Well-Separated Pair Decompositions for Doubling Metric Spaces

By

Daming Xu

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario

August 2005

© Copyright
2005, Daming Xu

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-494-10123-7
Our file Notre référence
ISBN: 0-494-10123-7

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Abstract

The Well-Separated Pair Decomposition has been widely used for solving proximity problems in Euclidean space. Theoretically, it can be extended to any metric space. But in fact, only few work has been done on such extensions. We extend the classic well-separated pair decomposition to the robust class of doubling metric spaces. We present a deterministic algorithm that constructs a Well-Separated Pair Decomposition in any doubling metric space. The number of pairs in this decomposition is $O(n \log \Delta)$, where n is the number of elements and Δ is the aspect ratio of the metric space.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Acknowledgements

I would like to express my gratitude to all those who gave me the possibility to complete this thesis.

I would like to express my sincere gratitude to my thesis supervisor Dr. Michiel Smid for the opportunity to explore the field of Computational Geometry. His encouragement, guidance, and support has helped me in all the time of research and writing of this thesis.

I would also like to give my special thanks to my wife Qin Wang. Without her patient love, it would have been impossible for me to complete this work.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Contents

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Figures

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 1

Introduction

This chapter presents motivation, scope of the research and organization of this thesis.

1.1 Motivation

Many problems on metric spaces can be ultimately solved by computing the distances between all pairs of elements. But computing all distances can be very expensive since the metric space can be intrinsically complicated. For example, in a metric space induced by a graph, even for a planar graph, computing the shortest path distance between any two nodes will need at least linear time [30]. Thus, how to avoid computing all distances in metric spaces becomes an attractive topic.

Callahan and Kosaraju [7] firstly defined the notion of Well-Separated Pair Decomposition (WSPD) in d-dimensional Euclidean space. A pair of point sets (S_1, S_2), $S_1, S_2 \subseteq \mathbb{R}^d$, is c-Well-Separated if the distance between S_1 and S_2 is at least c times the diameter of both S_1 and S_2 (see Figure 1.1). A c-WSPD of a set S of n points in $\mathbb{R}^d$ is a collection of pairs (S_1, S_2) of subsets of S that are c-Well-Separated and

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

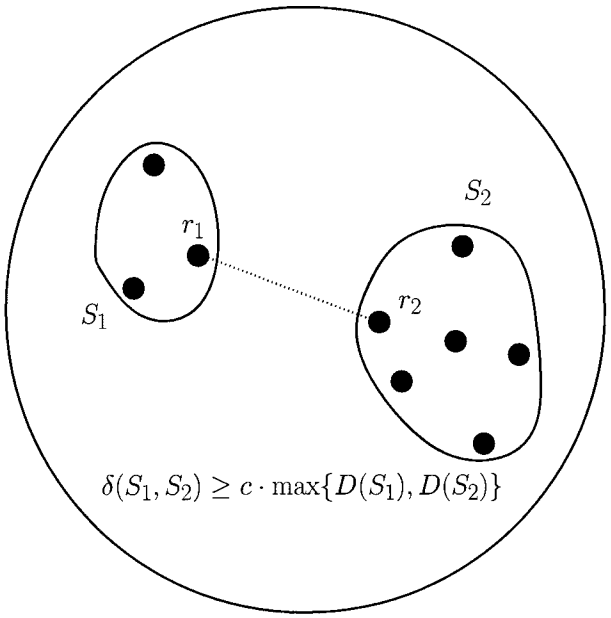


Figure 1.1: Illustration of two Well-Separated sets

that cover all pairs of points in S. The main idea in the Well-Separated Pair Decomposition is that for a pair (S_1, S_2) of Well-Separated sets, the distance between any point in S_1 and any point in S_2 can be approximated by the distance between two representative points of S_1 and S_2. That means we compress a point set to its representative point. Thus checking $\Theta(n^2)$ pairwise distances can be approximately solved by examining distances between the Well-Separated pairs.

Given any constant $c \geq 1$, Callahan and Kosaraju show how to compute a c-WSPD in Euclidean space, where the number of pairs is $O(n)$ and the running time is $O(n \log n)$. This c-WSPD basically consists of $O(n)$ pairs of points that approximate all $\binom{n}{2}$ distances in S.

Gao and Zhang [16] extend the definition of c-WSPD to the unit-disk graph metric space. They present a procedure to produce a c-WSPD in such a metric space, where the number of pairs is $O(n \log n)$ and the running time is $O(n \log n)$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Talwar [42] and Har-Peled and Mendel [24] extend the WSPD to doubling metric space. They present *randomized* algorithms for computing a c-WSPD in such spaces. The algorithm in [42] produces a c-WSPD consisting of $O(n \log \Delta)$ pairs, where

$$\Delta = \frac{\max_{x,y \in S} \delta(x,y)}{\min_{x,y \in S, x \neq y} \delta(x,y)}$$

is the aspect ratio of the metric space S and δ is the distance function; but no analysis of the running time is given. The algorithm in [24] produces a c-WSPD consisting of $O(n)$ pairs, in $O(n \log n)$ expected time.

The Well-Separated Pair Decomposition has found numerous applications in solving proximity problems in Euclidean space [1,2,4,6,7,13,20,35,37]. It has also been shown very useful for obtaining efficient dynamic, parallel and external memory algorithms [5–7,13,19]. For example, the Well-Separated Pair Decomposition can be used to compute closest pair, all nearest neighbors, spanners and approximate minimum spanning trees [5–7].

1.2 Scope of the Research

This thesis focuses on developing *deterministic* algorithms for constructing a c-WSPD in doubling metric spaces. Doubling metric spaces form a robust class of metric spaces which contains many families of metrics that occur in applied settings [22]. Understanding and applying such metric spaces received considerable attention in the last few years [9,17,22,24–26,29,31,32,39,41,42]. For example, the communication graphs of wireless networks and peer-to-peer networks often satisfy the properties of doubling metric spaces. Thus, studying doubling metric space will be very essential for developing load balanced routing and finding nearest neighbors (nodes) in those graphs.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Suppose (S, δ) is a metric space. The *doubling parameter (constant)* is the smallest value λ such that every ball B in S can be covered by λ balls of half the radius of B. The *doubling dimension* of S is defined as $dim(S) = \log_2 \lambda$. If the doubling dimension of (S, δ) is bounded, it's called a doubling metric space.

Doubling metric spaces extend the notion of growth restricted metric spaces of [17, 29], which have several practical applications such as peer-to-peer networks [38] and data analysis [28]. If we could embed a doubling metric space into a low dimensional Euclidean space with a small distortion, we would be able to exploit the well studied structural and algorithmic properties of these spaces. However, this is not always possible, because there are metric spaces that cannot be embedded with a small distortion in any Hilbert space [33,34]. In fact, Gupta et al. [22] show that there exist doubling metric spaces with n points having the property that any embedding in a Euclidean space has distortion $\Omega(\sqrt{\log n})$.

The concept of Well-Separated Pair Decomposition can be naturally extended to doubling metric space without embedding. Thus embedding metric spaces into low dimensional Euclidean space with low distortion is not the topic of this thesis. Our attention is mainly focused on how to give the metric space a hierarchical decomposition and efficiently check whether the resulting subsets are Well-Separated. Finding an easy way to approximately compute the distance between two subsets is another important problem when constructing a Well-Separated Pair Decomposition.

In this thesis we introduce an intuitive approach that builds a series of ball coverings of the original doubling metric space with different radii. This series of ball coverings gives the metric space a hierarchical decomposition and greatly reduces the number of pairs to be checked. As a result, we obtain a deterministic algorithm that constructs a c-WSPD, consisting of $O(n \log \Delta)$ pairs, where Δ is the aspect ratio of the doubling metric space. Unfortunately, the running time for constructing c-WSPD

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

depends on the time to build the ball coverings.

1.3 Organization of the Thesis

The rest of this thesis is organized as follows:

In Chapter 2, we focus on introducing doubling metric spaces. We begin with the definition of doubling metric spaces. Then we prove a basic result that allows us to obtain an upper bound on the size of any set of elements whose aspect ratio is "small". We also present an algorithm that covers the metric space by a limited number of balls of any given radius.

In Chapter 3, we present the definition of Well-Separated Pair Decomposition (WSPD). Then we review the previous work achieved by Callahan and Kosaraju [7] in Euclidean space and Gao and Zhang [16] in the unit-disk graph metric space.

In Chapter 4, we introduce an intuitive approach for constructing a WSPD for doubling metric spaces. We first give the outline of our algorithm and then describe the process of our algorithm in detail.

