bounded by

$$\sum_j |V_j^i|^2 \le O(\lceil\log(R/w_{\min})\rceil\log^2 n)^{\lceil\log_{1-\epsilon_0}\mu\rceil}n^{1-\mu}n^{2\mu} \ .$$

We bound the number of edges produced at level i, where $0 \leq i < \lceil \log_{1-\epsilon_0} \mu \rceil$. For $1 \leq j \leq r(i)$, consider an application of the algorithm to a subgraph (V_j^i, E_j^i). There are $\lceil \log(R/w_{\min}) \rceil$ iterations at Step ii. In each iteration, Step ii4 produces $O(|V_j^i| \log^2 |V_j^i|)$ edges. Step ii5 produces

$$O((|V_j^i| \log^2 |V_j^i|)/|V_j^i|^{1-\epsilon_0})^2 = O(|V_j^i|^{2\epsilon_0} \log^4 |V_j^i|)$$

edges. It follows that if $\epsilon_0 \leq 1/2$, level i produces

$$O(\lceil \log(R/w_{\min}) \rceil (1-\epsilon_0)^{4i} \log^4 n \sum_j |V_j^i|)$$

$$= O(\lceil \log(R/w_{\min}) \rceil \log^2 n)^{i+1} n (1-\epsilon_0)^{4i} \log^2 n$$

edges. If $\epsilon_0 > 1/2$, level i produces

$$O(\lceil \log(R/w_{\min}) \rceil \log^2 n)^{i+1} n^{1-(1-\epsilon_0)^i} n^{2\epsilon_0 (1-\epsilon_0)^i} (1-\epsilon_0)^{4i} \log^2 n$$

$$= O(\lceil \log(R/w_{\min}) \rceil \log^2 n)^{i+1} n^{1-(1-\epsilon_0)^i (1-2\epsilon_0)} (1-\epsilon_0)^{4i} \log^2 n$$

edges. Therefore, the algorithm produces

$$O(\lceil \log(R/w_{\min}) \rceil \log^2 n)^{\lceil \log_{1-\epsilon_0} \mu \rceil} n^{1+\mu}$$

edges if $\epsilon_0 \leq 1/2$ and

$$O(\lceil \log(R/w_{\min}) \rceil \log^2 n)^{\lceil \log_{1-\epsilon_0} \mu \rceil} n^{1+\mu} + O(n^{2\epsilon_0} \lceil \log(R/w_{\min}) \rceil \log^4 n)$$

edges if $\epsilon_0 > 1/2$. ∎

7.2 Establishing the hop set properties

In this Subsection we prove that:

i. For all $\{u_1, u_2\} \subset V$, $\mathrm{dist}_{E^*}(u_1, u_2) \geq \mathrm{dist}_E(u_1, u_2)$.

ii. For every $\{u_1, u_2\} \subset V$ such that $\mathrm{dist}_E^d(u_1, u_2) \leq R$, we have

$$\mathrm{dist}_{E^* \cup E}^{d^*}(u_1, u_2) \leq (1+\hat{\epsilon})(1+4/\rho)^{\lceil \log_{(1-\epsilon_0)} \mu \rceil} \mathrm{dist}_E^d(u_1, u_2) \ ,$$

where $d^* = (4\beta\rho(1+\delta)+3)^{\lceil \log_{(1-\epsilon_0)} \mu \rceil}$.

Proposition 7.2 *Consider an edge $e = (u_1, u_2)$ that was created at the top level of Algorithm 5.1.*

i. If e was created at Step i:

- $w(e) \geq \mathrm{dist}_E(u_1, u_2)$
- $w(e) \leq (1+\hat{\epsilon}) \mathrm{dist}_E^d(u_1, u_2)$

ii. If e was created at Step ii4, then $w(e) = w_i/(2\rho) \geq \mathrm{dist}_{E \cap X^2}(u_1, u_2)$.

iii. Suppose that e was created in Step ii5. Let $\{X_1, X_2\} \subset \chi_b$ be such that $u_1 \in X_1$ and $u_2 \in X_2$ are the respective centers.

- $w(e) \geq \mathrm{dist}_E(u_1, u_2)$.

