Combinatorial Algorithms for Nearest Neighbors, Near-Duplicates and Small-World Design

Yury Lifshits[*] Shengyu Zhang[†]

Abstract
We study the so called combinatorial framework for algorithmic problems in similarity spaces. Namely, the input dataset is represented by a comparison oracle that given three points x, y, y' answers whether y or y' is closer to x. We assume that the similarity order of the dataset satisfies the four variations of the following *disorder inequality*: if x is the a'th most similar object to y and y is the b'th most similar object to z, then x is among the $D(a+b)$ most similar objects to z, where D is a relatively small disorder constant.

Though the oracle gives much less information compared to the standard general metric space model where distance values are given, one can still design very efficient algorithms for various fundamental computational tasks. For nearest neighbor search we present deterministic and exact algorithm with almost linear time and space complexity of preprocessing, and near-logarithmic time complexity of search. Then, for near-duplicate detection we present the first known deterministic algorithm that requires just near-linear time + time proportional to the size of output. Finally, we show that for any dataset satisfying the disorder inequality a *visibility graph* can be constructed: all outdegrees are near-logarithmic and greedy routing deterministically converges to the nearest neighbor of a target in logarithmic number of steps. The later result is the first known work-around for Navarro's impossibility of generalizing Delaunay graphs.

The technical contribution of the paper consists of handling "false positives" in data structures and an algorithmic technique *up-aside-down-filter*.

1 Introduction

The Combinatorial Framework. The statement of many algorithmic problems is starting with "Given n objects in metric space...": nearest neighbor search, clustering, closest pairs, near-duplicate detection. To formalize these problems we have to (1) define the data representation and (2) make some assumptions about the dataset. So far many algorithms were designed for specific data model, such as Hamming cube [26], or for so called *general metric spaces* [34] with assumptions that some intrinsic dimension [13, 21, 25] is small.

[*]Yahoo! Research. The work was done when the author was at California Institute of Technology, supported by Lee Center for Advanced Networking and Center for the Mathematics of Information.

[†]The Chinese University of Hong Kong. Most work was done when the author was at California Institute of Technology, supported by NSF Grant PHY-0456720 and ARO Grant W911NF-05-1-0294.

However, there is a large class of instances that requires principally new approach. We say a dataset has the *separation effect* if the ratio between the largest and the smallest pairwise distances is below 2. Take for example the similarity measure "number of joint friends" for members of a social network. In a typical case there are at most 20 out 200 friends in common. This give us 0.9 in the so called Jaccard distance for two really similar people and 1.0 for completely unrelated. The similar story is observed with texts (similarity by words), webpages (similarity by referring sites), movies (similarity by reviewers and ratings) and so on. In all this cases all objects appeared to be "unique". Unfortunately, datasets with separation effect always have the doubling dimension close to the worst possible. Also the branch-and-bound techniques have to visit all objects, because the metric triangle inequality produces no new knowledge on yet uncomputed distances.

Addressing the challenge of separation effect, in 2008 Goyal, Lifshits and Schütze proposed the *combinatorial framework* [17]. Instead of the availability of the distance between any two points in U, suppose we are only given a *comparison oracle* O which can tells whether y is closer to x or y' is closer to x. In other words, it discards all the metric information between points, with only the relative closeness comparisons remained. We define $\text{rank}_x(y)$ to be an integer position of y in the list of all object in the dataset sorted by closeness to x. The central assumption of this paper is that $R(x, y) = \max(\text{rank}_x(y), \text{rank}_y(x))$ is a quasimetric with some relatively small *disorder constant* D. More formally, we assume that the following *disorder inequality* holds: $R(x, z) \leq D(R(x, y) + R(y, z))$. This assumption is based on experimental observations on Reuters corpus of news articles [17].

We think that the combinatorial framework is worth studying for a number of reasons:

New solvable classes. Problems like nearest neighbors are intractable in general. Thus, researchers are looking for additional assumptions and subclasses of the input data that make the problem easier. And as we show in this paper, the combinatorial framework provides such a new subclass.

This subclass turns out to be another relaxation of low-expansion rate after doubling dimension framework, and this subclass and that for constant doubling dimension are incomparable. In particular, we can solve nearest neighbors even for some datasets not satisfying the metric triangle inequality.

Addressing heterogeneous data models. In many modern applications object representations that are far from simple abstractions like Euclidean space. For instance, consider a description of a blog: language, geographic location (dictionary parameters), age, number of posts (numerical parameters), referring links and reader list (graph parameters), text of profile and posts (text parameters) and posting timestamps (time parameters). Such a heterogeneous description needs a quite complicated similarity function. E.g. it can include manually defined logical rules and threshold functions. The combinatorial framework can work with arbitrarily complicated similarity functions without any customization. This is an important advantage comparing to well-established locality-sensitive hashing approach [20] that requires designing new hash functions for every data model.

Using only relative order. In many applications, designing a similarity (distance) function is challenge by itself. With the combinatorial framework this task becomes easier. Say, when similarity itself is a subject to learn [3], we need only comparative training information.

Non-reducibility to studied models. The combinatorial framework draw some inspiration from previously studied models, most notably doubling dimension [18, 24, 25]. Despite some known results on repairing quasimetrics [28], we do not see a way to reduce this new model to the old ones. If you try to define a metric for some similarity order satisfying the disorder inequality you can achieve the metric triangle inequality *or* a small doubling dimension, but not both. Also, in the combinatorial framework, the full similarity is not given as a part of the input. Thus, evaluating any distance based on similarity ranking can be computationally very expensive.