In Chapter 5, we conclude this thesis and discuss future work.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 2

Doubling Metric Spaces

In this chapter we define metric spaces, as well as the subclass of doubling metric spaces. We also present a greedy algorithm for covering a doubling metric space by balls of equal radius, and prove an upper bound on the number of balls in this covering.

2.1 Metric Spaces

Definition 2.1.1. *Let S be a set, and let $\delta : S \times S \to \mathbb{R}$ be a function. The pair (S, δ) is called a* ***metric space****, if for all $x, y, z \in S$:*

1. $\delta(x, y) \geq 0$;
2. $\delta(x, y) = 0$ *if and only if* $x = y$;
3. $\delta(x, y) = \delta(y, x)$;
4. $\delta(x, z) \leq \delta(x, y) + \delta(y, z)$.

The fourth requirement is called the triangle inequality.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In this thesis, S is finite and we denote number of elements of S by n.

For example, if S is a finite set of points in $\mathbb{R}^d$, and $\delta(x, y)$ is the Euclidean distance between x and y in $\mathbb{R}^d$, then (S, δ) constructs a Euclidean metric space. To give another example, let $G = (S, E)$ be a weighted graph, and $\delta(x, y)$ is length of the shortest path in G between x and y, then (S, δ) constructs a graph metric space.

The *distance* between two subsets A and B of S is defined as

$$\delta(A, B) = \min\{\delta(a, b) : a \in A, b \in B\}$$

The *diameter* of a subset A of S is defined as

$$D(A) = \max\{\delta(a, b) : a, b \in A\}$$

If $x \in S$ and $r \geq 0$, then $B(x, r) = \{y \in S : \delta(x, y) \leq r\}$ is the *ball* with center x and radius r.

The *aspect ratio* of the metric space (S, δ) is defined as

$$\Delta = \frac{\max_{x,y \in S} \delta(x, y)}{\min_{x,y \in S, x \neq y} \delta(x, y)}$$

An *r-cover* of a metric space (S, δ) is a set Y of points such that $S \subseteq \bigcup_{y \in Y} B(y, r)$.

An *r-packing* of a metric space (S, δ) is a set Y of points such that $\delta(y, y') > r$ for all $y, y' \in Y$ with $y \neq y'$.

An *r-net* of a metric space (S, δ) is a set Y of points such that Y is both an r-cover and an r-packing.

2.2 Doubling Metric Spaces

Definition 2.2.1. *Let (S, δ) be a metric space. The **doubling parameter** of (S, δ) is defined to be the smallest real number λ such that for every real number $r > 0$, the*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

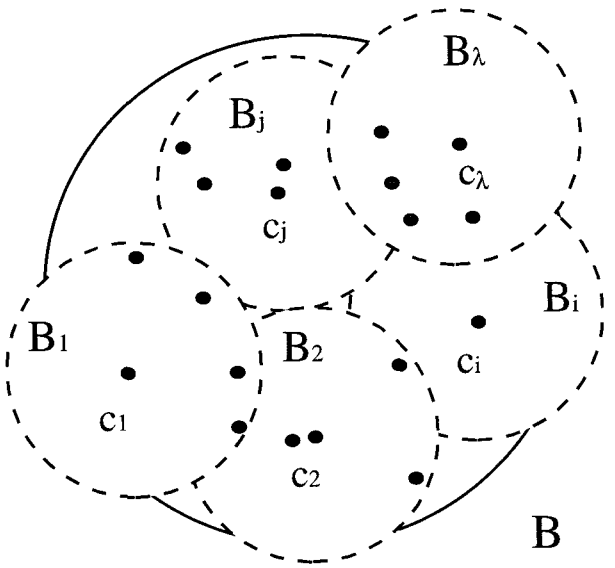


Figure 2.1: A covering of a ball B with radius r by λ balls $B_1, \cdots, B_\lambda$ of radius $r/2$.

following holds: Every ball in S of radius r can be covered by λ balls of radius $r/2$ (See Figure 2.1). The ***doubling dimension*** *of (S, δ) is defined to be $dim(S) = \log_2 \lambda$. If the value of λ is independent of the size of S, then we call (S, δ) a* ***doubling metric space.***

Assouad [3] and Heinonen [25] firstly presented the definition of doubling dimension and showed that this notion contains several natural properties. The doubling dimension can be thought as a generalization of the Euclidean dimension in Euclidean space. In Section 2.4, we will prove $dim(S) = \log_2 \lambda \leq d \cdot \log_2 5$ for $\mathbb{R}^d$.

Not all metric spaces are doubling metric spaces. There exist some metric spaces whose doubling parameter λ depends on the size of S. For example, let (S, δ) be the metric space, $|S| = n$, where, for all $x, y \in S$,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$\delta(x,y) = \begin{cases} 0 & if \quad x = y \\ 1 & if \quad x \neq y \end{cases}$$

Let $x \in S$ and consider the ball B with center x and radius 1. Then all the points of S will be covered by B. But we have to use n balls with radius $\frac{1}{2}$ to cover B since any two points will not be covered by a ball with radius $\frac{1}{2}$. Thus, the doubling parameter λ of (S,δ) depends on the size of S and (S,δ) is not a doubling metric space.

Doubling metric spaces is a robust class of metric spaces which contains many families of metrics that occur in applied settings [22]. Doubling metric spaces extend the notion of growth restricted metric spaces of [17,29], which have several practical applications such as peer-to-peer networks [38] and data analysis [28]. Understanding and applying such metric space received considerable attention in the last few years [9,17,22,24–26,29,31,32,39,41,42].

Proposition 1 *Let (S,δ) be a metric space with doubling parameter λ, let Y be a subset of S, and let $\Delta(Y) = \frac{\max_{x,y\in Y}\delta(x,y)}{\min_{x,y\in Y, x\neq y}\delta(x,y)}$. Then $|Y| \leq \lambda^{2+\lfloor \log \Delta(Y)\rfloor}$.*

Proof. Without loss of generality, assume $\min \delta(x,y) = 1$, and thus $\Delta(Y) = \max \delta(x,y)$. Let B be a ball of radius $\Delta(Y)$ that covers Y. By repeatedly applying Definition 2.2.1: For $i \geq 0$, B can be covered by λ^i balls of radius $\Delta(Y)/2^i$. The diameter of any ball of radius $\Delta(Y)/2^i$ is at most $\Delta(Y)/2^{i-1}$. Let $i = 2 + \lfloor \log \Delta(Y)\rfloor$. Since $\lfloor \log \Delta(Y)\rfloor > \log \Delta(Y) - 1$, we have $i > 1 + \log \Delta(Y)$, $2^{i-1} > \Delta(Y)$, and $\Delta(Y)/2^{i-1} < 1$. Since B can be covered by $\lambda^{2+\lfloor \log \Delta(Y)\rfloor}$ balls of diameter less than 1, each such ball can contain only one point of Y, thus $|Y| \leq \lambda^{2+\lfloor \log \Delta(Y)\rfloor}$. □

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.3 A Ball Covering Algorithm

From Definition 2.2.1, we know that any ball of radius r can be covered by λ balls of radius $r/2$. But how to compute this covering is still unclear. Now, we consider a general case. Let S be an element set and S is contained in a ball of radius R. In this section we introduce a greedy ball-covering algorithm that computes a covering consisting of at most $\lambda^{\lfloor \log \frac{2R}{r} \rfloor}$ balls of radius r ($r \leq R$).

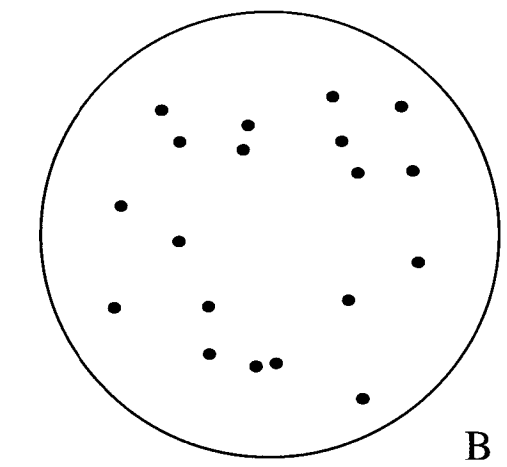
For example, suppose S is contained in a ball B with radius R in a metric space (See Figure 2.2(a)). We first choose an arbitrary element, say c_1, and compute the ball B_1 with center c_1 and radius r (See Figure 2.2(b)). B_1 will cover several elements of B. Next we choose another element which has not been covered by B_1, say c_2, and compute the ball B_2 with center c_2 and radius r (See Figure 2.2(c)). The elements in the intersection of B_1 and B_2 will be assigned to the former. Then we keep doing the above process and finally we obtain a ball-covering with t balls $B_1, B_2, \cdots, B_t$ of radius r (See Figure 2.2(d)).