- *For all $v_1 \in X_1$ and $v_2 \in X_2$, $w(e) \leq (1+\hat{\epsilon})\mathrm{dist}^d_E(v_1,v_2) + w_i/\rho$.*

Proof: Part i is immediate. Part ii is immediate using part ii of Definition 2.5. To prove part iii, note that

$$w(e) = w_i/\rho + (1+\hat{\epsilon})\min\{\mathrm{dist}^d_E(v'_1,v'_2) | v'_1 \in X_1, v'_2 \in X_2\} .$$

Hence,

$$w(e) \leq (1+\hat{\epsilon})\mathrm{dist}^d_E(v_1,v_2) + w_i/\rho .$$

It remains to show that $w(e) \geq \mathrm{dist}_E(u_1,u_2)$. For every $v_1 \in X_1$ and $v_2 \in X_2$, we have $\mathrm{dist}_E(u_1,u_2) \leq \mathrm{dist}_E(v_1,v_2) + \mathrm{dist}_E(v_1,u_1) + \mathrm{dist}_E(v_2,u_2) \leq \mathrm{dist}_E(v_1,v_2) + w_i/\rho$. The last inequality follows from Definition 2.5 part ii. There exists $v_1 \in X_1$ and $v_2 \in X_2$, such that $w(e) \geq \mathrm{dist}_E(v_1,v_2) + w_i/\rho$. The last two inequalities imply that $w(e) \geq \mathrm{dist}_E(u_1,u_2)$. ∎

We prove that the edges E^* do not decrease distances:

Corollary 7.3 *For all $\{u_1,u_2\} \subset V$,*

$$\mathrm{dist}_{E^*}(u_1,u_2) \geq \mathrm{dist}_E(u_1,u_2)$$

Proof: It suffices to show that for every edge $(u_1,u_2) \in E^*$, $w(e) \geq \mathrm{dist}_E(u_1,u_2)$. We prove the claim by induction on the depth of the recursion. The base case follows from part i of Proposition 7.2. Suppose that the algorithm performs recursive calls. The claim for edges generated at the top level follows from parts ii and iii in Proposition 7.2. The claim for edges generated by the recursive calls follows using the induction hypothesis and the fact that the input to the recursive calls are subgraphs of the original graph, and distances on subgraphs are not smaller than distances on the graph itself. ∎

Let d' and $\epsilon' \geq \hat{\epsilon}$ be as follows. For all $1 \leq i \leq \lceil \log(R/w_{\min}) \rceil$, consider the ith iteration of Step ii. For each cluster $X \in \chi_s$, let $E_X \subset X^2$ be the edges produced by the recursive call applied in Step ii3 to X. For all $\{u_1,u_2\} \subset X$, if $\mathrm{dist}^d_{E\cap X^2}(u_1,u_2) \leq w_i/(2\beta\rho)$, then

$$\mathrm{dist}^{d'}_{E_X \cup E}(u_1,u_2) \leq (1+\epsilon')\mathrm{dist}^d_{E\cap X^2}(u_1,u_2) .$$

Proposition 7.4 *Let $\{u_1,u_2\} \subset V$ be such that $\mathrm{dist}^d_E(u_1,u_2) \leq R$. There exists a path $p^* \subset E^* \cup E$ between u_1 and u_2 such that*

$$|p^*| \leq (4\beta\rho(1+\delta)+1)d' + 2$$

and

$$w(p^*) \leq (1+\epsilon')(1+4/\rho)\mathrm{dist}^d_E(u_1,u_2) .$$

Proof: Consider a path p between u_1 and u_2 such that $|p| \leq d$ and $w(p) = \mathrm{dist}^d_E(u_1,u_2)$. We explicitly construct a path p^* in $E^* \cup E$ between u_1 and u_2 such that $|p^*| \leq (4\beta\rho(1+\delta)+2)d' + 3$ and $w(p^*) \leq (1+\epsilon')(1+4/\rho)w(p)$.