Results. We present three new algorithms in the paper.

Deterministic and exact algorithm for nearest neighbor search. Nearest-Neighbor Search (NNS) is the problem of preprocessing a set S of n objects lying in some space U with a similarity function σ so that given a query $q \in U$, one can efficiently locate a point in S that is most similar to q (among the points in S). We present a deterministic preprocessing algorithm of $poly(D)n\log^2 n$ time complexity and a searching algorithm of $poly(D)\log n$ time complexity for the NNS problem. Note that the subquadratic time complexity requires a computation even less than obtaining the comparison information between all pairs.

Deterministic discovery of all near-duplicates. Detecting and eliminating replicated documents is recognized as one of the central problems for search engines [6, 4, 10, 23]. Notice that this is a typical situation of the separation effect: Almost all distances in the range $[\frac{1}{2}, 1]$, since 50% similarity is considered to be threshold for duplicates and thus all "original" documents have over one half distances between each other. Thus the metric triangle inequality is non-informative and doubling constant/expansion rate are close to the maximal possible. In this paper we show that the combinatorial framework provides efficient solution to near-duplicates. Assume our similarity oracle can also answer queries "Whether similarity between x and y is above the duplicate threshold or not?" Then we can find *deterministically* all pairs of near-duplicates in time $poly(D)(n\log^2 n + |Output|)$.

Small world design. Starting from the seminal paper of Kleinberg [22], navigating schemes for various small world models were intensively studied. Designing peer-to-peer network protocols such as Meridian [33] raises the following Small World Design problem: given n nodes in some metric space, construct a small number of out-going connections for every node such that from any given starting point greedy routing leads to the nearest neighbor of the target point. For any dataset with disorder constant D, we construct its *visibility graph*, with $\mathcal{O}(D^4\log n)$ out-degrees in $poly(D)n\log^2 n$ time. The greedy-style local search in visibility graph finds the exact nearest neighbor of the target destination by at most $\log n$ moves on the graph edges. And as a corollary, visibility graph creates $\mathcal{O}(D^4\log n)$-to-check certificates of being the exact nearest neighbor. Such certificates are not known for previously studied models.

The Voronoi diagram and the closely related Delaunay graph have the following "local-implies-global" property: If some point in the dataset is closer to query q than the center of any adjacent Voronoi cell (i.e. any Delaunay neighbor), then this point is the nearest neighbor of q. Thus, greedy walk through Delaunay graph will eventually lead to the nearest neighbor. In 1999 Navarro [29] made an attempt to find a structure similar to Delaunay graph for a general metric space. Unfortunately, he came up with the following *negative* result:

Given the pairwise distances for a finite subset S

of an unknown metric space U, for each $a, b \in S$ there exists a choice for U where a and b are connected in the Delaunay graph of S.

Our result on visibility graph avoids a similar negative barrier by using the concept of *visible neighbors* instead of Delaunay neighbors.

Relations between the disorder inequality and other concepts of intrinsic dimension, namely the doubling dimension and the expansion rate, are also discussed. We show that disorder inequality implies a combinatorial form of "doubling effect". At the same time, if a dataset has expansion rate c (definition by Karger and Ruhl [21]) then its disorder constant is at most c^2.

Technical contribution. Designing algorithms in the combinatorial framework is challenging for a number of reasons: (1) for a given pair, combinatorial ranking is not given and can not be evaluated in constant time, (2) we do not have numerical information about closeness which can be very informative, (3) the disorder inequality has a relaxation constant D and thus can not be used directly in a recursive manner.

While our basic data structures are similar to Krauthgamer-Lee [25] navigating nets and Chan-Dinitz-Gupta density nets [9], the crucial additional one, cousin table, is the original contribution of this paper. Basically, we accompany the "vertical" data structure (formed by a sequences of epsilon nets with shrinking radius) with a "horizontal" one, called *cousins*, to help to identify nearby members of the same layer. The precise definition is the following: two centers x and x' of combinatorial balls of radius r are called cousins if there are two members $y \in B(x, r)$ and $y' \in B(x', r)$ such that y and y' have rankat most r from each other. Cousins are different from "friends" in [14] by the following innovative property: they tolerate "false positives". E.g. checking whether a particular two objects are cousins is very expensive and thus we use "might-be-cousins" instead. On the algorithmic side, we contribute with the "up-aside-down-filter" technique. Having vertical and horizontal "maps" on some top levels we create a map of the next level of preciseness by looking for parents, for cousins of parents and finally for children-of-cousins-of-parents. Due to our definition of cousins this look up will always return the neighborhood we are constructing at a given step.

1.1 Comparisons with related work Similarity search in small intrinsic dimension. The combinatorial framework was introduced in [17], where the authors gave two randomized algorithms for NNS. We improve their results in two aspects (and therefore answering two open problems there). Namely, our algorithms are deterministic and use only subquadratic time for preprocessing.

Let us compare our results to studies of growth restricting metrics. Our algorithms are deterministic, as opposed to the randomized data structure construction in Hildrum et al. [19] (based on Karger and Ruhl [21] and Plaxton et al. [31]). Comparing to cover-trees [5] we have similar preprocessing and search time complexity. But our key advantage comparing to all these results is that the disorder inequality is an assumption strictly weaker than the bounded growth rate, as we will show in Section 2.