The ball covering algorithm is as follows:

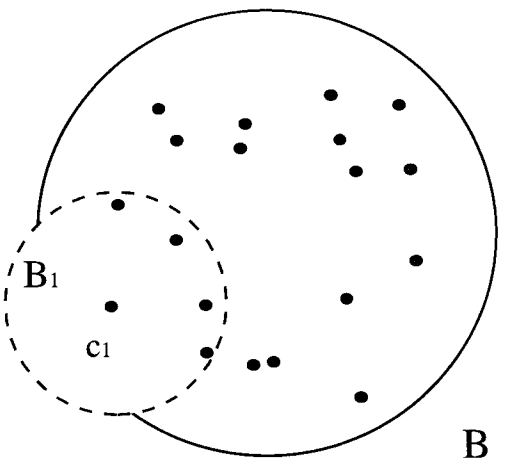
Algorithm *BallCovering* (S, r)
Input: An element set S and a radius r
Output: A covering $C = (B_1, \cdots, B_t)$ with t balls $B_1, B_2, \cdots, B_t$ of radius r

1. Let B be a ball of radius R such that S is contained in B
2. **if** $r > R$
3. **then return** nil
4. $C = \emptyset$, $X = B$
5. **while** $X \neq \emptyset$
6. **do** choose an arbitrary element $x \in X$, compute the ball B' with center x and radius r, add $B' \cap X$ to the covering C, and set $X = X \setminus B'$

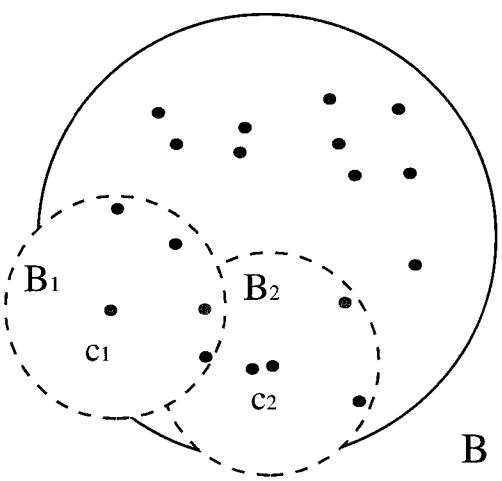
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



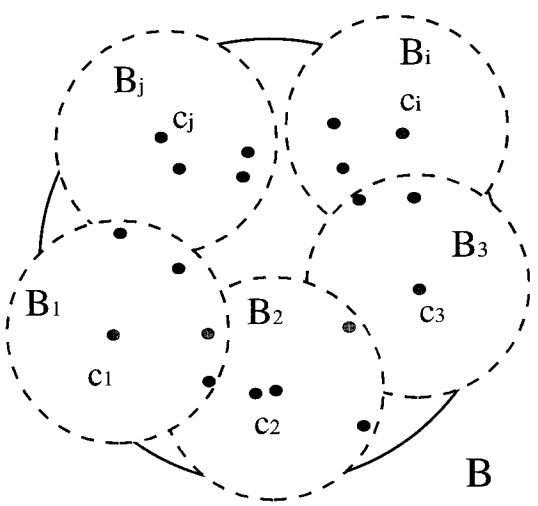
(a) S is contained in a ball B of radius R



(b) The first ball B_1 with radius r



(c) The second ball B_2 with radius r



(d) The resulting ball-covering

Figure 2.2: Constructing a ball-covering

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

7. **return** C.

Let $Y = \{c_1, c_2, \cdots, c_t\}$ be the set of centers of $B_1, B_2, \cdots, B_t$. Then we have $\Delta(Y) \leq \frac{2R}{r}$, and thus, from Proposition 1, $t \leq \lambda^{2+\lfloor \log \frac{2R}{r} \rfloor}$. However, the following Lemma 2.3.1 proves a better bound on t.

Lemma 2.3.1. *Let $B_1, B_2, \cdots, B_t$ be the output of the above ball covering algorithm, then $t \leq \lambda^{\lfloor \log \frac{2R}{r} \rfloor}$.*

Proof. Let B be a ball of radius R. From Definition 2.2.1, B can be covered by $\lambda^{\lfloor \log \frac{2R}{r} \rfloor}$ balls of radius $r/2$.

Since the diameter of each ball B_k is at most r, and since $\delta(c_i, c_j) > r$ for all $1 \leq i, j \leq t, i \neq j$, it follows that each ball B_k contains at most one of the centers $c_1, \cdots, c_t$. Thus, $t \leq \lambda^{\lfloor \log \frac{2R}{r} \rfloor}$. □

For example, in any doubling metric space, if (S, δ) can be represented by its $n \times n$ distance matrix, the running time of the above algorithm is $O(n + \lambda^{\lfloor \log \frac{2R}{r} \rfloor} m)$, where $m = |B|$. In the worst case, when $m = n$, the running time will be $O(\lambda^{\lfloor \log \frac{2R}{r} \rfloor} n)$.

In a special case, when (S, δ) is a graph metric space, $G = (S, E)$ is a weighted graph with n vertices and m edges. We apply Single-Source Dijkstra algorithm to compute ball B' of radius r. The running time is

$$O(|B'| \log |B'| + \sum_{v \in B'} deg(v))$$

Thus, the total time for the algorithm is

$$O(\sum_{i=1}^{t} (|B_i| \log |B_i| + \sum_{v \in B_i} deg(v)))$$

.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

From Lemma 2.3.1, $t \leq \lambda^{\lfloor \log \frac{2R}{r} \rfloor}$,

$$\begin{aligned} & \textstyle\sum_{i=1}^{t} |B_i| \log |B_i| \\ \leq & \textstyle\sum_{i=1}^{t} |B_i| \log n \\ \leq & t * \max_{1 \leq i \leq t} |B_i| \log n \\ \leq & tn \log n. \end{aligned}$$

And

$$\sum_{i=1}^{t} \sum_{v \in B_i} deg(v) \leq t * \max_{1 \leq i \leq t} \sum_{v \in B_i} deg(v) \leq t * 2m$$

Thus, the total time is $O(tn \log n + tm) = O(\lambda^{\lfloor \log \frac{2R}{r} \rfloor} n \log n + \lambda^{\lfloor \log \frac{2R}{r} \rfloor} m)$.

2.4 The Doubling Dimension of Euclidean Space

As illustrated in Figure 2.3, let B be a ball with center x and radius r in the Euclidean metric space $\mathbb{R}^d$. Let $B_1, \cdots, B_t$ be the balls of radius $\frac{r}{2}$ that cover B obtained from the above ball covering algorithm. Let $c_1, \cdots, c_t$ be the centers of $B_1, \cdots, B_t$, respectively. Then $\delta(c_i, c_j) > \frac{r}{2}$ for all $i \neq j$. Let B_i' be the ball with center c_i and radius $r/4$. Then $B_i', 1 \leq i \leq t$ are pairwise disjoint. Let B' be a ball with center x and radius $\frac{5}{4}r$. Then $B_1', \cdots, B_t'$ are contained in B'. Thus, $t \leq \frac{volume(B')}{volume(B_i')} = \frac{c \cdot (\frac{5}{4}r)^d}{c \cdot (\frac{r}{4})^d} = 5^d$. Since $\lambda \leq t \leq 5^d$, $dim(S) = \log_2 \lambda \leq d \cdot \log_2 5$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