Let $1 \leq i \leq \lceil \log(R/w_{\min}) \rceil$ be such that $\mathrm{dist}^d_E(u_1,u_2) \in [w_i/2, w_i]$. Consider the ith iteration of Step ii and let χ be the cover computed at the ith iteration. To construct p^*, we partition p into at most

$$\lfloor 2w(p)/(w_i/(2\rho\beta(1+\delta))) \rfloor + 1 \leq 4\beta\rho(1+\delta) + 1$$

segments, where each segment is either a single edge or of weight at most $w_i/(2\rho(1+\delta)\beta)$, and any two consecutive segments have combined weight at least $w_i/(2\rho(1+\delta)\beta)$. Such a partition can be computed using a single traversal of p, where segments are cut when maximal. It follows from

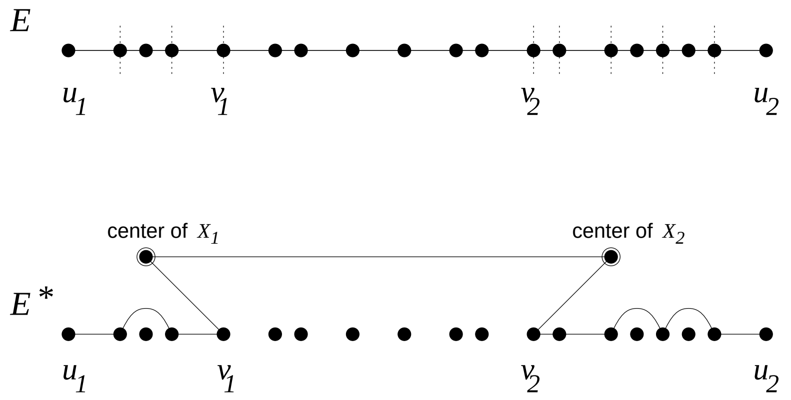

Figure 4: A path in E and a corresponding small size $E^* \cup E$ path

Remark 2.6 that every path in E of size d and weight at most $w_i/(2\rho\beta(1+\delta))$ is contained in at least one cluster of χ.

Let p' be the minimal subpath of p that contains all segments that are of weight at most $w_i/(2\rho\beta(1+\delta))$ and are not contained in any χ_s cluster. Note that each such segment is contained in some χ_b cluster. Let v_1 and v_2 be the endpoints of p'. It follows that $\{v_1, v_2\} \subset \bigcup_{X\in\chi_b} X$. Let $X_1 \in \chi_b$ and $X_2 \in \chi_b$ be such that $v_1 \in X_1$ and $v_2 \in X_2$. We specify at most 3 E^* edges that constitute a path p'^* between v_1 and v_2. We include p'^* in p^*. The path p'^* consists of an edge from v_1 to the center of X_1 (if other than v_1), an edge between the centers of X_1 and X_2 (if $X_1 \neq X_2$), and an edge between v_2 and the center of X_2 (if other than v_2). See Figure 4 for an illustration. The edges from v_1 and v_2 to the respective centers are included in E^* at Step ii4 and have weight $w_i/(2\rho)$. The edge between the centers is included in E^* at Step ii5 and it follows from Proposition 7.2 part iii that it has weight at most

$$(1+\hat{\epsilon})\mathrm{dist}^d_E(v_1, v_2) + w_i/\rho \ .$$

It follows that

$$w(p'^*) \leq (1+\hat{\epsilon})\mathrm{dist}^d_E(v_1, v_2) + 2w_i/\rho \ .$$

Note that $w(p') \geq \mathrm{dist}^d_E(v_1, v_2)$. Therefore,

$$w(p'^*) \leq (1+\hat{\epsilon})w(p') + 2w_i/\rho \ .$$

Consider a segment $\hat{p} = p_{u'_1 u'_2} \subset p \setminus p'$. We show that $\hat{p}$ has a corresponding path $\hat{p}^* \subset E^* \cup E$ between u'_1 and u'_2 of size $|\hat{p}^*| \leq d'$ and weight $w(\hat{p}^*) \leq (1+\epsilon')w(\hat{p})$. For every such segment $\hat{p}$, we place $\hat{p}^*$ in p^*. If $w(\hat{p}) > w_i/(2\rho(1+\delta)\beta)$ then $\hat{p}$ is a single edge and we choose $\hat{p}^* = \hat{p}$. Otherwise, $w(\hat{p}) \leq w_i/(2\rho(1+\delta)\beta)$. Every segment in $p \setminus p'$ of weight at most $w_i/(2\rho(1+\delta)\beta)$ must be contained in some cluster of χ_s. Let $X \in \chi_s$ be a cluster that contains $\hat{p}$. Note that $|\hat{p}| \leq d$. It follows from assumptions that

$$\mathrm{dist}^{d'}_{E_X \cup E}(u'_1, u'_2) \leq (1+\epsilon')\mathrm{dist}^d_{E\cap X^2}(u'_1 u'_2) \leq (1+\epsilon')w(\hat{p}) \ .$$

