Fast C-K-R Partitions of Sparse Graphs[*]

Manor Mendel
Computer Science Division
The Open University of Israel
mendelma@gmail.com

Chaya Schwob
Computer Science Division
The Open University of Israel cschwob@nds.com

June 22, 2009

Abstract

We present fast algorithms for constructing probabilistic embeddings and approximate distance oracles in sparse graphs. The main ingredient is a fast algorithm for sampling the probabilistic partitions of Calinescu, Karloff, and Rabani in sparse graphs.

1 Introduction

Metric decompositions aim to partition the points of a metric space into blocks such that close-by points tend to be placed in the same block while distant pairs of points in different blocks. For most metric spaces, no straightforward interpretation of these goals exists.

One successful compromise is the notion of *probabilistic partition.* A Δ-bounded probabilistic partition is a probability distribution over partitions of the metric space, such that in every partition in the distribution, the diameters of the blocks are at most Δ, while "close-by" pairs of points are in the same block with "high" (or at least "non-negligible") probability.

Probabilistic partitions first appeared, to the best of our knowledge,[1] in a paper of Linial and Saks [19], and publicized in the work of Bartal [3] on probabilistic embeddings. Calinescu, Karloff and Rabani [10] introduced the following probabilistic partition of metric spaces which we describe as an algorithm that samples a partition from the probability distribution.

[*]M. Mendel was partially supported by an ISF grant no. 221/07, a BSF grant no. 2006009, and a gift from Cisco research center. This work is part of the M.Sc. thesis of C. Schwob prepared in the Computer Science Division of the Open University of Israel.

[1]Closely related notions of partitions appeared before, e.g. in [17].

Algorithm 1 CKR-Partition

```
Input: A finite metric space $(X, \rho)$, scale $\Delta > 0$
Output: Partition $P$ of $X$
  $\pi \coloneqq$ random permutation of $X$
  $R \coloneqq$ random number in $\left[\frac{\Delta}{4}, \frac{\Delta}{2}\right]$
  for $i = 1$ to $|X|$ do
      $C_i \coloneqq \{y \in X : \rho(y, x_{\pi(i)}) \le R\} \setminus \bigcup_{j=1}^{i-1} C_j$
  return $P \coloneqq \left\{C_1, \dots, C_{|X|}\right\} \setminus \{\emptyset\}$
```

We call the probabilistic partition P sampled by Algorithm 1, Δ-*bounded CKR partition.* Naïve implementations of Algorithm 1 take $\Omega(n^2)$ time for n-point metric spaces. It seems hard to break the $\Omega(n^2)$ barrier on the running time in general finite metric spaces. However, in many situations, the metric spaces we deal with come from the shortest-path metric on relatively sparse graphs. In those cases we can do better, as the following theorem shows.

Theorem 1. *Suppose we are given a positive number $\Delta > 0$ and an undirected graph with positive edge weights $G = (X, E, \omega)$. Suppose G has n vertices and m edges, and let ρ denote the shortest-path metric in G. One can sample a Δ-bounded CKR partition of (X, ρ) in expected $O(m \log n + n \log^2 n)$ time.*

The sampling will be accomplished by Algorithm 2 (Section 3).

CKR partitions have found many algorithmic (as well as mathematical) applications, and we mention only few of them here. They were introduced as part of an approximation algorithm to the 0-extension problem [10, 12]. Fakcharoenphol, Rao and Talwar [13] used them to obtain an asymptotically tight probabilistic embedding into trees, which we call FRT-embedding. Probabilistic embeddings are used in many of the best known approximation and online algorithms as a reduction step from general metrics into tree metrics. Mendel and Naor [21] showed that FRT-embedding possesses a stronger embedding property, which they called “maximum gradient embedding”. Recently, Räcke [23] used FRT-embeddings to obtain hierarchical decompositions for congestion minimization in networks, and used them to give an $O(\log n)$ approximation algorithm for the minimum bisection problem and an $O(\log n)$ competitive online algorithm for the oblivious routing problem. Krauthgamer et. al [16] used CKR-partitions to give a new proof of Bourgain’s embedding theorem. Mendel and Naor [22] used them to obtain an asymptotically tight metric Ramsey theorem and approximate distance oracles.

The improved running time of the sampling of CKR partitions may improve the running time of many of their applications. In order to keep the paper short we work out the details of only two (related) applications of CKR partitions: FRT-embeddings, and approximate distance oracles based on CKR-partitions.

Probabilistic embedding into ultrametrics [2, 3].

An ultrametric ν on X is a metric which satisfies $\nu(x,z) \le \max\{\nu(x,y), \nu(y,z)\}$, for every $x, y, z \in X$. A probabilistic embedding of a metric space (X, ρ) into ultrametrics with distortion D is a probability distribution Π over ultramerics ν on X such that

1. For every $x, y \in X$, $\Pr_\Pi[\nu(x,y) \ge \rho(x,y)] = 1$.
2. For every $x, y \in X$, $\mathbb{E}_\Pi[\nu(x,y)] \le D \cdot \rho(x,y)$.

FRT-embedding is a probabilistic embedding into ultrametrics with distortion $O(\log n)$ for every n-point metric space [13]. This bound is asymptotically tight for certain classes of finite metric spaces, such as graphs of high girth[3], grids [2], and expanders [18].

Approximate distance oracles.