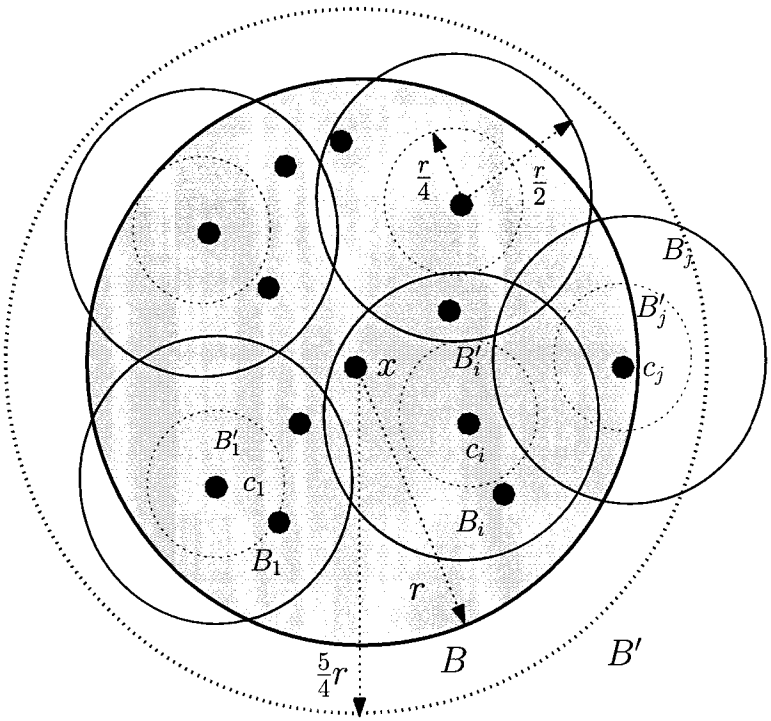


Figure 2.3: Example: the doubling dimension of Euclidean space

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 3

The Well-Separated Pair Decomposition

In this chapter we will define and review previous work on the Well-Separated Pair Decomposition (WSPD). First, we present the definition of the WSPD. Then we review the previous work of Callahan and Kosaraju [7] for Euclidean spaces and Gao and Zhang [16] for the unit-disk graph metric space.

3.1 Definition of the Well-Separated Pair Decomposition

Definition 3.1.1. *Let (S, δ) be a metric space. Let $c > 0$ be a real number. Two non-empty subsets S_1 and S_2 of S are called* ***c-Well-Separated*** *if $\delta(S_1, S_2) \geq c \cdot \max(D(S_1), D(S_2))$.*

Definition 3.1.2. *For any two subsets A and B of S, a set of pairs $P = \{P_1, P_2, \cdots, P_m\}$, where each $P_i = (A_i, B_i)$, is called a* ***pair decomposition*** *of (A, B) if:*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1. for all $1 \leq i \leq m$, $A_i \subseteq A$, $B_i \subseteq B$;

2. for all $1 \leq i \leq m$, $A_i \cap B_i = \emptyset$;

3. for every $a \in A$ *and every* $b \in B$, *there exists a unique index* i *such that* $a \in A_i$, $b \in B_i$. *We say that* (a, b) *is covered by the pair* $P_i = (A_i, B_i)$.

Definition 3.1.3. *Let* $c > 0$ *be a real number, let* A *and* B *be subsets of* S, *and let* P *be a pair decomposition of* (A, B). *If every pair in* P *is* c*-Well-Separated,* P *is called a* c-***Well-Separated Pair Decomposition*** *(*c-***WSPD***) of* (A, B).

Definition 3.1.4. *A* c*-WSPD of the metric space* (S, δ) *is a* c*-WSPD of* (S, S).

3.2 Well-Separated Pair Decomposition in Euclidean Space

Callahan and Kosaraju were the first to formally define the Well-Separated Pair Decomposition in Euclidean Space. They show that a Well-Separated Pair Decomposition of linear size can be computed in $O(n \log n)$ time. In this section, we review their approach.

Definition 3.2.1. *Let* S *be a set of* n *points in* $\mathbb{R}^d$, *where* d *is a constant denoting the dimension. The* ***bounding rectangle*** *of* S, *denoted by* $R(S)$ *is defined to be the smallest rectangle that encloses all points in* S, *where the word "rectangle" denotes any Cartesian product* $R = [x_1, x_1'] \times [x_2, x_2'] \times \cdots \times [x_d, x_d']$ *in* $\mathbb{R}^d$. *The* ***open rectangle*** *is defined to be the corresponding product of open intervals.*

The length of R *in the* i*th dimension is defined as* $l_i(R) = x_i' - x_i$. *The maximum and minimum length is denoted by* $l_{max}(R)$, $l_{min}(R)$. *When all* $l_i(R)$ *are equal,* R *is called a* d*-cube and we denote its length by* $l(R) = l_{max}(R) = l_{min}(R)$. *A* d*-ball of*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

radius r is defined as the set of all points in $\mathbb{R}^d$ that are at a distance less than or equal to r from a fixed point, called the center of the ball.

Two point sets A and B are c-well-separated if $R(A)$ and $R(B)$ can each be contained in d-balls of the same radius r whose minimum distance is at least $2cr$, where $c > 0$ is the separation.

The above definition on c-well-separated is equivalent to those of Section 3.1. Here A and B are contained in d-balls of the same radius r, then $\max(D(A), D(B)) \leq 2r$. Thus $\delta(A, B) \geq 2cr \geq c \cdot \max(D(A), D(B))$.

3.2.1 The Fair Split Tree

A *split* of S is defined to be its partition into two nonempty point sets lying on either side of a hyperplane (called the splitting hyperplane) perpendicular to one of the coordinate axes, and not intersecting any points in S.

A *split tree* of S is defined to be a binary tree constructed recursively as follows:

If $|S| = 1$, its unique split tree consists of the node S. Otherwise, a split tree is any tree with root S and two subtrees that are split trees of the subsets formed by a split of S. For any node A in the tree, its parent (if it exists) is denoted by $p(A)$.

A *fair split* of A is defined to be a split of A in which the splitting hyperplane is at a distance of at least $l_{max}(R(A))/3$ from each of the two boundaries of $R(A)$ parallel to it. A split tree formed using only fair splits is called a *fair split tree.*

It is clear that a fair split tree can be constructed in $O(n^2)$ time. A faster algorithm to find a fair split tree for any point sets is as follows:

Presort the points of point set S by their coordinates. For each coordinate, maintain a doubly-linked list of points in increasing order with cross-references.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Determine the split location in this list using a linear search that alternates between both ends of the list, heading toward the middle. The search time will be proportional to the size of the smaller piece.

Delete all of the points in the smaller piece from all of the lists. As deleting each point, we mark it with a pointer back to each list, but we do not insert it yet.This phase also requires time proportional to the size of the smaller piece.

Repeat the above splitting process on the remaining collection of lists until the resulting set of points is no more than half the size of the original. As doing this, we can construct a partial split tree consisting of a linear chain starting with the whole set and ending with the set of points left in the final step. Each node in this chain has an additional subtree consisting of the split tree of the set of points deleted at that step. Note that the amount of work is proportional to the number of points deleted, and hence is linear.

The set of points deleted at a step is always no more than half the size of the original set. Hence, if we can construct sorted lists for all of these point-sets in linear time, we can continue the process recursively and obtain $O(n \log n)$ time by a divide-and-conquer approach. To construct all the lists for all the deleted points sorted by some coordinate, we return to the original list of points sorted by that coordinate (which we copied prior to the deletions), and traverse it in increasing order.

Each time we find a point, we append it to the end of the appropriate list, given by the pointer that was set at the time of deletion. There are a constant number of coordinates, so this phase takes linear time.

Thus, the time complexity, including presorting, is clearly $O(n \log n)$.

Theorem 3.2.2. *Given a set S of n points in $\mathbb{R}^d$, a fair split tree of S can be constructed in $O(n \log n)$ time.*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.2.2 Computing Well-Separated Pairs

The following is the algorithm of computing WSPD in Euclidean metric space.

Algorithm *WSPD Algorithm (Euclidean Metric Space)*

Input: A point set S with a fair split tree T

Output: c-WSPD P

1. Initially, $A = B = S$, place (A, B) in a queue Q
2. **while** $Q \neq \emptyset$
3. **do** remove the first element (S_1, S_2) from Q
4. **if** S_1 and S_2 are c-Well-Separated
5. **then** add (S_1, S_2) to P
6. **else if** $l_{\max}(S_1) \geq l_{\max}(S_2)$ and $|S_1| > 1$
7. **then** add to Q two pairs (S_{11}, S_2) and (S_{12}, S_2) where S_{11}, S_{12} are two children of S_1 in T
8. **else if** $l_{\max}(S_2) \geq l_{\max}(S_1)$ and $|S_2| > 1$
9. **then** add to Q two pairs (S_1, S_{21}) and (S_1, S_{22}) where S_{21}, S_{22} are two children of S_2 in T
10. **return** P.

Callahan and Kosaraju show that the number of pairs in a c-WSPD is $O(n)$ by the following lemma.