Speaking about doubling dimension, our algorithm finds the exact nearest neighbor, compared to the approximate ones by Krauthgamer and Lee [25, 24] and by Cole and Gottlieb [14]. Our near-linear construction of combinatorial nets is deterministic while the data structure constructions by Clarkson [12] and Har-Peled and Mendel [18] are randomized.

We still have to match two more results from [5] and [14]: linear space bound and handling insertions/delitions.

Near-duplicates. Compared to the results based on the minhashing [7], ours has deterministic subquadratic guarantee for running time and works for arbitrary similarity functions.

Small world design. Early work on this includes Arya and Mount's discussion of a greedy "routing" scheme for approximate nearest neighbor search in R^d [2]. Recent algorithms by Fraigniaud, Lebhar and Lotker [16] and Slivkins [32] provide efficient solutions assuming doubling dimension is $\mathcal{O}(\log \log n)$. Moreover, [16] provides counterexamples for larger doubling dimensions. Our solution has a couple of advantages. First, it has *deterministic guarantee* for convergence in logarithmic number of steps. Second, it has no dependence on the aspect ratio, i.e. ratio between maximal and minimal distances.

2 The Combinatorial Framework

The combinatorial framework is a model of computation for problems dealing with informal concept of "closeness". In literature both terms of similarity and distance are used: (1) when points are close to each other we say "distance is small" or "similarity is high", (2) for distance functions, the triangle inequality is usually assumed. The combinatorial framework is focused on the notion of similarity by not assuming the triangle inequality for distance values.

Instead of providing exact values of similarity function σ we give an access to the *comparison oracle* which, for each query (x, y, y'), tells whether $\sigma(x, y) < \sigma(x, y')$ or $\sigma(x, y) > \sigma(x, y')$. For simplicity, throughout this paper we assume that no tie exists.

Consider some dataset S of n points. For each point x, all other $n-1$ points y can be sorted by $\sigma(x,y)$. We call all these n sorted lists together to be a *similarity order* for the dataset S. Indeed, this is the only information we can get from the comparison oracle. For any points $x, y \in S$, define $\mathrm{rank}_{x,S}(y)$ to be the position of y when the elements of S are sorted according to similarity to x in the deceasing order. We omit S when it is clear from the context. For convenience let $\mathrm{rank}_x(x) = 0$. We define a *combinatorial ball* as $B(x,r) = \{y : \mathrm{rank}_x(y) < r\}$.

Once we waive the real values of similarities, we lost almost all the knowledge about the dataset. In order to make algorithmic problems tractable we introduce a consistency assumption for the similarity order. Namely, we assume that the rankfunction satisfies the following weak triangle inequalities:

$$\mathrm{rank}_x(y) \le D(\mathrm{rank}_x(z) + \mathrm{rank}_y(z)), \tag{2.1}$$
$$\mathrm{rank}_x(y) \le D(\mathrm{rank}_x(z) + \mathrm{rank}_z(y)), \tag{2.2}$$
$$\mathrm{rank}_x(y) \le D(\mathrm{rank}_z(x) + \mathrm{rank}_y(z)), \tag{2.3}$$
$$\mathrm{rank}_x(y) \le D(\mathrm{rank}_z(x) + \mathrm{rank}_z(y)). \tag{2.4}$$

The above inequalities are called the *disorder inequalities*, and the minimal constant D making them true is called the *disorder constant*. Some comments are in order. First, we list all four possible disorder inequalities simply because there seems not to be any strong reason that some are more reasonable than others. Second, by putting $z = y$ in the first inequality, we have that

$$\mathrm{rank}_x(y) \le D\mathrm{rank}_y(x). \tag{2.5}$$

Thus any one of the above four disorder inequalities implies the other three with a constant $D' = D^2$. As shown in [17], the disorder constant D captures the intuitive notion of intrinsic dimension. The following lemma illustrate this correspondence in the case of fixed dimensions:

Lemma 2.1. *For sufficiently large hypercube in the d-dimensional integer grid $\mathbb{Z}^d$ with Euclidean distance, its disorder constant is 2^{d-1} up to a multiplicative constant close to one.*

We also define the following notion to compare to the doubling dimension.

Definition 2.1. *(disorder dimension) Let (S, σ) be a set in similarity space and $D(S)$ be the disorder constant of S. Then we call $(1 + \log D)$ the* disorder dimension *of S.*

2.1 Disorder constant vs. doubling constant and expansion rate

Definition 2.2. *The metric ball $MB(p,r)$ is the set of all possible objects within distance r from p. The expansion rate of a set S in a metric space is the minimal number c* s.t. *$\forall p \in S$, $\forall r$, it holds that $|MB(p,2r) \cap S| \le c|MB(p,r) \cap S|$.*

Proposition 2.1. *(Small expansion rates imply small disorder constants) For any n-point set S with the expansion rate c in a metric space, all the four disorder inequalities are satisfied with $D = c^2$.*

Proof. Let us prove, say $\mathrm{rank}_x(y) \le c^2(\mathrm{rank}_z(y) + \mathrm{rank}_z(x))$. Assume that $d(x,z) \ge d(z,y)$, then $\mathrm{rank}_x(y) = |S \cap MB(x, d(x,y))| \le |S \cap MB(z, d(x,y) + d(z,x))|$ by the metric triangle inequality. Note that $d(x,y) + d(z,x) \le 2d(z,x) + d(z,y) \le 3d(z,x)$, we have $|S \cap MB(z, d(x,y) + d(z,x))| \le c^2|S \cap MB(z, d(z,x))| = c^2\mathrm{rank}_z(x)$ by the definition of expansion rate. The case of $d(x,z) \le d(z,y)$ and all the other three disorder inequalities are proved in a similar way.