The path p contains at most $4\beta\rho(1+\delta)+1$ segments. If $p' \neq \emptyset$, then p' contains at least one segment. It follows that $p^* \subset E^* \cup E$ constructed above constitutes a path between the endpoints of p of size $|p^*| \leq (4\beta\rho(1+\delta)+1)d' + 2$ and weight

$$w(p^*) \leq w(p)(1+\max\{\epsilon', \hat{\epsilon}\}) + 2w_i/\rho \leq w(p)(1+\epsilon')(1+2w_i/(\rho w(p))) \leq w(p)(1+\epsilon')(1+4/\rho) \ .$$

▮

Consider a set E^* produced by an application of Algorithm 5.1 with parameter μ. Denote the recursion depth by $k(\mu) = \lceil \log_{1-\epsilon_0} \mu \rceil$ (see Proposition 6.1). We establish the correctness of the second claim:

Corollary 7.5 *For every pair of vertices $\{u_1, u_2\} \subset V$ such that $\mathrm{dist}^d_E(u_1, u_2) \le R$, we have*

$$\mathrm{dist}^{d^*}_{E^* \cup E}(u_1, u_2) \le (1 + \hat{\epsilon})(1 + 4/\rho)^{k(\mu)} \mathrm{dist}^d_E(u_1, u_2) \ ,$$

where $d^* = (4\beta\rho(1 + \delta) + 3)^{k(\mu)}$.

Proof: The proof is by induction on $k(\mu)$. If $k(\mu) = 0$, the proof follows from part i of Proposition 7.2. Consider an application of the algorithm when $k(\mu) \ge 1$. For each iteration $1 \le i \le \lceil \log(R/w_{\min}) \rceil$ consider the cover χ and a cluster $X \in \chi_s$. The induction hypothesis establishes that for all $\{u_1, u_2\} \subset X$, if $\mathrm{dist}^d_E(u_1, u_2) \le w_i/(2\beta\rho)$ then

$$\mathrm{dist}^{d'}_{E_X \cup E}(u_1, u_2) \le (1 + \hat{\epsilon})(1 + 4/\rho)^{k(\mu')} \mathrm{dist}^d_E(u_1, u_2) \ ,$$

where $d' = (4\beta\rho(1 + \delta) + 3)^{k(\mu')}$. It follows that $d' = (4\beta\rho(1 + \delta) + 3)^{k(\mu')}$ and ϵ' such that $(1 + \epsilon') = (1 + \hat{\epsilon})(1 + 4/\rho)^{k(\mu')}$ satisfy the assumptions of Proposition 7.4. Note that

$$(4\beta\rho(1 + \delta) + 1)d' + 2 \le (4\beta\rho(1 + \delta) + 3)^{k(\mu')+1} = (4\beta\rho(1 + \delta) + 3)^{k(\mu)} = d^*$$

and

$$(1 + \epsilon')(1 + 4/\rho) \le (1 + \hat{\epsilon})(1 + 4/\rho)^{k(\mu)} \ .$$

Therefore, Proposition 7.4 establishes that for every $\{u_1, u_2\} \subset V$ such that $\mathrm{dist}^d_E(u_1, u_2) \le R$, we have

$$\mathrm{dist}^{d^*}_{E^* \cup E}(u_1, u_2) \le (1 + \hat{\epsilon})(1 + 4/\rho)^{k(\mu)} \mathrm{dist}^d_E(u_1, u_2) \ .$$

▮

8 The Full-HopSet algorithm

We present algorithm Full-HopSet that inputs a weighted graph (V, E) and produces a hop set E^* of diameter d^*. Full-HopSet is based on iteratively performing calls to the Limited-HopSet Algorithm, that computes limited hop sets for d-edge distances.