An approximate distance oracle is a data structure with "compact" ($o(n^2)$) storage that answers (approximate) distance queries in a given n-point metric space in constant time. A simple counting argument over all bi-partite graphs shows that exact, and even 2.99 approximation is impossible when the storage is $o(n^2)$. The history of this problem is nicely summarized in [25]. In particular, Thorup and Zwick [25] gave an asymptotically tight trade-off between the approximation and the storage[2]: For every $k \in \mathbb{N}$, they constructed $(2k-1)$-approximate distance oracle requiring $O(kn^{1+1/k})$ storage, and answering queries in $O(k)$ time. Recently Mendel and Naor [22] presented different approximate distance oracles, based on CKR partitions. While those oracles do not give optimal approximation/storage trade-off,[3] they answer distance queries in an absolute constant time, regardless of the approximation parameter.

Theorem 2. *Let $G = (X, E, \omega)$ be an n-vertex weighted graph with m edges, and let ρ be the shortest-path metric on X. Then*

1. *It is possible to sample from FRT-embedding of (X, ρ) in $O(m \log^3 n)$ expected time.*
2. *It is possible to construct in $O(mn^{1/k} \log^3 n)$ expected time an $O(k)$-approximate distance oracle for (X, ρ) based on CKR partitions whose storage is $O(n^{1+1/k})$.*

For approximate distance oracles, it is also possible to improve the naïve construction time even when the metric is given as distance matrix, by first constructing a spanner of the metric with $o(n^2)$ edges, and then use the fast CKR partitions for sparse graphs on that spanner.

[2]The lower bound on the approximation assumes a conjecture of Erdös about the number of edges possible in graph with a given number of vertices and a given girth, see [25].

[3]As reported in [22], oracles of size $O(n^{1+1/k})$ support approximation factor of $128k$ in the queries. While the constant 128 can be reduced by optimizing the parameters in the construction, it is unlikely to get below 8.

Theorem 3. *For n-point metric spaces given as distance matrix, it is possible to construct $O(k)$-approximate distance oracle based on CKR partitions whose storage is $O(n^{1+1/k})$, in $O(n^2)$ expected time.*

We remark that a different probabilistic partition, developed by Bartal [4, 5] and Abraham et. al. [1], have properties similar to (and even stronger than) CKR partitions. However, we do not see an easy way to quickly obtain a sample from this distribution when the graph is sparse.

Further Results

The second named author presents in [24] an efficient PRAM algorithm for sampling CKR partitions and constructing approximate distance oracles in weighted graphs. The running time of the algorithm is $\text{polylog}(n)$ and the total work is $O(m \,\text{polylog}(n))$.

Outline of the paper.

After setting up in Section 2 the notation and reviewing the properties of CKR partitions, we prove Theorem 1 in Section 3.

In many applications (and in particular, probabilistic embeddings and approximate distance oracles), probabilistic partitions are applied hierarchically, using an exponentially decreasing series of scales. This naïvly implies an added $O(\log \Phi)$ factor in the running time, where Φ is the spread[4] of the metric. There is a standard technique that converts this $\log \Phi$ factor into a $\log n$ factor. However, we are not aware of a concrete implementation that satisfies the efficiency requirements needed in this paper. We therefore sketch the details of this technical step in Section 4. The specific applications examined in this paper, Theorem 2 and Theorem 3, are discussed in Section 5.

2 Preliminaries

For simplicity of the presentation, the model of computation we assume is a unit-cost, real-word RAM machine. In this model words can hold real numbers and arithmetic, comparison, and truncation operations take unit time. Our algorithms, however, do not take advantage of the unrealistic power of this model, and can also be presented in a more realistic computational models such as the unit cost floating-point word RAM model (cf. [15, Sec. 2.2]).

The diameter of a finite subset $Y \subseteq (X, \rho)$ is defined as $\text{diam}(Y) = \max\{\rho(x,y) : x, y \in Y\}$. For simplicity of the presentation, we assume that the given finite metric (X, ρ) has minimum non-zero distance 1, and diameter $\text{diam}(X) = \Phi$.

The (closed) ball around x at radius r is defined as $B_\rho(x,r) = \{y \in X : \rho(x,y) \le r\}$. When ρ is clear from the context we may omit it from the notation.

[4]The spread is the ratio between the diameter and the smallest non-zero distance in the metric

Given a partition P of X, and $x \in X$, we denote by $P(x)$ the block of P which contains x.

Δ-bounded CKR partitions have an obvious upper bound of Δ on the diameter of the blocks in the partition. The following is the padding property they enjoy.

Lemma 2.1 ([13, 22])**.** *Let P be a Δ-bounded CKR partition of the metric (X, ρ). Then, for every $x \in X$, and $t \le \Delta/8$,*

$$\Pr_{P \sim \mathcal{P}} \left[B\left(x, t\right) \subseteq P(x) \right] \ge \left(\frac{|B_X(x, \Delta/8)|}{|B_X(x, \Delta)|} \right)^{\frac{16t}{\Delta}} . \tag{1}$$

In this paper we define hierarchical partition of a metric space (X, ρ) as a sequence of $\lceil \log_8 \Phi \rceil + 2$ partitions $P_{-1}, P_0, \ldots, P_{\lceil \log_8 \Phi \rceil}$ such that P_i is a partition of X at scale 8^i, and P_i is a refinement of P_j when $i \le j$, i.e., for every $x \in X$, $P_i(x) \subseteq P_j(x)$. Given a sequence $(Q_j)_{j \ge -1}$ of partitions, where Q_j is 8^j-bounded partition of X, the common refinement of $(Q_j)_j$ is a hierarchical partition $(P_j)_{j \ge -1}$ in which $P_j = \{\bigcap_{\ell \ge j} C_\ell : \ C_\ell \in Q_\ell\}$

By sampling stochastically independent CKR partitions at the different scales and then taking their common refinement, we obtain the following result.