Proposition 2.2. *(Doubling effect) Consider a set S of n points satisfying the disorder inequalities. Then for any point p and integer r the combinatorial ball $B(p,2r)$ can be covered by at most $O(D^2)$ combinatorial balls of radius r.*

Proof. Construct the covering in an greedy way: add yet uncovered members of $B(p,2r)$ as new centers one by one. Then, in any pair of centers the rankfrom the later-selected to the earlier-selected is at least r. Thus, if we look for "core" combinatorial balls of radius $r/2D$ around the same centers, they all are disjoint. On the other hand, by the disorder inequality the members of core balls having at most $2Dr$ rankto p. Thus, there can not be more than $4D^2$ centers in our greedy covering.

Indyk's examples. In email correspondence with the first author Indyk showed a dataset with small doubling constant but large disorder constant, and another dataset with small disorder constant but large doubling constant and expansion rate. We put these examples to Appendix A.

3 Nearest Neighbor Search in the Combinatorial Framework

Recall that the Nearest Neighbor Search problem is as follows. Given an n-point data set S and the comparison oracle O, preprocess the data *s.t.* given any other query point $q \notin S$, we can efficiently find the nearest neighbor x_{NN} of q, *i.e.* the point x_{NN} with $\mathrm{rank}_q(x_{NN}) = 1$. Let D be the disorder constant of $S \cup \{q\}$. Throughout the paper, we will use $\mathrm{rank}_x(y)$ to mean $\mathrm{rank}_{x,S\cup\{q\}}(y)$.

3.1 Data Structures We are going to use five data structures: a layered list of objects $\mathcal{L}$, a direct index $\mathcal{D}$, an inverted index $\mathcal{I}$, a set of navigation pointers $\mathcal{N}$ and a cousin table $\mathcal{C}$.

The layered list of objects $\mathcal{L}$ is an ordering of all objects in the dataset $p_1, \ldots, p_n$ together with integer thresholds $t_0 = 1 \leq t_1 \leq \ldots t_{k-1} \leq t_k = n$. These thresholds define layers in the list: layer $L_i = \{p_1, \ldots, p_{t_i}\}$. We require every layer i to satisfy the *combinatorial net* definition with radius $r_i = \frac{n}{2^i}$:

Coverage. $\forall 0 \leq i \leq k \;\; \forall p \in S \;\; \exists p' \in L_i$ s.t. $\mathrm{rank}_{p'}(p) < r_i$

Separation. $\forall 0 \leq i \leq k \quad \forall p, p' \in L_i : \quad \mathrm{rank}_p(p') \geq r_i$ OR $\mathrm{rank}_{p'}(p) \geq r_i$

The direct index $\mathcal{D}$ represents neighborhoods of objects in the dataset. The larger i, the level of appearance of an object, is, the smaller neighborhood we store. Formally for every level i and every $p \in L_i - L_{i-1}$ we keep the sorted (by similarity to p) list of $n/2^i$ most similar to p objects. We call these objects *children* of p.

The inverted index $\mathcal{I}$ contains the same information as the direct index but sorted by children rather than by parent. Formally, for every i and for every $p \in S$ we list all $p' \in L_i$ such that $\mathrm{rank}_{p'}(p) < \frac{n}{2^i}$.

The navigation pointers $\mathcal{N}$ contain the collections of pointers from every member of L_{i-1} to nearby members of L_i. Formally, there is a navigation link from $p \in L_{i-1}$ to $p' \in L_i$ if $\mathrm{rank}_p(p') \leq 3D^2 \frac{n}{2^i}$. Actually, due to technical reasons we allow "false positives" in navigation pointers: all pointers that qualify should be listed, but some non-qualifying links can be also included.

Cousins table $\mathcal{C}$ is the crucial auxiliary data structure in our work. For every member of every layer i it contains links to nearby members of the same layer. Formal rule is the following: $p, p' \in L_i$ considered to be cousins if they have children $s \in B(p, \frac{n}{2^i}), s' \in B(p', \frac{n}{2^i})$ such that $\mathrm{rank}_s(s') < \frac{n}{2^{i+1}}$ OR $\mathrm{rank}_{s'}(s) < \frac{n}{2^{i+1}}$. Again, we allow false positives in this data structure: some non-cousin links can be included.

3.2 Algorithms

Preprocessing Algorithm. Given the comparison oracle and the disorder constant D we construct our five data structures. Here is the overall structure of the algorithm, followed by more detailed explanations:

```
1. Take an arbitrary $p_1$, set $L_0 = \{p_1\}$.
   Sort all objects in $S$ by their similarity to $p_1$.
   Initialize $\mathcal{D}, \mathcal{I}, \mathcal{C}$ with information corresponding to layer 0.
2. For every $1 \leq i \leq \lceil \log_2 n \rceil$ do:
     Update $\mathcal{I}$, compute the list of yet uncovered objects $U$
     Repeat until $U$ is empty:
       Take some point $p \in U$
       Compute the ball $B(p, \frac{n}{2^i})$
       Update $\mathcal{D}$, $\mathcal{I}$ and $U$
     Compute navigation pointers from $L_{i-1}$ to $L_i$
     Compute cousin pointers for $L_i$
```

Computing U: The first $n/2^i$ objects in $\mathcal{D}$ column corresponding to some $p \in L_{i-1}$ are its layer-i children. Thus, we can mark p as their layer-i parent in our inverted index $\mathcal{I}$. Then we scan $\mathcal{I}$ and create the list U of all objects that does not have any layer-i parent yet.