Input Full-HopSet is provided with:

- A graph (V, E) with edge-weights $w : E \leftarrow \mathcal{R}_+$.
- A parameter $\epsilon_0 < 1$, $\epsilon_0 = \Omega(1/\log\log n)$.
- A parameter $\mu \le 1$, $\mu = \Omega(1/\log\log n)$.
- $\rho \gg 4$ determines tradeoffs between accuracy and time.
- $\hat{\epsilon} > 0$ determines the accuracy in the limited single-source shortest-paths computations performed by Limited-HopSet.
- An integer $\nu > 1$.

Output Full-HopSet constructs a collection E^* of weighted edges such that:

i. $|E^*| =$
$$O(\lceil \log_\nu (n/d^*) \rceil ((C \log^3 n)^{\lceil \log_{1-\epsilon_0} \mu \rceil} n^{1+\mu} + n^{2\epsilon_0} \log^5 n)) \ ,$$
for some fixed constant C.

ii. For all $\{u_1, u_2\} \subset V$, $\mathrm{dist}_{E^*}(u_1, u_2) \geq \mathrm{dist}_E(u_1, u_2)$.

iii. For all $\{u_1, u_2\} \subset V$,
$$\mathrm{dist}^{d^*}_{E^* \cup E}(u_1, u_2) \leq \left((1+\hat{\epsilon})(1+4/\rho)^{\lceil \log_{(1-\epsilon_0)} \mu \rceil} \right)^{\lceil \log_\nu (n/d^*) \rceil} \mathrm{dist}_E(u_1, u_2) \ ,$$
where $d^* = (4.4\rho \log n + 3)^{\lceil \log_{(1-\epsilon_0)} \mu \rceil} = O(\rho \log n)^{\lceil \log_{(1-\epsilon_0)} \mu \rceil}$

Algorithm 8.1 [Full-HopSet$((V, E), \epsilon_0, \mu, \rho, \nu)$]

i. Set $\beta \leftarrow \log n$ and $\delta \leftarrow 0.1$.

ii. $d^* \leftarrow (4\beta\rho(1+\delta) + 3)^{\lceil \log_{(1-\epsilon_0)} \mu \rceil}$

iii. $E_0 \leftarrow E$

iv. For $i = 1, \ldots, \lceil \log_\nu (n/d^*) \rceil$:
$E^*_i \leftarrow$ Limited-HopSet$(\mu, (V, E_{i-1}), nw_{\max}, \nu d^*)$
$E_i \leftarrow E_{i-1} \cup E^*_i$

v. Return $E^* \leftarrow \bigcup_{1 \leq i \leq \lceil \log_\nu (n/d^*) \rceil} E^*_i$

8.1 Correctness

We establish that the set E^* constructed by the algorithm possesses the specified properties. Properties i and ii follow immediately from properties i and ii, respectively, of the output of Algorithm 5.1. For property i note that $\lceil \log_2 nw_{\max}/w_{\min} \rceil = O(\log n)$.

To prove that property iii holds, we use the following:

Proposition 8.2 *Let E' and $\hat{E}$ be sets of weighted edges. Let d', $\hat{d}$, and k be integers. Let $A \in \mathcal{R}_+$. If for all $\{u_1, u_2\} \subset V$, $\mathrm{dist}^{d'}_{E'}(u_1, u_2) \leq A\mathrm{dist}^{\hat{d}}_{\hat{E}}(u_1, u_2)$ then for all $\{u_1, u_2\} \subset V$, $\mathrm{dist}^{kd'}_{E'}(u_1, u_2) \leq A\mathrm{dist}^{k\hat{d}}_{\hat{E}}(u_1, u_2)$.*

Proof: Consider a path $\hat{p} \subset \hat{E}$ between u_1 and u_2 such that $|\hat{p}| \leq k\hat{d}$. We construct a path $p' \subset E'$ such that $|p'| \leq kd'$ and $w(p') \leq Aw(\hat{p})$. Partition $\hat{p}$ to segments $\hat{h}_i$ $(1 \leq i \leq r)$, where $r \leq k$ and for $1 \leq i \leq r$, $|\hat{h}_i| \leq \hat{d}$. It follows from the assumptions that there are paths $h'_i \subset E'$ $(1 \leq i \leq r)$ such that for $1 \leq i \leq r$, h'_i has the same endpoints as $\hat{h}_i$, $|h'_i| \leq d'$, and $w(h'_i) \leq Aw(\hat{h}_i)$. It follows that $p' = \bigcup_{1 \leq i \leq r} h'_i$ is as claimed. ∎