Lemma 2.2 ([22])**.** *Fix a finite metric space (X, ρ). Then there exists (efficiently sampleable) probability distribution $\mathcal{H}$ over hierarchical partitions such that for every $x \in X$, and every $0 < \beta < 1/8$,*

$$\Pr_{(P_{-1}, \ldots, P_{\lceil \log_8 \Phi \rceil}) \sim \mathcal{H}} \left[\forall\ k \ge -1,\ B(x, \beta 8^k) \subseteq P_k(x) \right] \ge |X|^{-16\beta} .$$

A finite ultrametric (X, ν) can be represented by a tree as follows.

Definition 4. An ultrametric tree (T, Γ) is a metric space whose elements are the leaves of a rooted finite tree T. Associated with every vertex $u \in T$ is a label $\Gamma\left(u\right) \ge 0$ such that $\Gamma\left(u\right) = 0$ iff u is a leaf of T. If a vertex u is a child of a vertex v then $\Gamma\left(u\right) \le \Gamma\left(v\right)$. The distance between two leaves $x, y \in T$ is defined as $\Gamma\left(\mathbf{lca}\left(x, y\right)\right)$, where $\mathbf{lca}\left(x, y\right)$ is the least common ancestor of x and y in T.

Every finite ultrametric can be represented by an ultrametric tree, and vice versa: the metric on ultrametric tree is a finite ultrametric. Hierarchical partition $\{P_k\}_{k=-1}^{\lceil \lg \phi \rceil}$ of (X, ρ) naturally corresponds to an ultrametric ν on X where $\nu(x, y) = 8^{\min\{j:\ P_j(x) = P_j(y)\}}$.

Let $G = (X, E, \omega)$ be an undirected positively weighted graph. Let $\rho : X \times X \to [0, \infty)$ be the shortest-path metric on G. We denote by $n = |X|$ the number of vertices, and by $m = |E|$ the number of edges. We assume an adjacency list representation of graphs.

The single source shortest paths in weighted undirected graphs problem [USSSP] is used as a subroutine in our algorithm. Given a weighted graph with n vertices and m edges, Dijkstra's classical USSSP algorithm [11] with source w maintains for each vertex v an upper bound on the distance between w and v, $\delta(v)$. If $\delta(v)$ has not been assigned yet, it is interpreted as infinite. Initially, we just set $\delta(w) = 0$, and we have no visited vertices. At each iteration, we select an unvisited vertex u with the smallest finite $\delta(u)$, visit it, and relax all its edges. That is, for each incident edge $(u, v) \in E$, we set $\delta(v) \leftarrow \min\{\delta(v), \delta(u) + \omega(u, v)\}$. We continue until no vertex is left unvisited. Using Fibonacci heaps [14] or Bordal's priority queues [9], Dijkstra's algorithm is implemented in $O(m + n \lg n)$ time.

3 Fast CKR partitions

Given an undirected positively weighted graph $G = (X, E, \omega)$ with n vertices and m edges whose shortest path metric is denoted by ρ, and $\Delta > 0$, we show how to implement Algorithm 1 in $O\left(m \lg n + n \log^2 n\right)$ expected time.

First, we sample a random permutation π, which can be generated in linear time using several methods, *e.g.,* Knuth Shuffle (see [8]). Next, we sample R uniformly[5] in the range $\left[\frac{\Delta}{4}, \frac{\Delta}{2}\right]$.

We then use a variant of Dijkstra's algorithm for computing the blocks. The algorithm performs $|X|$ iterations. In the i-th iteration, all vertices in $B_\rho\left(x_{\pi(i)}, R\right)$ not yet assigned to some block are put in C_i. In order to gain the improved running time of Theorem 1, we change Dijkstra's algorithm to return the distance of a point v from $\pi(i)$ *only if* this distance is smaller then the distance of v from $\pi(j)$ for all $j < i$.

Technically, this is done as follows. Consider the i-th iteration and let $\delta(v)$ be the variable that holds the Dijkstra's algorithm's current estimate on the distance between $\pi(i)$ and v. In Dijkstra's algorithm $\delta(v)$ is usually initialized to ∞ and then gradually decreases until u is extracted from the priority queue, at which point $\delta(v) = \rho(\pi(i), v)$. In the variant of Dijkstra's algorithm used in Algorithm 2, $\delta(\cdot)$ are not reinitialized when the value of i is changed. This means that now at the end of the $(i-1)$-th iteration, $\delta(v) = \min_{j<i} \rho(\pi(j), v)$, which in turn means that an edge (u, w) is relaxed in the i-th iteration only when $\pi(i)$ is the closest center to both u, and w among $\pi(j)$, $j \leq i$. This dramatically reduces the number of relaxations being done, and does not hurt the correctness of the algorithm. The full details are given in Algorithm 2.

Lemma 3.1. *After the i-th iteration of the loop on lines 6–18 of Algorithm 2,*

$$\delta(v) = \begin{cases} \min_{j \leq i} \rho(\pi(j), v) & \textit{if } \min_{j \leq i} \rho(\pi(j), v) \leq R, \\ \infty & \textit{otherwise.} \end{cases} \tag{2}$$

[5] A closer look on the analysis of the CKR partitions (see [22]) reveals that it is sufficient to sample R from discrete distribution having resolution of $\Delta/c \log n$, and therefore this step can be carried out in a "realistic" computational model such as the unit cost floating-point word RAM model.