Computing a new ball: Here we use our core technique: *up-aside-down-filter*. Take some $p \in U$. Use $\mathcal{I}$ to identify some layer-$(i-1)$ parent of p. Use $\mathcal{C}$ to take all cousins of this parent. Use $\mathcal{D}$ to take all children of all cousins of identified parent of p. Now sort them by similarity to p and keep just $n/2^i$ most similar. These are the actual members of $B(p, n/2^i)$. We prove it in the next subsection. The sorted list of members of the new ball goes to direct index, all membership facts go to inverted index and now-covered objects are removed from U.

Computing navigation pointers from layer $i-1$ to layer i: Take some $p \in L_{i-1}$. Look up some of its parents on the layer $i - \log(6D^2)$. Take all cousins of this parent. Take all layer-i children of these cousins. Sort them by similarity to p and keep the $12D^4$ most similar ones. Again, we use up-aside-down-filter technique.

Computing cousin links on layer i: For every $p \in L_i$ do the following. Take an arbitrary parent on the layer $i - \log(6D^2)$. Take all cousins of this parent. Then, take all children of these cousins. Next, remove the objects that are not members of layer i. Finally, sort the remaining list by similarity to p and keep just $12D^4$ most similar ones. Return the selected ones as the cousins of p.

Nearest Neighbor Search. We start the walk over layered list $\mathcal{L}$ from $p^{(0)} = p_1 \in L_0$. Then for every i given some $p^{(i)} \in L_i$ we retrieve the endpoints of navigation pointers from $p^{(i)}$ to layer $i+1$, sort them by similarity to query object q and set $p^{(i+1)}$ to be the best of them. The search algorithm returns $p^{(k)}$ from the last layer k as the nearest neighbor of q in the dataset S.

3.3 Analysis

Lemma 3.1. (Properties of data structures)
1. Layer size $|L_i|$ is at most $2D\frac{n}{r_i} = 2D \cdot 2^i$

2. *For any $R \geq r$, and any object v, there are at most $4D^2(\frac{R}{r})$ centers x in an r-net with rank$_v(x) \leq R$.*

Proof. 1. All combinatorial balls of radius $r_i/2D$ around the L_i members are disjoint. Otherwise, there exist $p, p' \in L_i$, $z \in S$ *s.t.* $\text{rank}_p(z) < \frac{r_i}{2D}$ and $\text{rank}_{p'}(z) < \frac{r}{2D}$. But then, by the disorder inequality, $\text{rank}_p(p') < r$, which contradicts the separation property of combinatorial net. Note that we can have at most $n/(r_i/2D) = 2Dn/r_i$ disjoint balls with size $r_i/2D$ in an n-point set S. Therefore $|L_i| \leq 2Dn/r_i$.

2. Consider $r/2D$ balls around all centers x *s.t.* $\text{rank}_v(x) \leq R$. They are disjoint. By the disorder inequality the members of these "core balls" having rankat most $2DR$ to v, Therefore, the number of x is no more than $2DR/(r/2D) = 4D^2(R/r)$.

Theorem 3.1. (Preprocessing Analysis)

1. *The procedure for computing a new ball procedure returns the true members of the ball.*

2. *The procedure for computing navigation pointers procedure returns all of the true navigation links (possibly with false positives).*

3. *The procedure for computing cousin pointers returns all of the true cousins (possibly with false positives).*

4. *The total time complexity of preprocessing is $\mathcal{O}(D^7 n \log^2 n)$ and the space complexity is $\mathcal{O}(D^5 n + Dn \log n)$.*

Proof. 1. Let p be a new added center to the layer i and p' be its true children. By definition of cousins, their layer-$(i-1)$ parents are cousins. Thus, we will discover all the true members of $B(p, r_i)$. Sorting the list of candidates and keeping just r_i best of them gives us the exact set of ball members.

2. Let $p \to p'$ be an eligible navigation pointer from layer $i-1$ to layer i. Then $\text{rank}_p(p') \leq 3D^2 r_i$. Let us look at their parents s, s' on level $i - \log(6D^2)$. We have $\text{rank}_s(p) < 6D^2 r_i, \text{rank}'_s(p') < 6D^2 r_i$. Thus, s and s' must be cousins. Therefore, our walk on $p \to$ a parent $\to$ its cousins $\to$ their children will discover all endpoints of the true navigation links.

By Lemma 2.2 there are at most $12D^4$ centers in L_i that have rankat most $3D^2$ to p. Thus, if we sort candidates by their similarities to p and keep $12D^4$ best of them, the true navigation pointers will survive.

3. Let p, p' be an eligible cousin pair in layer i. Then, there is a chain with ranks $r_i - r_i/2 - r_i$ between them. Applying the disorder inequality twice we have that the rankfrom cousin to cousin is at most $D(D(r_i + r_i/2) + r_i) < 3D^2 r_i$. Consider their parents s, s' on level $i - \log(6D^2)$. We have $\text{rank}_s(p) < 6D^2 r_i, \text{rank}_{s'}(p') < 6D^2 r_i$. Thus, s and s' must be cousins. Therefore, our walk on $p \to$ a parent $\to$ its cousins $\to$ their children will discover all endpoints of all true cousin links.