We establish that property iii holds:

Proposition 8.3 *For $0 \leq i \leq \lceil \log_\nu (n/d^*) \rceil$ and all $\{u_1, u_2\} \subset V$,*
$$\mathrm{dist}^{d^*}_{E_i}(u_1, u_2) \leq (1+\hat{\epsilon})^i (1+4/\rho)^{i\lceil \log_{(1-\epsilon_0)} \mu \rceil} \mathrm{dist}^{\nu^i d^*}_{E_0}(u_1, u_2) \ .$$

Proof: The proof is by induction on i. The case where $i = 0$ is immediate. Consider $i \geq 1$. The induction hypothesis asserts that for all $\{u_1, u_2\} \subset V$,

$$\mathrm{dist}^{d^*}_{E_{i-1}}(u_1,u_2) \leq (1+\hat{\epsilon})^{i-1}(1+4/\rho)^{(i-1)\lceil \log_{(1-\epsilon_0)}\mu\rceil}\mathrm{dist}^{\nu^{i-1}d^*}_{E_0}(u_1,u_2) .$$

It follows from Proposition 8.2 that

$$\mathrm{dist}^{\nu d^*}_{E_{i-1}}(u_1,u_2) \leq (1+\hat{\epsilon})^{i-1}(1+4/\rho)^{(i-1)\lceil \log_{(1-\epsilon_0)}\mu\rceil}\mathrm{dist}^{\nu^{i}d^*}_{E_0}(u_1,u_2) .$$

Property iii of the output of Algorithm 5.1 asserts that for all $0 \leq i \leq \lceil \log_\nu(n/d^*)\rceil$ and for all $\{u_1, u_2\} \subset V$,

$$\mathrm{dist}^{d^*}_{E_i}(u_1,u_2) \leq (1+\hat{\epsilon})(1+4/\rho)^{\lceil \log_{(1-\epsilon_0)}\mu\rceil}\mathrm{dist}^{\nu d^*}_{E_{i-1}}(u_1,u_2) .$$

The proof follows by combining the last two inequalities. ∎

8.2 Complexity

We bound the resources (time and work) utilized by Full-HopSet when the subroutine Limited-HopSet employs parallel weighted BFS for d-edge shortest paths computations.

Proposition 8.4 Full-HopSet *runs in*

$$O(\lceil \log_\nu(n/d^*)\rceil(\nu d^*\epsilon_0^{-1}\hat{\epsilon}^{-1}\log^3 n))$$

time and performs

$$O(\lceil \log_\nu(n/d^*)\rceil m'(n^{\epsilon_0}\log^4 n + (D\log^3 n)^{\lceil \log_{1-\epsilon_0}\mu\rceil}n^\mu \log n))$$

work, where

$$m' \leq m + O(\lceil \log_\nu(n/d^*)\rceil((C\log^3 n)^{\lceil \log_{1-\epsilon_0}\mu\rceil}n^{1+\mu} + n^{2\epsilon_0}\log^5 n)) .$$

Proof: The work is dominated by the calls to Limited-HopSet. The algorithm performs $\lceil \log_\nu(n/d^*)\rceil$ calls to Limited-HopSet, sequentially. Each call to Limited-HopSet takes

$$O(\epsilon_0^{-1}\hat{\epsilon}^{-1}\nu d^*\log^3 n)$$

time. Hence, the algorithm runs in

$$O(\lceil \log_\nu(n/d^*)\rceil\epsilon_0^{-1}\hat{\epsilon}^{-1}\nu d^*\log^3 n)$$

time. The ith call to Limited-HopSet performs

$$O(|E_{i-1}|n^{\epsilon_0}\log^4 n + (D\log^3 n)^{\lceil \log_{1-\epsilon_0}\mu\rceil}|E_{i-1}|n^\mu\log n)$$

work. Note that $|E_{i-1}| \leq m + |E^*|$. Therefore, the work is bounded by

$$O(\lceil \log_\nu(n/d^*)\rceil m'(|n^{\epsilon_0}\log^4 n + (D\log^3 n)^{\lceil \log_{1-\epsilon_0}\mu\rceil}n^\mu\log n)) ,$$

where $m' = m + |E^*|$. ∎

When the subroutine Limited-HopSet employs parallel Bellman-Ford for d-edge shortest paths computations, $\hat{\epsilon} = 0$. We obtain the following bounds:

Proposition 8.5 Full-HopSet *runs in*

$$O(\lceil \log_\nu(n/d^*)\rceil(\nu d^*\epsilon_0^{-1}\log^3 n))$$

time and performs

$$O(\lceil \log_\nu(n/d^*)\rceil m'\nu d^*(n^{\epsilon_0}\log^3 n + (D\log^3 n)^{\lceil \log_{1-\epsilon_0}\mu\rceil}n^\mu))$$

work, where

$$m' \leq m + O(\lceil \log_\nu(n/d^*)\rceil((C\log^3 n)^{\lceil \log_{1-\epsilon_0}\mu\rceil}n^{1+\mu} + n^{2\epsilon_0}\log^5 n)) .$$

8.3 Bounds for hop sets constructions in parallel

For the choice $\hat{\epsilon} = O(1/\rho)$ and $\nu = 2$. The hop set attributes are:

d^*	approximation	$\lvert E^*\rvert$
$O(\rho \log n)^{\lceil \log_{(1-\epsilon_0)} \mu \rceil}$	$(1 + O(1/\rho))^{\lceil \log n \rceil \lceil \log_{(1-\epsilon_0)} \mu \rceil}$	$O(n^{1+\mu}(C \log n)^{3\lceil \log_{(1-\epsilon_0)} \mu \rceil + 1} + n^{2\epsilon_0} \log^6 n)$

Polylogarithmic time In addition, assume that $\rho = O(\log^r n)$, $\mu = \epsilon_0$, and that ϵ_0 is fixed. The hop set attributes are:

d^*	approximation	$\lvert E^*\rvert$
$O(\log n)^{(r+1)k(\epsilon_0)}$	$(1 + O(1/\log^{r-1} n))$	$O(n^{1+\epsilon_0} \log^{3k(\epsilon_0)+1} n)$

The algorithm runs in

$$O(\log^{(r+1)(k(\epsilon_0)+1)+2} n)$$

time and performs

$$O((m + |E^*|)n^{\epsilon_0} \log^{3k(\epsilon_0)+2} n)$$

work.

Improved work but worse time If we choose $\epsilon_0 = \mu_0 = \theta(1/\log\log n)$, $\nu = 2$, and $\rho = O(\log^r n)$ for some fixed r, the hop set attributes are:

d^*	approximation	$\lvert E^*\rvert$
$(\log n)^{O((\log\log n)^2)}$	$(1 + O((\log\log n)^2/\log^{r-1} n))$	$O(n^{1+O(1/\log\log n)})$

The algorithm performs

$$\tilde{O}(mn^{O(1/\log\log n)})$$

work and runs in time

$$(\log n)^{O((\log\log n)^2)} .$$

Improved approximation quality A choice of $\rho = n^{-\delta}$ for some fixed δ, would lead to a hop set with better approximation quality (but larger diameter and hence, worse time bounds). To maintain the approximation quality we choose $\hat{\epsilon} = O(\lceil \log_{(1-\epsilon_0)} \mu \rceil / \rho)$, if the parallel weighted BFS algorithm is employed, or otherwise use the parallel Bellman-Ford algorithm.

9 Concluding remarks

We give some remarks and suggest future research directions.

We used two different schemes, Algorithm 4.2 and Algorithm 4.8, for the sequential hop sets constructions. The parallel version, Algorithm 5.1 follows the scheme of Algorithm 4.8. We remark that similarly, we can design a parallel algorithm that follows the scheme of Algorithm 4.2 and obtain different tradeoffs. By using the second scheme we obtain more efficient parallel algorithms for shortest paths between distant pairs of vertices (as done in Subsection 4.3 for the sequential version).