Algorithm 2 Graph-CKR-Partition

```
Input: Graph $G = (X, E, \omega)$, scale $\Delta > 0$
Output: Partition $P$ of $X$
 1: Generate random permutation $\pi$ of $X$
 2: Sample a random $R \in \left[\frac{\Delta}{4}, \frac{\Delta}{2}\right]$
 3: for all $v \in X$ do
 4:     $\delta(v) \coloneqq \infty$
 5:     $P(v) \coloneqq 0$
 6: for $i \coloneqq 1$ to $|X|$ do  // Perform modified Dijkstra's alg starting from $\pi(i)$
 7:     $\delta(\pi(i)) \coloneqq 0$
 8:     $Q \coloneqq \emptyset$  // $Q$ is a priority queue with $\delta$ being the key
 9:     $w \coloneqq \pi(i)$
10:     while $\delta(w) \le R$ do  // $w$ is visited now
11:         if $P(w) = 0$ then
12:             $P(w) \coloneqq i$
13:         for all $u :\ (u, w) \in E$ do
14:             if $\delta(u) > \delta(w) + \omega(u, w)$ then  // Relax edges adjacent to $w$
15:                 $\delta(u) \coloneqq \delta(w) + \omega(u, w)$
16:                 if $u \notin Q$ then
17:                     Insert $u$ into $Q$
18:         Extract $w \in Q$ with minimal $\delta(w)$
19: return $P$
```

Sketch of a proof. Proof by induction on i. When $i = 1$, and as long $\delta(w) \le R$ in the *While* loop of line 10, the algorithm behaves *exactly* as Dijkstra's algorithm and hence (2) is true for $i = 1$.

Assume inductively that (2) is correct for $i - 1$. If $\min_{j \le i-1} \rho(\pi(j), v) \le \rho(\pi(i), v)$, then clearly the i-th iteration will not change $\delta(v)$, and by the inductiion hypothesis we are done.

Assume now that $\min_{j \le i-1} \rho(\pi(j), v) > \rho(\pi(i), v)$, and $\rho(\pi(i), v) \le R$. Let $\pi(i) = v_0, v_1, \ldots, v_\ell = v$ be a shortest-path between $\pi(i)$ and v. We claim that for every $t \in \{1, \ldots, \ell\}$, $\min_{j \le i-1} \rho(\pi(j), v_t) > \rho(\pi(i), v_t)$, since otherwise we had

$$\min_{j \le i-1} \rho(\pi(j), v) \le \min_{j \le i-1} \rho(\pi(j), v_t) + \rho(v_t, v) \le \rho(\pi(i), v_t) + \rho(v_t, v) = \rho(\pi(i), v).$$

Hence all the edges along the path $\pi(i) = v_0, \ldots, v_\ell = v$ will be relaxed in the i-th iteration, and so in the end of the i-th iteration, $\delta(v) = \rho(\pi(i), v)$. □

Proof of Theorem 1. We first prove the correctness of Algorithm 2, i.e., that $P(v) = \min\{i :\ \rho(\pi(i), v) \le R\}$ for every $v \in V$. Let $i_0 = \min\{i :\ \rho(\pi(i), v) \le R\}$. This means that $\min_{j < i_0} \rho(\pi(j), v) > R \ge \rho(\pi(i_0), v)$. By Lemma 3.1 at the beginning of the i_0-th iteration, $\delta(v) = \infty$, and hence $P(v) = 0$ and by the end of the (i_0)-th iteration, $\delta(v) = \rho(\pi(i_0), v)$, and necessarily $P(v) = i_0$. Note that once $P(v)$ is set to a non-zero value, its value will not change.

We next bound the running time. we will show that every vertex is inserted into the queue $O(\log n)$ times in expectation, and every edge (u,v) of G undergoes $O(\log n)$ relaxations in expectation. Consider the non-increasing sequence $a_i = \min_{j\le i} \rho(\pi(j), v)$. In the i-th iteration, $\delta(v)$ decreases if and only if $a_{i-1} > a_i$. Note that $a_{i-1} > a_i$ means that $\rho(\pi(i), v)$ is the minimum among $\{\rho(\pi(j), v) \mid j \le i\}$, and the probability (over π) for this to happen is at most $1/i$. By linearity of the expectation, the expected number of rounds of the i-loop where $\delta(v)$ decreases (and hence v is inserted into the queue) is at most

$$\sum_{i=1}^{n} \frac{1}{i} \le 1 + \ln n.$$

Furthermore, by another application of the linearity of expectation, the expected number of edge relaxations is at most

$$O\Big(\sum_{v\in V} \ln n \cdot \deg(v)\Big) = O(m \log n).$$

Using Fibonacci heaps [14] or Brodal's priority queues [9], the total running time of Algorithm 2 is $O(r + s\log n)$, where r is the number of relaxations, and s is the number of "insert" and "extract minimum" operations. In our cases $\mathbb{E}[r] = O(m\log n)$, and $\mathbb{E}[s] = O(n\log n)$. Therefore the total expected running time of Algorithm 2 is $O(m\log n + n\log^2 n)$. □

4 Hierarchical Partitions

In this section we explain how to dispense with the $O(\log\Phi)$ factor in the naïve implementation of the hierarchical partitions, and replace it with $O(\log n)$. The method being used is standard. Similar arguments appeared previously, *e.g.*, in [3, 15, 22, 21]. However, the context here is slightly different, and the designated time bound is $O\left(m\log^3 n\right)$, which is faster than the implementations we are aware of. While the argument is relatively straightforward, a full description of it is tedious to write and read. Instead we only sketch the implementation here. A complete description, including algorithmic implementation, appears in [24].