Lemma 3.2.3. *Let S be a point set with an associated fair split tree T. Then S has a c-Well-Separated realization of size $O(n)$ that uses T.*

Finally, the following theorem can be proved:

Theorem 3.2.4. *Given a set S of n points in $\mathbb{R}^d$ and a fair split tree of S, a c-Well-Separated Pair Decomposition of S can be constructed in $O(n)$ time.*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Combing Theorem 3.2.1 and 3.2.4, we can obtain the following theorem:

Theorem 3.2.5. *Given a set S of n points in $\mathbb{R}^d$, a c-Well-Separated Pair Decomposition of S can be constructed in $O(n \log n)$ time, and the number of pairs is $O(n)$.*

3.3 Well-Separated Pair Decomposition in Unit-Disk Graph Metric Spaces

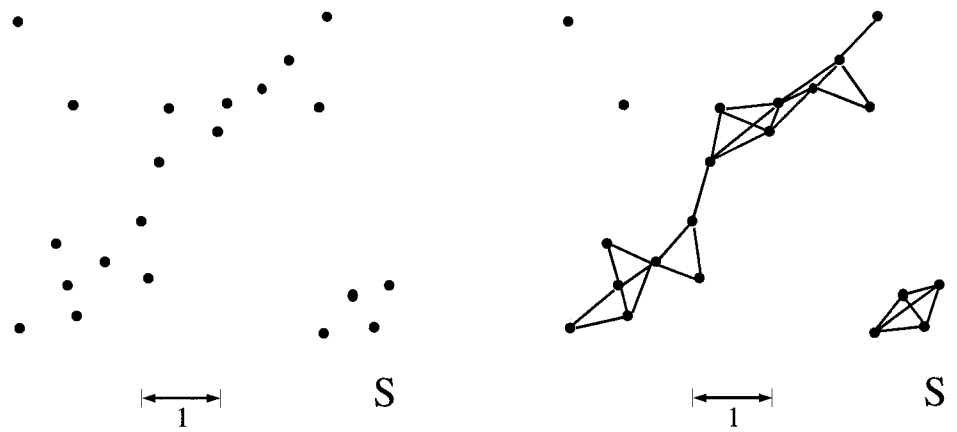
Gao and Zhang [16] extend the classic notion of Well-Separated Pair Decomposition to the unit-disk graph metric: the shortest path distance metric induced by the intersection graph of unit disks. They show that for the unit-disk graph metric of n points in the plane and for any constant $c \geq 1$, there exists a c-Well-Separated Pair Decomposition with $O(n \log n)$ pairs, and the decomposition can be computed in $O(n \log n)$ time. In this section, we review their approach.

Definition 3.3.1. *Denote by $|\ ,\ |$ the Euclidean metric. For a set of points S in the plane, the unit-disk graph $I(S) = (S, E)$ is defined as the weighted graph where an edge $e = (p, q)$ is in the graph if $|p, q| \leq 1$, and the weight of e is $|p, q|$ (See Figure 3.1). (S, δ) is a unit-disk graph metric space, where $\delta(p, q)$ is the length of the shortest path between p and q in $I(S)$.*

3.3.1 Construction of Well-Separated Pair Decomposition

Definition 3.3.2. *The density α of a point set S is defined to be the maximum number of points in S covered by any unit disk. S has bounded density if its density is $O(1)$.*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



(a) Metric space S and unit distance

(b) Unit-disk graph on S

Figure 3.1: Unit-disk graph

Given a point set S with bounded density, we start to compute a spanning tree of $I(S)$ in the following way: We first compute the relative neighborhood graph of S, which contains $O(n)$ edges. The relative neighborhood graph of a set of points S is the graph (S, E), where $(p, q) \in E$ iff there is no point $z \in S$ such that $\delta(p, z) < \delta(p, q)$ and $\delta(q, z) < \delta(p, q)$, where $\delta(\cdot, \cdot)$ is the distance function defined on $S \times S$. Let G be the graph that consists of all edges of length at most one in the relative neighborhood graph. It is known that G is a subgraph of $I(S)$, its maximum degree is 6, and G has the same connected components as $I(S)$ [40]. Then we compute a spanning tree T of G, which is also a spanning tree of $I(S)$ where the maximum degree of T is 6. This step takes $O(n \log n)$ time [40].

It is also known that any n-vertex tree with maximum degree β can be divided into two parts by removing a single edge so that each subtree contains at least n/β vertices.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Then recursively apply the balanced partitioning to obtain a balanced hierarchical decomposition of T. The decomposition can be represented as a rooted binary tree T' where each node $v \in T'$ corresponds to a (connected) subtree $T(v)$ of T. The root of T' corresponds to T, and for a node $v \in T'$, v's two children v_1, v_2 represent the two connected subtrees $T(v_1)$ and $T(v_2)$ obtained by removing an edge from $T(v)$. Denote by $S(v)$ the set of points in the subtree in $T(v)$. The height of the tree T' is $O(\log n)$.

Finally, run the following algorithm to compute a WSPD.

Algorithm *WSPD Algorithm (Unit-Disk Graph Metric)*
Input: A balanced hierarchical decomposition T' of T, and the root r of T'
Output: c-WSPD P

1. **for** each node $v \in T$
2. **do** pick an arbitrary point from $S(v)$ as a representative of $S(v)$ and denote it by $R(v)$
3. place $(S(r), S(r))$ in a queue Q
4. **while** $Q \neq \emptyset$
5. **do** remove the first element $(S(v_1), S(v_2))$ from Q
6. **if** $\delta(R(v_1), R(v_2)) \geq (c+2) \cdot \max(|S(v_1)| - 1, |S(v_2)| - 1)$
7. **then** add $(S(v_1), S(v_2))$ into P
8. **else if** $|S(v_1)| = |S(v_2)| = 1$
9. **then** discard the pair
10. **else if** $|S(v_1)| \geq |S(v_2)|$
11. **then** add to Q two pairs $(S(u_1), S(v_2))$ and $(S(u_2), S(v_2))$ where u_1, u_2 are two children of v_1
12. **else** add to Q two pairs $(S(v_1), S(u_1))$ and $(S(v_1), S(u_2))$ where u_1, u_2 are two children of v_2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

13. **return** P.

3.3.2 Analysis of the Number of Pairs

Gao and Zhang first prove the following lemma to show P is a c-WSPD of S:

Lemma 3.3.3. *P is a c-WSPD of S. Furthermore, each ordered pair of distinct points (p, q) is covered by exactly one pair in P.*

The following lemma shows that the sizes of two sets in the same pair do not differ too much.

Lemma 3.3.4. *Each pair (A, B) that ever appears in the queue satisfies $1/\beta \leq |A|/|B| \leq \beta$.*

The following lemma bounds the size of P.

Lemma 3.3.5. *If $(A, B_i) \in P, i = 1, \cdots, m$, then $B_i \bigcap B_j = \emptyset$, and $m = O(c^2|A|)$.*

Finally they present the bound on the number of pairs in the c-Well-Separated Pair Decomposition:

Lemma 3.3.6. $|P| = O(c^2 n \log n)$.

One of the main results obtained by them is as follows:

Theorem 3.3.7. *For any n points with constant-bounded density in the plane and any $c \geq 1$, there exists a c-WSPD with $(c^2 n \log n)$ pairs, which can be computed in $O(c^2 n \log n)$ time.*

Using clustering techniques, Gao and Zhang can obtain similar result for the unit-disk graph of point sets with unbounded density as follows:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Theorem 3.3.8. *For any set S of n points in the plane and any $c \geq 1$, there exists a c-WSPD of S under the unit-disk graph metric where P contains $(c^4 n \log n)$ pairs and can be computed in $O(c^4 n \log n)$ time.*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 4

Constructing WSPD for Doubling Metric Spaces

In this chapter, we will introduce an intuitive approach for constructing a Well-Separated Pair Decomposition for doubling metric spaces. Our algorithm is based on the approaches introduced in Sections 3.2 and 3.3 and extended in [42]. Contrary to [42], however, our algorithm is deterministic. The key of our algorithm is to apply r-net (see Section 2.1) and Proposition 1. This algorithm can be divided into two phases. In the first phase, we build a series of ball coverings and a split tree to give the doubling metric space a hierarchical decomposition. In the second phase, we run a pair selection procedure on the split tree to produce a c-WSPD of the original doubling metric space.