By Lemma 2.2 there are at most $12D^4$ centers in L_i that have rankat most $3D^2$ to p. Thus, if we sort candidates by their similarity to p and keep $12D^4$ best of them, the true cousins will survive.

4. In each iteration i,

- Computing list of uncovered object requires $\mathcal{O}(|L_{i-1}|r_{i-1}) = \mathcal{O}(Dn)$.
- Computing a single new ball requires $\mathcal{O}(D^4 r_{i-1} \cdot \log(D^4 r_{i-1})) = \mathcal{O}(D^4 r_{i-1} \log n)$. Thus, the overall time for constructing new balls is at most $|L_i| \mathcal{O}(D^4 r_{i-1} \log n) = \mathcal{O}(D^5 n \log n)$.
- Computing navigation links for a single layer-$(i-1)$ object requires $\mathcal{O}(D^4 \cdot 6D^2 r_i \cdot \log(D^4 \cdot 6D^2 r_i)) = \mathcal{O}(D^6 r_i \log n)$. Thus, the overall time for constructing a new layer of navigation links is at most $|L_{i-1}| \cdot \mathcal{O}(D^6 r_{i-1} \log n) = \mathcal{O}(D^7 n \log n)$.
- Computing cousin links for a single layer-i object requires $\mathcal{O}(D^4 \cdot 6D^2 r_i \cdot \log(D^4 \cdot 6D^2 r_i)) = \mathcal{O}(D^6 r_i \log n)$. Thus, the overall time for constructing cousin pointers in a new layer is at most $|L_{i-1}| \cdot \mathcal{O}(D^6 r_{i-1} \log n) = \mathcal{O}(D^7 n \log n)$.

Putting everything together and sum up for all $\log n$ layers, we get an upper bound of $O(D^7 n \log^2 n)$ for the total time complexity.

And finally, here are the upper bounds on the space complexity of our algorithm:

$$|\mathcal{L}| = n$$

$$|\mathcal{D}| = |\mathcal{I}| = \sum_{i=1}^{\log n} \frac{n}{2^i} 2D2^i = \mathcal{O}(Dn \log n),$$

$$|\mathcal{N}| = \mathcal{O}(D^4) \sum |L_i| = \mathcal{O}(D^5 n)$$

,

$$|\mathcal{C}| = \mathcal{O}(D^4) \sum |L_i| = \mathcal{O}(D^5 n)$$

Theorem 3.2. (Search Analysis)

1. *Our search algorithm always returns the exact nearest neighbor of q.*

2. *The time complexity is $\mathcal{O}(D^4 \log n)$.*

Proof. Let x_{NN} be the closest object to q in the whole dataset and x_i^* be closest object within the layer i. First we will show that $\text{rank}_q(x_i^*) \leq Dr_i$ for all i.

Let z_i be the layer-i parent of x_{NN}. Thus, we have $\text{rank}_{z_i}(x_{NN}) \le r_i - 1$. And since $\text{rank}_q x_{NN} = 1$, we have $\text{rank}_q(z_i) < Dr_i$. However, x_i^* is the i-layer center closest to q, and therefore $\text{rank}_q(x_i^*) \le Dr_i$ as well.

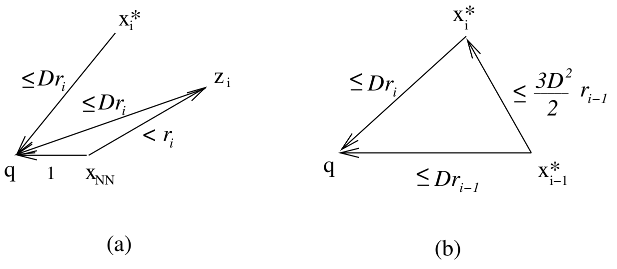

Figure 1: Illustrations for the search analysis

We will further show that the $p^{(i)}$ obtained in the searching algorithm is actually x_i^* defined as above. We prove this by induction. The base case is trivially true. Suppose $p^{i-1} = x_{i-1}^*$. Then since $\text{rank}_q(x_{i-1}^*) \le Dr_{i-1}$ and $\text{rank}_q(x_i^*) \le Dr_i$, we have $\text{rank}_{x_{i-1}^*}(x_i^*) \le 3D^2 r_i$ by the disorder inequality. Since the navigation pointers from x_{i-1}^* go to all objects in the next that are $3D^2r_i$-close to x_{i-1}^*, we know that x_i^* has been linked from x_{i-1}^*. Thus the $p^{(i)}$ founded in the searching algorithm is the x_i^*. And finally, since the last layer contains all points in S, the best one x_k^* is the neatest neighbor x_{NN}.

The searching algorithm uses $O(D^4 \log n)$ time to find the nearest neighbor again because there are at most $O(D^4)$ navigation links for each object and layer number.

Remark. In practice we may not know the disorder constant D in advance. Well, we suggest to estimate D via random sampling and use this value in implementation. We also suggest to be more optimistic and use smaller values instead of $\mathcal{O}(D^4)$ worst-case bounds on number of navigation and cousins pointers.