In a naïve implementation, the number of scales in which we sample CKR partitions is $\Theta(\lg\Phi)$. This leads to $O((n\log^2 n + m\log n)\log\Phi)$ bound on the expected running time. Here we develop an implementation having $O\left(m\log^3 n\right)$ expected running time. We define for each scale an appropriate quotient of the input graph. We then show that CKR partitions of those substitutive graph metrics retain the properties of CKR partitions on original metric. Using those quotients, not all scales need to be processed, and the total size of the quotient graphs in all processed scales is $O(m\lg n)$.

For $y, y' \subseteq X$, let $\rho(y,y') = \min\{\rho(x,x') \mid x\in y, x'\in y'\}$. Given a partition

Y of the space (X,ρ) we define the quotient metric ν on Y as

$$\nu\left(y,y'\right)=\min\Big\{\sum_{j=1}^{l}\rho\left(y_{j-1},y_j\right)\ :\ y_0,\ldots,y_l\in Y, y_0=y, y_l=y'\Big\}.$$

Definition 5. A space (Y,ν) is called Δ-*bounded quotient* of an n-point metric space (X,ρ) if Y is a Δ-bounded partition of X, ν is a quotient metric on Y, and for every $x\in X$, $B_\rho(x,\Delta/n)\subseteq Y(x)$.

Note that a Δ-bounded quotient of n-point metric space exists: define a relation $x\sim x'$ if $\rho(x,x')\le\Delta/n$, and take the transitive closure. The quotient subsets are the equivalence classes, and by the triangle inequality, the diameter of those equivalence classes is at most Δ.

The following lemma follows easily from Lemma 2.1, see the proof of [20, Lemma 5].

Lemma 4.1. *Fix $\Delta>0$, and let (Y,ν) be a $\frac{\Delta}{2}$-bounded quotient of (X,ρ). Let $\sigma:X\to Y$ be the natural projection, assigning each vertex $x\in X$ to its cluster $Y(x)$. Let L be a $(\Delta/2)$-bounded CKR partition of Y.*

Let P be the pullback of L under σ, i.e., $P=\left\{\sigma^{-1}(A)\ \middle|\ A\in L\right\}$. *Then P is a Δ-bounded partition of X such that for every $0<t\le\Delta/16$ and every $x\in X$,*

$$\Pr\left[B_\rho\left(x,t\right)\subseteq P\left(x\right)\right]\ge\left(\frac{\left|B_\rho\left(x,\Delta/16\right)\right|}{\left|B_\rho\left(x,\Delta\right)\right|}\right)^{\frac{32t}{\Delta}}. \tag{3}$$

and furthermore, if $t\le\Delta/2n$, then

$$\Pr\left[B_\rho\left(x,t\right)\subseteq P(x)\right]=1. \tag{4}$$

We define $G|_\Delta$ as the subgraph of G with edges of weight at most Δ and no isolated vertices.

Definition 6. Given a weighted graph $G=(X,E,\omega)$ and $\Delta>0$. Define the graph $G|_\Delta=(X|_\Delta,E|_\Delta,\omega|_\Delta)$ as follows.

$$\begin{aligned}
&E|_\Delta=\left\{(u,v)\in E\ :\ u\neq v,\ \text{and}\ \omega\left(u,v\right)\le\Delta\right\},\\
&X|_\Delta=\left\{u\in X\ :\ \exists v\in X,\ (u,v)\in E|_\Delta\right\},\ \text{and}\\
&\omega|_\Delta=\omega|_{E|_\Delta}.
\end{aligned}$$

Lemma 4.2. *Given a weighted graph $G=(X,E,\omega)$, and $\Delta>0$. Let L be a Δ-bounded CKR partition of $X|_\Delta$, using the metric induced by $G|_\Delta$. Then $P=L\cup\{\{v\}\ :\ v\in X\setminus(X|_\Delta)\}$ is a Δ-bounded CKR partition of X, using the metric induced by G.*

Proof. Let ρ be the shortest-path metric on G. Observe that when computing a Δ-bounded CKR partition of (X,ρ) no edge of weight larger than Δ is "used" by the Dijkstra's algorithm for computing the balls, and therefore discarding them does not change the behavior of the algorithm. Also, for each $v\in X\setminus X|_\Delta$, $B_\rho\left(v,\Delta\right)=\{v\}$, *i.e.*, in any Δ-bounded CKR partition of X, v will appear in a singleton subset. $\square$

Lemma 4.2 and Lemma 4.1 form the basis for dispensing with the dependence on the spread in the construction time. We next sketch the scheme we use. Denote the input graph $G = (X, E, \omega)$, $|X| = n$, $|E| = m$, and let ρ be the graph metric on G.

We first construct an ultrametric ν on V, represented by an ultrametric tree $H = (T, \Gamma)$ such that for every $u, v \in V$, $\rho(u, v) \le \nu(u, v) \le n\rho(u, v)$. H can be constructed in $O(m + n \log n)$ time using minimum spanning tree procedure, see [15, Section 3.2]. For a given $\Delta \ge 0$, and a leaf $v \in T$, denote by $\sigma_\Delta(v)$ the highest ancestor u of v for which $\Gamma(u) \le \frac{\Delta}{2n}$. Using the level-ancestor data structure (cf. [7]) the tree T can be preprocessed in $O(n)$ time such that queries for $\sigma_\Delta(v)$ (given Δ, and v) are answered in $O(\log n)$ time. See [15, Section 3.5] for a similar supporting data structure.