Throughout this chapter, (S, δ) denotes a doubling metric space, where S is a set of n elements, and $\delta(\cdot, \cdot)$ is the distance function defined on $S \times S$. We denote the doubling parameter by λ and the aspect ratio by Δ. Without loss of generality, suppose that $\min\{\delta(x, y) | x, y \in S, x \neq y\} = 1$. Thus, Δ is also the diameter of S.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.1 Building a Split Tree

Building a split tree T is the first step in the construction of a c-Well-Separated Pair Decomposition for the doubling metric space. Our algorithm computes a series of ball coverings C_i, $1 \leq i \leq 1 + \lfloor \log \Delta \rfloor$ of the entire space (S, δ) using algorithm BallCovering of Section 2.3. Each C_i is a covering of S by balls of radius $\frac{\Delta}{2^i}$.

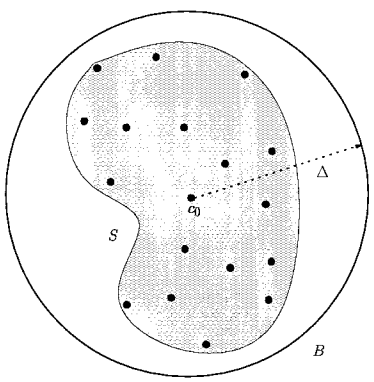
The details are described in the following algorithm. (See Figure 4.1 for an illustration)

Algorithm *BuildSplitTree*(S, Δ)
Input: a doubling metric space with diameter Δ and minimum distance 1.
Output: a pointer to the root of the split tree for S

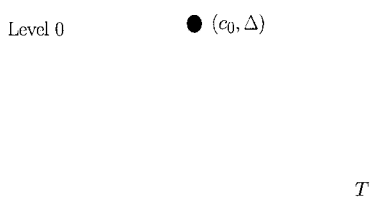
1. create a root z, let r_z be an arbitrary element of S, let $\Delta_z = \Delta$.
2. store r_z and Δ_z with z.
3. **for** $k = 1$ **to** $1 + \lfloor \log \Delta \rfloor$
4. **do** run algorithm BallCovering$(S, \frac{\Delta}{2^k})$ and let B_i, $1 \leq i \leq \lambda^{k+1}$, be the cover of S with balls of radius $\frac{\Delta}{2^k}$.
5. **for** $i = 1$ **to** λ^{k+1}
6. **do** let c_i be the center of B_i, create a node u, let $r_u = c_i$ and $\Delta_u = \frac{\Delta}{2^k}$, store r_u and Δ_u with u. let v be a node at level $k-1$ such that $\delta(r_u, r_v) \leq \Delta_v$, make u a child of v
7. **return** the root z

Since totally there are $1 + \lfloor \log \Delta \rfloor$ levels in T, and at each level, we run the ball covering algorithm of Chapter 2, the total running time for building a split tree is $O(\log \Delta * T)$, where T is the running time for constructing a ball covering of Chapter 2.

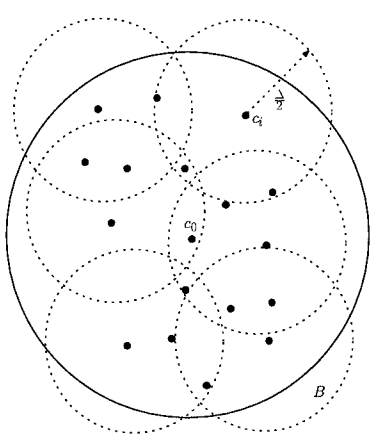
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



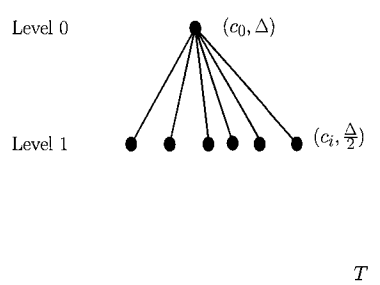
(a) S contained in $B(c_0, \Delta)$



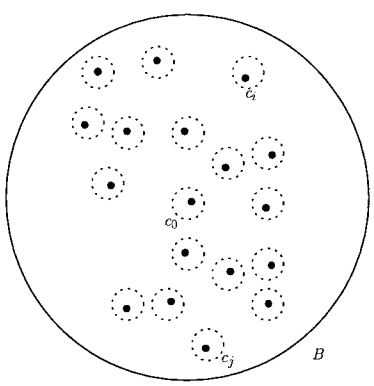
(b) Corresponding split tree



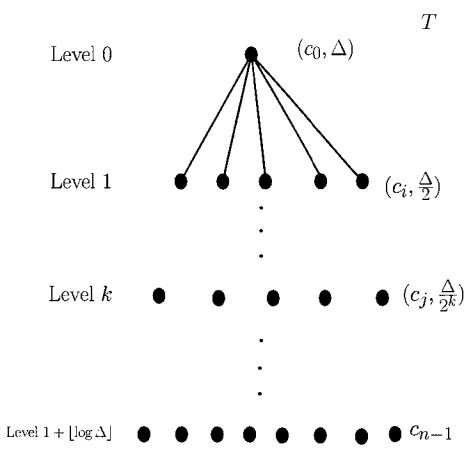
(c) Decomposition at level 1



(d) Corresponding split tree



(e) Decomposition at level $1 + \lfloor \log \Delta \rfloor$)



(f) Corresponding split tree

Figure 4.1: Building split tree

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.2 Computing a c-Well-Separated Pair Decomposition

In this part we will discuss an algorithm for selecting c-Well-Separated pairs. The idea is that we traverse pairs of nodes of the split tree T and check if they satisfy the condition of being c-Well-Separated. If two nodes in a pair satisfy this condition, we will include this pair into the c-Well-Separated Pair Decomposition. Otherwise we go to their children to check the condition again.

Algorithm *ComputeWSPD*(T, c)
Input: T is the split tree for S, $c > 1$
Output: A c-WSPD P for S

1. $P = \emptyset$;
2. Q = empty queue;
3. u = root of T;
4. add (u, u) to Q;
5. **while** $Q \neq \emptyset$
6. **do** (u, v) = first element in Q;
7. **if** $\delta(r_u, r_v) \geq 4(c+1) \cdot \max(\Delta_u, \Delta_v)$
8. **then** add (u, v) to P
9. **else** **if** $\Delta_u < \Delta_v$
10. **then** swap u and v
11. **for** each child w of u
12. **do** add (w, v) to Q
13. **return** P

The above algorithm is similar to the algorithm in [16] which is, however, randomized. c-WSPD is the output of the above algorithm.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

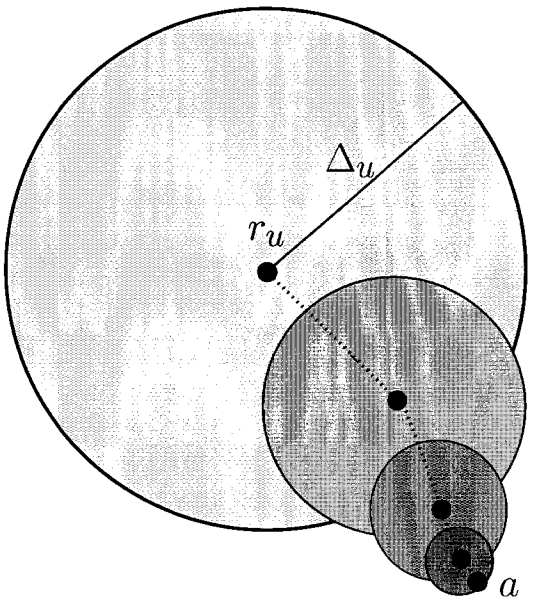


Figure 4.2: Illustration of the path between a and u in T of Lemma 4.2.1.

For any node u of T, let S_u be the set of all elements of S that are stored in the subtree of u in T. We now make the following claims and prove P is a c-WSPD.