The randomization in our parallel shortest paths stems from the pairwise cover constructions in [5]. Deterministic parallel cover algorithm would yield a deterministic parallel approximation scheme for shortest paths.

Our algorithms do not extend to general directed graphs because pairwise covers can not be constructed for directed graphs. Our algorithms, however, do apply to "almost symmetric" directed graphs, where for every pair of vertices $(u_1, u_2) \in V \times V$, the distance from u_1 to u_2 is within a constant or a polylogarithmic factor of the distance from u_2 to u_1. We can obtain pairwise covers for almost symmetric digraphs by applying the cover algorithm to the graph where edge-directions are ignored. It is easy to see that the resulting collection of clusters, with restored edge directions, constitutes a pairwise cover.

A way to save some logarithmic factors is by searching for parallel constructions that produce sparser covers than the constructions in [5]. The hop set algorithm uses covers to partition the graph recursively in a divide-and-conquer style approach. The ratio of the size of the cover to the size of the original graph is crucial for the complexity: Each level of the recursion increases the total size of the subgraphs considered by a factor that equals this ratio. This ratio is about $O(\log^2 n)$ in the parallel cover construction, but is much tighter in the sequential constructions. We remark that the size of the covers was not as critical for the applications in [5]

One of the contributions of this paper is explicitly employing hop sets, and efficient hop set constructions, to produce efficient parallel approximate shortest paths algorithms for undirected graphs. We believe that searching for good hop sets is a promising general approach for work-efficient parallel exact shortest paths, directed shortest-paths, and reachability algorithms. In addition, we find the existence of good hop sets to be an intriguing research problem on its own right.

We established the existence and presented efficient algorithms for constructing sparse hop sets of polylogarithmic diameter and approximation quality $(1 + 1/\operatorname{polylog} n)$ for undirected graphs. We ask whether comparable tradeoffs (and possibly efficient construction algorithms) exist for the following problems: (in conjectured order of difficulty)

i. Exact hop sets ($\epsilon = 0$) for undirected graphs.

ii. Directed reachability.

iii. Directed graphs with nonnegative weights (approximate or exact),

iv. Directed graphs with real weights.

Acknowledgements The author would like to thank David Applegate and Noga Alon for discussions regarding existence of good hop sets, and Phil Klein, John Reif, and Mihalis Yannakakis for discussions on parallel shortest paths algorithms.

References

[1] B. Awerbuch, B. Berger, L. Cowen, and D. Peleg. Near-linear cost sequential and distributed constructions of sparse neighborhood covers. In *Proc. 34th IEEE Annual Symposium on Foundations of Computer Science*, pages 638–647. IEEE, 1993.

[2] B. Awerbuch and D. Peleg. Sparse partitions. In *Proc. 31st IEEE Annual Symposium on Foundations of Computer Science*, pages 503–513. IEEE, 1990.

[3] E. Cohen. Efficient parallel shortest-paths in digraphs with a separator decomposition. *J. Alg.*, 21:331–357, 1996.

[4] E. Cohen. Using selective path-doubling for parallel shortest-path computations. *J. Alg.*, 22:30–56, 1997.

[5] E. Cohen. Fast algorithms for t-spanners and stretch-t paths. *SIAM J. Comput.*, 28:210–236, 1999.

[6] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to algorithms.* McGraw-Hill Book Co., New York, 1990.

[7] P. N. Klein. A parallel randomized approximation scheme for shortest paths. Draft of journal version, 1992.

[8] P. N. Klein and S. Sairam. A parallel randomized approximation scheme for shortest paths. In *Proc. 24th Annual ACM Symposium on Theory of Computing*, pages 750–758. ACM, 1992.

[9] P. N. Klein and S. Sairam. A linear-processor polylog-time algorithm for shortest paths in planar graphs. In *Proc. 34th IEEE Annual Symposium on Foundations of Computer Science*, pages 259–270. IEEE, 1993.

[10] T. H. Spencer. Time-work tradeoffs for parallel algorithms. *J. Assoc. Comput. Mach.*, 44:742–778, 1997.

[11] J. D. Ullman and M. Yannakakis. High-probability parallel transitive closure algorithms. *SIAM J. Comput.*, 20:100–125, 1991.

Contents