Given $\Delta > 0$, define the weighted graph $G_{(\Delta)}$ as follows. $G_{(\Delta)} = \left(X_{(\Delta)}, E_{(\Delta)}, \omega_{(\Delta)}\right)$ where,

$$\begin{aligned} X_{(\Delta)} &= \{\sigma_\Delta(v) :\ v \in X\}, \\ E_{(\Delta)} &= \{(\sigma_\Delta(u), \sigma_\Delta(v)) :\ (u, v) \in E, \sigma_\Delta(u) \ne \sigma_\Delta(v)\}, \\ \omega_{(\Delta)}(u, v) &= \min\{\omega(w, z) :\ \sigma_\Delta(w) = u,\ \sigma_\Delta(z) = v\}. \end{aligned}$$

Let $\rho_{(\Delta)}$ be the shortest-path metric on $G_{(\Delta)}$. Then, directly from the definitions, $\left(X_{(\Delta)}, \rho_{(\Delta)}\right)$ is a $\frac{\Delta}{2}$-bounded quotient of (X, ρ).

For an integer $j \ge -1$ denote $G_j = (V_j, E_j, \omega_j)$ where $G_j = \left(G_{(8^j)}\right)|_{8^j/2}$. The following lemma gives an upper bound on the total size of the graphs G_j.

Lemma 4.3.
$$\sum_{j \ge -1} (|V_j| + |E_j|) = O(m \lg n).$$

Proof. Fix $(u, v) \in E$ and $j \ge -1$ such that $(\sigma_{8^j}(u), \sigma_{8^j}(v)) \in E_{(8^j)}$. By the definition of $E_{(8^j)}$, $\sigma_{8^j}(u) \ne \sigma_{8^j}(v)$. By the definition of $\omega_{(8^j)}$, $\omega(u, v) \ge \omega_{(8^j)}(u, v) \ge \frac{8^j}{2n}$. Also, $(\sigma_{8^j}(u), \sigma_{8^j}(v)) \in E_j$ if and only if $\omega_{(8^j)}(\sigma_{8^j}(u), \sigma_{8^j}(v)) \le \frac{8^j}{2}$. So by the triangle inequality $\omega(u, v) \le \omega_{(8^j)}(\sigma_{8^j}(u), \sigma_{8^j}(v)) + 8^j$, and hence $\omega(u, v) \le 1.5 \cdot 8^j$. That is, each edge of G is represented in G_j only when $\omega(u, v) \in \left[\frac{8^j}{2n}, 1.5 \cdot 8^j\right]$. A total of $O(\log n)$ scales. By definition, G_j contains only non-isolated vertices, so $\forall j$, $|V_j| \le 2|E_j|$. □

Let **Processed** $= \{j \ge -1 :\ V_j \ne \emptyset\}$.

Lemma 4.4. *The set of graphs $(G_j)_{j \in \mathbf{Processed}}$ can be constructed in $O(m \log^2 n)$ expected time.*[6]

Sketch of a proof. First we sort the edges in $E = \{e_1, \dots e_m\}$ in non increasing order. Keep a "sliding window" $[i_L(t), i_R(t)]$, $i_L(t), i_R(t) \in \{1, \dots, m\}$, $t \in \{1, \dots, |\mathbf{Processed}|\}$, as follows: Let $j_1 = \lceil \log_8 \Phi \rceil$. $i_L(1) = 1$, $i_R(1) =$

[6]With a bit more care the running time can be improved to $O(m \log n)$. This improvement, however, will not improve the total construction time of the hierarchical partition.

$\max\{i: \ \omega(e_i) \geq 8^{j_1}/2n\}$. Assuming j_{t-1} is already defined, define $j_t = \max\{j < j_{t-1}: \ \exists i, \ 8^j \geq \omega(e_i) \geq 8^j/2n\}$, $i_L(t) = \min\{i: \ \omega(e_i) \leq 8^{j_t}\}$, and $i_R(t) = \max\{i: \ \omega(e_i) \geq 8^{j_t}/2n\}$. Note that $\{j_t\}_t =$ **Processed**, and the definition gives $O(m)$ time algorithm for computing the sequences $(j_t)_t$, $(i_L(t))_t$, and $(i_R(t))_t$. Constructing G_{j_t} can now be done in $O((i_R(t) - i_L(t) + 1)\log n)$ time, by observing that the set of vertices is

$$V_{j_t} = \{\sigma_{8^{j_t}}(u_i), \sigma_{8^{j_t}}(v_i): \ i \in \{i_L(t), \ldots, i_R(t)\}, \ (u_i, v_i) = e_i\},$$

and similarly the set of edges is

$$E_{j_t} = \{(\sigma_{8^{j_t}}(u_i), \sigma_{8^{j_t}}(v_i)): \ i \in \{i_L(t), \ldots, i_R(t)\}, \ (u_i, v_i) = e_i\}.$$

Another $\log n$ factor in the construction time comes from the $O(\log n)$ time needed for each query of the form "$\sigma_\Delta(u)$". Since $i_R(t) - i_L(t) + 1 = |E_{j_t}|$, by Lemma 4.3, $\sum_t (i_R(t) - i_L(t) + 1) = O(m \log n)$. □

Next, we sample $(8^{j_t}/2)$-bounded CKR partition L_{j_t} for each G_{j_t}. By Lemma 4.1, $(L_{j_t})_t$ (implicitly) represents CKR partitions of G in *all* scales. Using Theorem 1 and Lemma 4.3 computing $(L_{j_t})_t$ is done in $O(m \log^3 n)$ expected time.