Lemma 4.2.1. *Let P be the output of algorithm ComputeWSPD(T, c). Then (S_u, S_v), where $(u, v) \in P$, is a c-WSPD of S.*

Proof. From the algorithm, we know that a pair (u, v) is included in P only if $\delta(r_u, r_v) \geq 4(c+1)\cdot\max(\Delta_u, \Delta_v)$. Let $a \in S_u$ and $b \in S_v$, such that $\delta(a, b) = \delta(S_u, S_v)$. Consider the path $a = u_1, u_2, \cdots, u_k = u$ in T between a and u. Then

$$\begin{aligned}\delta(r_u, a) &\leq \textstyle\sum_{i=1}^{k-1} \delta(r_{u_i}, r_{u_{i+1}}) \\ &\leq \textstyle\sum_{i=1}^{k-1} \Delta_{u_{i+1}} \\ &\leq \textstyle\sum_{i=1}^{k-1} \frac{\Delta_u}{2^{k-i-1}} \\ &\leq 2\Delta_u\end{aligned}$$

Then we have:

$$\delta(r_u, r_v) \leq 2\delta(r_u, a) + \delta(a, b) + 2\delta(b, r_v)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$\leq 2\Delta_u + \delta(S_u, S_v) + 2\Delta_v$$
$$\leq \delta(S_u, S_v) + 4 \cdot \max(\Delta_u, \Delta_v).$$

It follows that

$$\delta(S_u, S_v) \geq \delta(r_u, r_v) - 4 \cdot \max(\Delta_u, \Delta_v)$$
$$\geq 4c \cdot \max(\Delta_u, \Delta_v).$$

Since S_u and S_v are contained in balls with r_u and r_v as centers and $2\Delta_u$ and $2\Delta_v$ as radii, respectively, we know that $D(S_u) \leq 4\Delta_u$ and $D(S_v) \leq 4\Delta_v$. Thus $\delta(S_u, S_v) \geq c \cdot \max(D(S_u), D(S_v))$. That is, every pair in (S_u, S_v), where $(u, v) \in P$ is a c-Well-Separated pair.

Next we prove that S_u and S_v are disjoint. We prove this by contradiction. Suppose $S_u \bigcap S_v \neq \emptyset$. From the construction of split tree, we know that $r_u \in S_v$ or $r_v \in S_u$. Without loss of generality, assume that $r_u \in S_v$. Then we have $\delta(r_u, r_v) \leq 2\Delta_v < 4(c+1) \cdot \max(\Delta_u, \Delta_v)$, contradicting the fact that the pair (S_u, S_v) is c-Well-Separated. Thus S_u and S_v are disjoint.

Since the node sets produced by the algorithm are disjoint, each ordered pair of distinct points (a, b) is covered by exactly one pair (S_u, S_v), where $(u, v) \in P$. That means (S_u, S_v) is a pair decomposition of (S, S) and also a c-Well-Separated Pair Decomposition of S. □

4.3 Analysis of the number of pairs in the WSPD

Lemma 4.3.1. *If (u, v) is ever included in the queue Q, then $\frac{1}{2} \leq \frac{\Delta_u}{\Delta_v} \leq 2$.*

Proof. We prove this by induction. Clearly, it is true for the pair (u, u), where u is the root of T. Now, without loss of generality, suppose a pair (u, v) comes from splitting

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$(p(u), v)$, where $p(u)$ is parent of u in T. Thus we have $\Delta_{p(u)} \geq \Delta_v$ and $1 \leq \frac{\Delta_{p(u)}}{\Delta_v} \leq 2$. Since $\Delta_u = \frac{1}{2}\Delta_{p(u)}$, we know that $\frac{\Delta_u}{\Delta_v} \leq \frac{\Delta_{p(u)}}{\Delta_v} \leq 2$ and $\frac{\Delta_u}{\Delta_v} \geq \frac{1}{2} \cdot \frac{\Delta_{p(u)}}{\Delta_v} \geq \frac{1}{2} \cdot 1 = \frac{1}{2}$. Hence $\frac{1}{2} \leq \frac{\Delta_u}{\Delta_v} \leq 2$. □

Lemma 4.3.2. *Let $(u, v) \in P$. Then $\delta(r_u, r_v) \leq (8c + 10) \cdot \max(\Delta_u, \Delta_v)$.*

Proof. Without loss of generality, suppose (u, v) comes from splitting $(p(u), v)$, where $p(u)$ is parent of u in T. By our c-Well-Separated pairs selecting procedure, we know that $\delta(r_{p(u)}, r_v) < 4(c + 1) \cdot \max(\Delta_{p(u)}, \Delta_v) \leq 8(c + 1) \cdot \max(\Delta_u, \Delta_v)$, because $\Delta_{p(u)} = 2\Delta_u$.

Since $\delta(r_{p(u)}, r_u) \leq \Delta_{p(u)} = 2 \cdot \Delta_u$, we have:

$$\begin{aligned}\delta(r_u, r_v) &\leq \delta(r_u, r_{p(u)}) + \delta(r_{p(u)}, r_v) \\ &\leq 2 \cdot \Delta_u + 8(c + 1) \cdot \max(\Delta_u, \Delta_v) \\ &\leq (8c + 10) \cdot \max(\Delta_u, \Delta_v).\end{aligned}$$

□

Theorem 4.3.3. *For any doubling metric space (S, δ) and for any $c > 1$, there exists a c-Well-Separated Pair Decomposition consisting of $O(\lambda^{O(\log c)} \cdot \log \Delta \cdot n)$ pairs, where $n = |S|$, λ is the doubling parameter and Δ is the aspect ratio of (S, δ).*

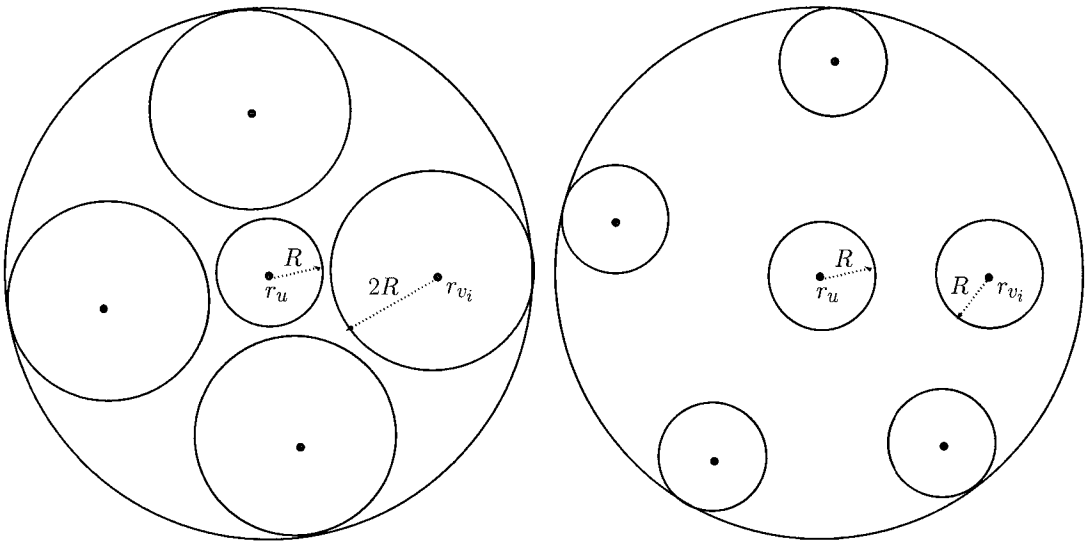
Proof. Let u be a node of the split tree T, and consider all pairs (u, v_i), $1 \leq i \leq k$, in P, such that each v_i is at the same or at a lower level than u.

We know from Lemma 4.3.1 that each node v_i is either at the same level in T as u or at the same level as $p(u)$. Let $R = \Delta_u$.

Case 1: v_i is at the same level as $p(u)$. Then $\Delta_{v_i} = 2R$. By Lemma 4.3.2:

$$\begin{aligned}\delta(r_u, r_{v_i}) &\leq (8c + 10) \cdot \max(\Delta_u, \Delta_{v_i}) \\ &\leq (8c + 10) \cdot \max(R, 2R) \\ &\leq (16c + 20)R.\end{aligned}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



(a) Case 1: v_i at same level as $p(u)$ (b) Case 2: v_i at same level as u

Figure 4.3: Illustration of two cases when analyzing number of pairs

Let $Y = \{v_i |\ v_i$ is at the same level as $p(u)\}$. The diameter of Y is at most $(32c+40)R$, and the minimum distance in Y is at least $2R$. Thus $\Delta(Y) \leq 16c+20$ and, by Proposition 1 of Chapter 2, $|Y| \leq \lambda^{2+\lfloor \log(16c+20) \rfloor} = \lambda^{O(\log c)}$.

Case 2: v_i is at the same level as u. Then $\Delta_{v_i} = R$. By Lemma 4.3.2:

$$\begin{aligned} \delta(r_u, r_{v_i}) &\leq (8c+10) \cdot \max(\Delta_u, \Delta_{v_i}) \\ &\leq (8c+10) \cdot \max(R, R) \\ &\leq (8c+10)R. \end{aligned}$$

Let $Y = \{v_i |\ v_i$ is at the same level as $u\}$. The diameter of Y is at most $(16c+20)R$, and the minimum distance in Y is at least R. Thus $\Delta(Y) \leq 16c+20$ and, by Proposition 1 of Chapter 2, $|Y| \leq \lambda^{2+\lfloor \log(16c+20) \rfloor} = \lambda^{O(\log c)}$.