4 Detecting Near-Duplicates

Assume, given a pair x, y our oracle can also tell whether the similarity value $\sigma(x, y)$ is above or below some particular fixed threshold T. The problem of Near-Duplicate Detection asks for an algorithm to find all pairs whose similarity is above the threshold.

THEOREM 4.1. *(Near-duplicate detection) For a given dataset of n objects with disorder constant D, all pairs with similarity above some threshold T can be found in $poly(D)(n \log^2 n + |Output|)$ time. Here $|Output|$ denotes the actual number of near-duplicate pairs.*

Proof. As the first step we compute the same five data structures as in nearest neighbor solution. Now let us describe an auxiliary procedure called *k-check*: it checks whether a particular object x has less than k near-duplicates. In the case of positive answer, the procedure outputs all these objects. Using the inverted index $\mathcal{I}$ we take x's parent y in layer i with $r_i = 2k$. Then we get all cousins of y and collect all their layer-i children. Finally, we check how many of these collected objects have over-T similarity to x. By the disorder inequality and the definition of cousin, all first k neighbors of x are retrieved: they are covered by some cousin y' of y. If the number of discovered near-duplicates is smaller than k then all near-duplicates are actually found. Since there are $poly(D)$ cousins, and each cousin has the covering list of length $2k$, the running time of k-check is $k \cdot poly(D)$ (no matter whether the answer if positive or not).

Now the algorithm is as follows. Perform k-check for x with range $k = 1, 2, 4, 8, \ldots$, until the procedure finds less than k duplicates. Then, indeed, all near-duplicates are found. The total complexity of all k-checks is still $k \cdot poly(D)$ due to summation formula for geometric progressions. Thus, if we denote j_x to be the largest j *s.t.* the j-th neighbor y of x satisfies $\sigma(x, y) > T$, we obtain the following upper bound on the running time: $poly(D) n \log^2 n + \sum_x poly(D) j_x = poly(D)(n \log^2 n + |Output|)$. The first item comes from computing our five basic data structures.

5 The Visibility graph

Assume the layered list $\mathcal{L}$ is already constructed. Then for any object in the dataset we can define its *visible neighbors* as follows: for every $0 \le i \le \lceil \log n \rceil$ connect each object p to centers in layer i that are $3D^2r_i$-close to p. We use term "visible" because a center is visible if its rankto p is at most $3D^2$ larger than its radius r_i: The farther you are, the larger radius you need to be visible.

THEOREM 5.1. *Out-degrees in visibility graph are at most $\mathcal{O}(D^4 \log n)$. The greedy walk deterministically converges to the exact nearest neighbor in at most $\log n$ moves.*

Proof. We add links from each point p to those centers in layer i that are $3D^2r_i$-close to p. By Lemma 2.2 there are at most $\mathcal{O}(D^4)$ links to the centers in layer i, and thus the total degree of a point is $\mathcal{O}O(D^4 \log n)$. Now let us study the greedy walk: at step i we move from object u_{i-1}^* to its visible neighbor u_i^* having the maximal similarity to q. We stop when all visible neighbors are worse than the current point. Again, let x_i^* be the closest to q center in layer i.

Claim $\operatorname{rank}_q(u_i^*) \le \operatorname{rank}_q(x_i^*)$ for all i.

Proof. We prove this by induction. Suppose the inequality is true for i. Since u_i^* has links to all centers p in layer $i+1$ with $\operatorname{rank}_{u_i^*}(p) \le 3D^2 r_{i+1}$, it is enough to show that $\operatorname{rank}_{u_i^*}(x_{i+1}^*) \le 3D^2 r_{i+1}$. The later statement is proven in the same way as $\operatorname{rank}_{x_i^*}(x_{i+1}^*) \le 3D^2 r_{i+1}$ inequality in Theorem 4.

By claim statement for $i = \log n$, after at most i steps we reach the nearest neighbor.

6 Directions for Further Work

The combinatorial framework leads to new results that were not known in other models: subquadratic deterministic detection of near duplicates, short certificates for being nearest neighbor, generalized analog of Delaunay graph. It also leads to new techniques: up-aside-down-filter trick and false positives in data structures.

The combinatorial framework requires further improvement. What if the D in the disorder inequality is small *on average* instead of in the worst case? If this is too restricting, what is the largest fraction of bad triples we can handle? In some scenarios, not all pairs are comparable, so for each x, the other $n-1$ points only form a partial order. How should we change our assumption and algorithms? In general, it is important to find a combinatorial notion of dimension that is robust to exceptions and perturbations but still provide efficient algorithms. When the disorder constant is not given, can we compute (or at least obtain some probabilistic guarantees) for its value in subquadratic time? How to handle insertions and deletions, (important for using visibility graph in network design), and dynamically changing parameters in similarity functions? Recall, insertions and deletions can be efficiently supported in case of small expansion rate [14] or doubling dimension [25]. We need more experimental work to validate disorder inequality assumption and get the real-world performance of our algorithms.

Finally, the combinatorial framework can lead to further progress on other problems and previously studied models. Does there exist an analogue of visibility graph for the datasets with small doubling dimension? Are there short certificates for being nearest neighbor in other models? Also can combinatorial and numerical methods be combined in some useful way? Can other problems dealing with distances be stated and efficiently solved in the combinatorial framework? It seems that for some applications replacing distances by ranks can be meaningful. In particular, it is interesting to consider linear arrangement problem [11], closest pairs [15], distance labelling [32], shortest paths [1], detecting communities [30], and dimensionality reduction [8]. How our method should be modified for bipartite problems (like users-ads matching [27]), where similarities are defined only for bichromatic pairs? We conclude with the following *Unification challenge*:

Is there a general framework for efficient solutions to similarity problems that contains both doubling dimension and the combinatorial approach as specific subcases?