Hierarchical partitions have an $O(n)$ storage representation. It is similar to an efficient ultrametric tree representation, such as the *nettree* in [15]. Using a rooted tree $\mathcal{P}$ whose leaves correspond to the points of X, each internal vertex u has at least two children, and is labeled with a (logarithm of) scale, $s(u)$. The 8^j-bounded partition P_j is now defined as follows: For $x \in X$, $P_j(x)$ is the highest ancestor u of x in $\mathcal{P}$ such that $s(u) \leq j$. Since the tree $\mathcal{P}$ does not have vertices of degree 2, except maybe the root, its size is $O(n)$.

We are left to describe how to compute the the common refinement of the pullbacks of $(L_{j_t})_t$ as a hierarchical partition represented in the tree structure $\mathcal{P}$ of the previous paragraph. This is done by top-down fashion as follows:

In the initialization step, $\mathcal{P}$ is created as a rooted star whose root, r is labeled by 8^{j_1}, and its leaves correspond to $\{\sigma_{8^{j_1}}(u): \ u \in X\}$.

Next, inductively assume that $\mathcal{P}$ is a hierarchical partition of $\{\sigma_{8^{j_{t-1}}}(u): u \in X\}$ corresponding to $\{L_{j_s}: \ s \leq t-1\}$. We refine $\mathcal{P}$ to include L_{j_t} as follows:

- Replace the leaves of $\mathcal{P}$: Each $\sigma_{8^{j_{t-1}}}(u)$ is replaced by

  $$\{\sigma_{8^{j_t}}(v): \ v \in X, \ \sigma_{8^{j_t}}(v) \text{ is a descendant of } \sigma_{8^{j_{t-1}}}(u)\}.$$

  This step is done in $O(|V_{j_t}|)$ time by simply starting from an "old leaf" $\sigma_{8^{j_{t-1}}}(u)$ as a vertex in T and descending in T to level $8^{j_t}/2n$.

- Next, incorporate L_{j_t} into the hierarchical partition in a straightforward way: Scan the leaves of $\mathcal{P}$, which are in V_{j_t} grouped by their parents. Fixing such a parent u whose children $v_1, \ldots, v_\ell$ are all leaves, we partition $v_1, \ldots, v_\ell$ to subsets $\{\{v_1, \ldots, v_\ell\} \cap C: \ C \in L_{j_t}\}$. For every such subset of size 2 or more we define a new parent w (which will be a child of u) with the label 8^{j_t}.

Hence, the t-th iteration in the algorithm above is executed in $O(|V_{j_t}|)$ time, so the total time for constructing the common refinement is $O(m \log n)$.

5 Applications

Proof of the first part of Theorem 2. As observed in Section 2, hierarchical partitions correspond to ultrametrics. As shown in [13], when the partition in every scale is a CKR partition, the resulting distribution over ultrametrics is a probabilistic embedding with $O(\log n)$ distortion.[7] The algorithm described in Section 4 samples a hierarchical partition (and hence an ultrametric) in $O(m \log^3 n)$ expected time. $\square$

Proof of the second part of Theorem 2. A point $x \in X$ is called β-padded in hierarchical partition $\mathcal{H} = (P_{-1}, \ldots, P_{\log \Phi})$, if for every j, $B(x, \beta 8^j) \subset P_j(x)$.

The main part of constructing $O(\beta^{-1})$-approximate distance oracle based on CKR partitions works as follows [22]: Set $X_0 = X$, and iteratively on $i = 0, 1, \ldots$ do: Compute a hierarchical CKR partition $\mathcal{H}_i$ of X_i, and obtain an ultrametric H_i from $\mathcal{H}_i$. Let Y_i be a set of β-padded points in $\mathcal{H}_i$ that is found in Lemma 2.2. Set $X_{i+1} \coloneqq X_i \setminus Y_i$, $i \coloneqq i + 1$ and repeat until $X_i = \emptyset$. The set of ultrametrics $(H_i)_i$, together with some supporting data-structures constitute the approximate distance oracle.

By Lemma 2.2, $\mathbb{E}|Y_i| \geq |X_i|^{1-32\beta}$, the number of iterations until $X_i = \emptyset$ is in expectation at most $O(\beta^{-1} n^{O(\beta)})$, and hence the total storage is as claimed.

There are two issues in the construction of $(H_i)_i$ that we have not covered yet: First, the task is to sample a hierarchical partition of X_i which is only a subset of the vertices in the graph $G = (X, E, \omega)$. This is rather easy to handle by adapting the algorithms in Section 3 and Section 4 to work with subsets of the vertices.