Combine all the above results, we have shown that k is bounded by $\lambda^{O(\log c)}$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Since there are at most $O(n \cdot \log \Delta)$ nodes u in T, the number of pairs in the c-Well-Separated Pair Decomposition is bounded by $O(\lambda^{O(\log c)} \cdot \log \Delta \cdot n)$.

□

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis we have considered the problem of constructing Well-Separated Pair Decompositions (WSPD) for doubling metric spaces. Given a doubling metric space (S, δ), where S is a set of elements, $|S| = n$ and $\delta(\cdot, \cdot)$ is the distance function defined on $S \times S$. λ is the doubling parameter and Δ is the aspect ratio of S. We have presented a deterministic algorithm that constructs a c-Well-Separated Pair Decomposition. The algorithm is quite intuitive and easy to manipulate. The number of pairs in this c-WSPD is $O(\lambda^{O(\log c)} \cdot \log \Delta \cdot n)$ and the running time of this algorithm is $O((n + T) \log \Delta)$, where T is the running time for constructing a ball covering of S.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.2 Future Work

The main open problem is whether there exists a deterministic algorithm, which can be used to construct a c-WSPD with $O(n)$ pairs in $O(n \log n)$ time. Namely, can we remove the dependency on the aspect ratio Δ so that the result matches the conclusion of Callahan and Kosaraju [7] for Euclidean space.

In addition, because the doubling parameter λ in high dimensional metric spaces is much bigger than that in low dimensional doubling metric space, and the number of pairs of resulting c-WSPD is highly depended on λ, we should consider whether these algorithms are quite fitted for high dimensional doubling metric space.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Bibliography

[1] Sunil Arya, Gautam Das, David M. Mount, Jeffrey S. Salowe, and Michiel Smid. Euclidean spanners: short, thin, and lanky. In *STOC '95: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 489–498, 1995.

[2] Sunil Arya, David M. Mount, and Michiel H. M. Smid. Randomized and deterministic algorithms for geometric spanners of small diameter. In *35th Annual Symposium on Foundations of Computer Science*, pages 703–712. IEEE, 1994.

[3] P. Assouad. Plongements lipschitziens dans $\mathcal{R}^n$. *Bull. Soc. Math. France*, 111(4):429–448, 1983.

[4] Paul B. Callahan. Optimal parallel all-nearest-neighbors using the well-separated pair decomposition (preliminary version). In *34th Annual Symposium on Foundations of Computer Science*, pages 332–340. IEEE, 1993.

[5] Paul B. Callahan and S. Rao Kosaraju. Faster algorithms for some geometric graph problems in higher dimensions. In *SODA '93: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 291–300, 1993.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[6] Paul B. Callahan and S. Rao Kosaraju. Algorithms for dynamic closest pair and n-body potential fields. In *SODA '95: Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 263–272, 1995.

[7] Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. ACM*, 42(1):67–90, 1995.

[8] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Math.*, 86(1-3):165–177, 1990.

[9] Kenneth L. Clarkson. Nearest neighbor queries in metric spaces. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 609–617. ACM Press, 1997.

[10] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein. *Introduction to Algorithms.* MIT Press and McGraw-Hill, second edition, 2001.

[11] Mark de Berg, Marc van Kreveld, Mark Overmas, and Otfried Schwarzkopf. *Computational Geometry, Algorithms and Application.* Springer-Verlag, second edition, 2000.

[12] Christian A. Duncan, Michael T. Goodrich, and Stephen Kobourov. Balanced aspect ratio trees: combining the advantages of k-d trees and octrees. In *SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 300–309. Society for Industrial and Applied Mathematics, 1999.

[13] Jeff Erickson. Dense point sets have sparse delaunay triangulations. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 125–134, 2002.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[14] Tomás Feder and Daniel Greene. Optimal algorithms for approximate clustering. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 434–444. ACM Press, 1988.

[15] Jie Gao, Leonidas J. Guibas, John Hershberger, Li Zhang, and An Zhu. Geometric spanner for routing in mobile networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 45–55. ACM Press, 2001.

[16] Jie Gao and Li Zhang. Well-separated pair decomposition for the unit-disk graph metric and its applications. In *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 483–492, 2003.

[17] Jie Gao and Li Zhang. Tradeoffs between stretch factor and load balancing ratio in routing on growth restricted graphs. In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 189–196. ACM Press, 2004.

[18] T. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoret. Comput. Sci.*, 38:293–306, 1985.

[19] Satish Govindarajan, Tamás Lukovszki, Anil Maheshwari, and Norbert Zeh. I/o-efficient well-separated pair decomposition and its applications. In *ESA '00: Proceedings of the 8th Annual European Symposium on Algorithms*, pages 220–231, 2000.

[20] Joachim Gudmundsson, Christos Levcopoulos, Giri Narasimhan, and Michiel Smid. Approximate distance oracles for geometric graphs. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 828–837, 2002.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[21] Leonidas Guibas, An Nguyen, Daniel Russel, and Li Zhang. Collision detection for deforming necklaces. In *SCG '02: Proceedings of the eighteenth annual symposium on Computational geometry*, pages 33–42. ACM Press, 2002.

[22] Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *44th Annual Symposium on Foundations of Computer Science*, page 534. IEEE, 2003.

[23] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300. ACM Press, 2004.

[24] Sariel Har-Peled and Manor Mendel. Fast construction of nets in low dimensional metrics, and their applications. *arXiv:cs.DS/0409057 (http://arxiv.org/abs/cs.DS/0409057)*, 2004.

[25] J. Heinonen. *Lectures on Analysis on Metric Spaces.* Springer-Verlag, 2001.

[26] Kirsten Hildrum, John Kubiatowicz, Sean Ma, and Satish Rao. A note on the nearest neighbor in growth-restricted metrics. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 560–561. Society for Industrial and Applied Mathematics, 2004.

[27] Piotr Indyk. Algorithms for dynamic geometric problems over data streams. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 373–380. ACM Press, 2004.

[28] V. de Silva J.B. Tenenbaum and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[29] David R. Karger and Matthias Ruhl. Finding nearest neighbors in growth-restricted metrics. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 741–750. ACM Press, 2002.

[30] Philip Klein, Satish Rao, Monika Rauch, and Sairam Subramanian. Faster shortest-path algorithms for planar graphs. In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 27–37. ACM Press, 1994.

[31] Robert Krauthgamer and James R. Lee. The black-box complexity of nearest neighbor search. In *31st Internat. Colloq. Automata Lang*, pages 858–869, 2004.

[32] Robert Krauthgamer and James R. Lee. Navigating nets: simple algorithms for proximity search. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 798–807. Society for Industrial and Applied Mathematics, 2004.

[33] T. J. Laakso. Plane with a_∞ weighted metric not bi-lipschitz embeddable to $\mathcal{R}^n$. *Bull. London Math. Soc.*, 34(6):667–676, 2002.

[34] U. Lang and C. Plaut. Bilipschitz embeddings of metric spaces into space forms. *Geom. Dedicata*, 87(1-3):285–307, 2001.

[35] Christos Levcopoulos, Giri Narasimhan, and Michiel H. M. Smid. Improved algorithms for constructing fault-tolerant spanners. *Algorithmica*, 32(1):144–156, 2002.

[36] Nathan Linial. Finite metric spaces: combinatorics, geometry and algorithms. In *SCG '02: Proceedings of the eighteenth annual symposium on Computational geometry*, pages 63–63. ACM Press, 2002.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[37] Giri Narasimhan and Michiel Smid. Approximating the stretch factor of euclidean graphs. *SIAM J. Comput.*, 30(3):978–989, 2000.

[38] T. S. E. Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. In *In 21st Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM-02)*, pages 170–179, 2002.

[39] M. Mendel R. Krauthgamer, J. R. Lee and A. Naor. Measured descent:a new embedding method for finite metric spaces. In *45th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 798–807, 2004.

[40] Kenneth J. Supowit. The relative neighborhood graph, with an application to minimum spanning trees. *J. ACM*, 30(3):428–448, 1983.

[41] Bruce M. Maggs T-H. Hubert Chan, Anupam Gupta and Shuheng Zhou. On hierarchical routing in doubling metrics. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, 2005.

[42] Kunal Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 281–290. ACM Press, 2004.

[43] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. In *FOCS '01: Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, page 242. IEEE Computer Society, 2001.

[44] Mikkel Thorup and Uri Zwick. Approximate distance oracles. In *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 183–192. ACM Press, 2001.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.