Acknowledgements. Authors thank Shuki Bruck, Ken Clarkson, Piotr Indyk, Leonard Schulmann, Alex Slivkins and especially Hsuan-Tien Lin for fruitful discussions.

References

[1] I. Abraham, C. Gavoille, A.V. Goldberg, and D. Malkhi. Routing in networks with low doubling dimension. In *ICDCS'06*.

[2] S. Arya and D. M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *SODA'93*, pages 271–280, 1993.

[3] M.-F. Balcan, A. Blum, and S. Vempala. A discriminative framework for clustering via similarity functions. In *STOC'08*.

[4] Z. Bar-Yossef, I. Keidar, and U. Schonfeld. Do not crawl in the DUST: different urls with similar text. In *WWW'07*.

[5] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *ICML'06*.

[6] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the Web. In *WWW'97*.

[7] A.Z. Broder, M. Charikar, A.M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *STOC'98*.

[8] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Transactions on Database Systems*, 2002.

[9] T.H.H. Chan, M. Dinitz, and A. Gupta. Spanners with slack. In *ESA'06*.

[10] M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC'02*.

[11] M. Charikar, K. Makarychev, and Y. Makarychev. A divide and conquer algorithm for d-dimensional linear arrangement. In *SODA'07*.

[12] K.L. Clarkson. Nearest neighbor queries in metric spaces. *Discrete and Computational Geometry*, 1999.

[13] K.L. Clarkson. Nearest-neighbor searching and metric space dimensions. In Nearest-Neighbor Methods for Learning and Vision: Theory and Practice, MIT Press, 2006.

[14] R. Cole and L.-A. Gottlieb. Searching dynamic point sets in spaces with bounded doubling dimension. In *STOC'06*.

[15] A. Corral, Y. Manolopoulos, Y. Theodoridis, and M. Vassilakopoulos. Closest pair queries in spatial databases. In *SIGMOD'00*.

[16] P. Fraigniaud, E. Lebhar, and Z. Lotker. A doubling dimension threshold $\theta(\log\log n)$ for augmented graph navigability. In *ESA'06*.

[17] N. Goyal, Y. Lifshits, and H. Schütze. Disorder inequality: A combinatorial approach to nearest neighbor search. In *WSDM'08*.

[18] S. Har-Peled and M. Mendel. Fast construction of nets in low dimensional metrics, and their applications. *SoCG'05, SIAM Journal on Computing 2006*.

[19] K. Hildrum, J. Kubiatowicz, S. Ma, and S. Rao. A note on the nearest neighbor in growth-restricted metrics. In *SODA'04*.

[20] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC'98*.

[21] D.R. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *STOC'02*.

[22] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *STOC'00*.

[23] A. Kolcz, A. Chowdhury, and J. Alspector. Improved robustness of signature-based near-replica detection via lexicon randomization. In *KDD'04*.

[24] R. Krauthgamer and J. R. Lee. The black-box complexity of nearest-neighbor search. *ICALP'04, Theoretical Computer Science 2005*.

[25] R. Krauthgamer and J. R. Lee. Navigating nets: simple algorithms for proximity search. In *SODA'04*.

[26] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *STOC'98, SIAM J. Comput. 2000*.

[27] Y. Lifshits and D. Nowotka. Estimation of the click volume by large scale regression analysis. In *CSR'07*.

[28] R. Macíyas and C. Segovia. Lipschitz functions on spaces of homogeneous type. *Advances in Mathematics*, 1979.

[29] G. Navarro. Searching in metric spaces by spatial approximation. *SPIRE'99, J. VLDB 2002*.

[30] M.E.J. Newman. Detecting community structure in networks. *The European Physical Journal B-Condensed Matter*, 2004.

[31] C.G. Plaxton, R. Rajaraman, and A.W. Richa. Accessing nearby copies of replicated objects in a distributed environment. *Theory Comput. Syst.*, 1999.

[32] A. Slivkins. Distance estimation and object location via rings of neighbors. *PODC'05, J. Distributed Computing 2007*.

[33] B. Wong, A. Slivkins, and E.G. Sirer. Meridian: A lightweight network location service without virtual coordinates. In *SIGCOMM'05*, 2005.

[34] P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search - The Metric Space Approach.* Springer, 2006.

A Indyk's Examples

Small doubling constant but large disorder: A union of (slightly perturbed) points on a circle, with its center z. Here, the doubling dimension is constant, while the disorder dimension is high. Specifically, the two nearest neighbors x and y of z could be a pair of antipodal points, but $\text{rank}_x(y)$ could be very large.

Small disorder but large doubling constant: A set of n points $p_1, ..., p_n$, such that, if $i \neq j, d(p_i, p_j) = 10n + |i-j|$. The rankfor ties are broken arbitrarily. It is not hard to see that $i, j,\ |i-j| \leq \text{rank}_{p_i}(p_j) \leq 2|i-j|$, therefore, the disorder constant is at most 2. However, note that the points are almost equidistant, in which case the doubling constant and the expansion rate is close to n.