The second issue is the computation of β-padded points. The β-padded points of a (single) Δ-bounded partition P of a weighted graph $G = (V, E, \omega)$ can be computed as follows: Add a new vertex s_0. For every edge $(u, v) \in E$ such that $P(u) \neq P(v)$, add an edge (s_0, u) whose weight is $\omega(u, v)$. Execute Dijkstra's shortest path algorithm from s_0, and delete all vertices at distance at most $\beta\Delta$ from s_0. This can be implemented in $O(m + n \log n)$ time. Note that in the hierarchical partition if a point is not in V_j then it is padded at scale 8^j. Hence in order to compute a β padded point set in hierarchical partition, for every t, we cross out the points which are not 2β-padded in L_{j_t}. The remained points are β-padded in the pullbacks of $(L_{j_t})_t$ (as follows from Lemma 4.1) and hence also in the hierarchical partition. When implemented carefully, this can

[7]Technically, in [13] the hierarchical partition was built differently: instead of taking a CKR partition of the whole space in every scale, and then the common refinement, at each scale they took many CKR partitions, one for each block of the partition of the previous scale. This technicality is inconsequential for probabilistic embeddings. However, for the construction of approximate distance oracles, the approach of [13] does not seem to work since it does not have stochastically independent partitions in the different scales. See also [22].

be done on every graph G_j in $O(|E_j|+|V_j|\log n)$, and by Lemma 4.3, in a total $O(m\log^2 n)$ time. $\square$

A t-spanner of a weighted graph $G=(V,E,\omega)$, is a subset of the edges $E'\subset E$ such that the shortest-path metric on $(V,E',\omega|_{E'})$ is at most t times the shortest-path metric on G. We need the following result.

Theorem 7 ([6]). *Let $G=(V,E,w)$ be a weighted graph with n vertices and m edges, and let $k\geq 1$ be an integer. A $(2k-1)$-spanner of with $O\left(kn^{1+1/k}\right)$ edges can be computed in $O\left(km\right)$ expected time.*

Proof of Theorem 3. By Theorem 7, given an n-point metric space (X,ρ), a 5-spanner H of (X,ρ) with $O(n^{4/3})$ edges can be constructed in $O(n^2)$ time. We next apply the second part of Theorem 2 whose running time is $o(n^2)$. $\square$

References

[1] Ittai Abraham, Yair Bartal, and Ofer Neiman. Advances in metric embedding theory. In Jon M. Kleinberg, editor, *STOC*, pages 271–286. ACM, 2006.

[2] Noga Alon, Richard M. Karp, David Peleg, and Douglas West. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comput.*, 24(1):78–100, 1995.

[3] Yair Bartal. Probabilistic approximations of metric space and its algorithmic application. In *Proc. 37th Ann. IEEE Symp. Foundations of Computer Science (FOCS'96)*, pages 183–193. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, 1996.

[4] Yair Bartal. Graph decomposition lemmas and their role in metric embedding methods. In Susanne Albers and Tomasz Radzik, editors, *ESA*, volume 3221 of *Lecture Notes in Computer Science*, pages 89–97. Springer, 2004.

[5] Yair Bartal. Embedding finite metric spaces in low dimension. Technical report, Hebrew University of Jerusalem, 2005.

[6] Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Struct. Algorithms*, 30(4):532–563, 2007.

[7] Michael A. Bender and Martín Farach-Colton. The level ancestor problem simplified. *Theor. Comput. Sci.*, 321(1):5–12, 2004.

[8] David Berman and M. S. Klamkin. A reverse card shuffle. *SIAM Review*, 18(3):491–492, 1976.

[9] Gerth Stølting Brodal. Worst-case efficient priority queues. In *SODA '96: Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 52–58, Philadelphia, PA, USA, 1996. Society for Industrial and Applied Mathematics.

[10] Gruia Calinescu, Howard Karloff, and Yuval Rabani. Approximation algorithms for the 0-extension problem. In *Proc. 12th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA'01)*, pages 8–16. SIAM, Philadelphia, PA, USA, 2001.

[11] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1:269–271, 1959.

[12] Jittat Fakcharoenphol, Chris Harrelson, Satish Rao, and Kunal Talwar. An improved approximation algorithm for the 0-extension problem. In *Proc. 14th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA'03)*, pages 257–265. ACM, New York, NY, USA, 2003.

[13] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. System Sci.*, 69(3):485–497, 2004.

[14] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987.

[15] Sariel Har-Peled and Manor Mendel. Fast construction of nets in low dimensional metrics, and their applications. *SIAM J. Comput.*, 35:1148–1184, 2006.

[16] Robert Krauthgamer, James R. Lee, Manor Mendel, and Assaf Naor. Measured descent: A new embedding method for finite metrics. In *Proc. 45th Ann. IEEE Symp. Foundations of Computer Science (FOCS'04)*, pages 434–443. IEEE Computer Society, Washington, DC, USA, 2004.

[17] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

[18] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[19] Nathan Linial and Michael Saks. Low diameter graph decompositions. *Combinatorica*, 13(4):441–454, 1993.

[20] Manor Mendel and Assaf Naor. Maximum gradient embeddings and monotone clustering, 2006. preliminary version of [21]. Accepted to Combinatorica. `arXiv:cs/0606109`.

[21] Manor Mendel and Assaf Naor. Maximum gradient embeddings and monotone clustering. In *Proc. Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 10th Int. Workshop (APPROX'07) and 11th Int. Workshop (RANDOM'07)*, pages 242–256. Springer, Berlin, Germany, 2007.

[22] Manor Mendel and Assaf Naor. Ramsey partitions and proximity data structures. *J. European Math. Soc.*, 9(2):253–275, 2007.

[23] Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proc. 40th Ann. ACM Symp. Theory of computing (STOC'08)*, pages 255–264. ACM, New York, NY, USA, 2008.

[24] Chaya Schwob. Ramsey partitions based approximate distance oracles. Master's thesis, The Open University of Israel, Ra'anana, Israel, 2008. available at `http://sites.google.com/site/mendelma/Home/students/RamseyPartitionsBasedADO_Final.pdf?attredirects=0`.

[25] Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005.