Small Candidate Set for Translational Pattern Search

Ziyun Huang
Department of Computer Science and Software Engineering, Penn State Erie, The Behrend College, Erie, PA, USA
zxh201@psu.edu

Qilong Feng
School of Computer Science and Engineering, Central South University, P.R. China
csufeng@mail.csu.edu.cn

Jianxin Wang
School of Computer Science and Engineering, Central South University, P.R. China
jxwang@csu.edu.cn

Jinhui Xu
Department of Computer Science and Engineering, State University of New York at Buffalo, USA
jinhui@buffalo.edu

Abstract

In this paper, we study the following pattern search problem: Given a pair of point sets A and B in fixed dimensional space $\mathbb{R}^d$, with $|B| = n, |A| = m$ and $n \geq m$, the pattern search problem is to find the translations $\mathcal{T}$'s of A such that each of the identified translations induces a matching between $\mathcal{T}(A)$ and a subset B' of B with cost no more than some given threshold, where the cost is defined as the minimum bipartite matching cost of $\mathcal{T}(A)$ and B'. We present a novel algorithm to produce a small set of candidate translations for the pattern search problem. For any $B' \subseteq B$ with $|B'| = |A|$, there exists at least one translation $\mathcal{T}$ in the candidate set such that the minimum bipartite matching cost between $\mathcal{T}(A)$ and B' is no larger than $(1 + \epsilon)$ times the minimum bipartite matching cost between A and B' under any translation (i.e., the optimal translational matching cost). We also show that there exists an alternative solution to this problem, which constructs a candidate set of size $O(n \log^2 n)$ in $O(n \log^2 n)$ time with high probability of success. As a by-product of our construction, we obtain a weak ϵ-net for hypercube ranges, which significantly improves the construction time and the size of the candidate set. Our technique can be applied to a number of applications, including the translational pattern matching problem.

2012 ACM Subject Classification Theory of computation → Pattern matching; Theory of computation

Keywords and phrases Bipartite matching, Alignment, Discretization, Approximate algorithm

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2019.26

Funding The research of the first and last authors was supported in part by NSF through grant CCF-1716400. The research of the last author was also supported by NSF through grant IIS-1910492. The research of the second and third authors was supported in part by NSFC through grants 61872450, 61828205, and 61672536.

1 Introduction

Pattern search/matching is an important problem in computer science and finds applications in many different domains such as computer vision, pattern recognition, robotics, autonomous driving, and surveillance. In this paper, we consider a special variant of the problem, where the objective is to find a small pattern (e.g., the image of a car) from a large environment (called background; e.g., the image of a road with traffic) which may contain multiple copies

of the pattern or its deformations. The problem is often encountered in our daily life. For example, most of the smart phones have the capability of identifying human faces (or other types of objects) in their camera softwares. The core problem for such a functionality is to efficiently find all appearances of a given object from the pictures. The pattern search problem may also appear in higher ($d > 2$) dimensional space. One of such examples is the pattern extraction problem arising in biological image analysis, where the objective is to identify the 3D spatial positioning patterns of chromosomes [9, 10, 22].

We approach this pattern search problem from a geometric perspective, using a formulation from [1] with some slight modifications. The pattern is represented by a point set A and the background by a point set B in $\mathbb{R}^d$ space for some fixed d. The sizes of A and B are m and n, respectively, with $n \geq m$ (note that n could be significantly larger than m). If there is a translation $\mathcal{T}$ which moves A to a new location $\mathcal{T}(A)$ such that the *difference* between $\mathcal{T}(A)$ and some $B' \subseteq B$ with $|B'| = |A|$ is minimized, we say that an instance B' of pattern A is discovered by $\mathcal{T}$, where the difference of two sets is measured by their bipartite matching cost (see Section Preliminaries).

The pattern search problem is a natural extension of the pattern matching problem, whose aim is to find a rigid transformation that minimizes the difference of two given sets A and B. Extensive research has been done for the pattern matching problem using different metrics as the measurement for the similarity/distance of two sets [13, 11, 14]. Commonly used metrics include Euclidean distance in 1-to-1 matching, Hausdorff distance in 1-to-many or many-to-1 matching, and Earth's Mover Distance (EMD) in many-to-many matching. An early result on this problem is the paper [18] which provides an $\widetilde{O}(mn^2)$-time solution to the matching problem under translation and Hausdorff distance in $\mathbb{R}^2$. A more recent result is the one in [11] which approximates (with ratio $(1+\epsilon)$) the pattern matching problem under rigid transformations and EMD metric in $\mathbb{R}^d$. The running time of their algorithm is $\widetilde{O}((mn)^{2d})$, which is near the lower bound (i.e., $\Omega(mn^{\Omega(d)})$ [5]) of the problem.

For the pattern search problem under translations, there is a number of results [21, 4] that are closely related to the work in this paper. Most of them use the concept of *partial-matching Voronoi diagram*.

For two given point sets A and B, their partial-matching Voronoi diagram (PMVD) is a partition of the translation space into regions so that each of them consists of translations $\mathcal{T}$ sharing the same locally optimal bipartite matching between $\mathcal{T}(A)$ and a subset of B. The PMVD uses the sum of squared distances as the measurement for the matching cost. Clearly, such a Voronoi diagram is capable of solving the pattern search/matching problem, as only one translation from every cell needs to be determined for finding the optimal translational alignment of A and B. The best known upper bound on the size of PMVD is $O(m!m^d n^{2d})$ [17]. Ben-Avraham et al. [4] constructed a partial-matching Voronoi diagram in $\mathbb{R}^2$ of complexity $O(n^2 m^{3.5} \log^m m)$, and found *locally* min-cost translations in $O(m^6 n^3 \log n)$ time.

In this paper, we develop a novel method for finding a small set $\mathbb{T}$ of candidate translations so that for any instance $B' \subseteq B$ of A, there is at least one translation $\mathcal{T}$ in the candidate set that matches $\mathcal{T}(A)$ and B' approximately. A subset B' of B is called an instance of A if $|B'| = |A|$. Note that B' can be any subset of B as long as it has the same size as A and may have a large difference with A; this is somewhat different from the normal meaning of instance. We say that a translation $\mathcal{T}$ *discovers* an instance B' if it minimizes the bipartite matching cost of $\mathcal{T}(A)$ and B'. For any $\epsilon > 0$, a translation $\mathcal{T}$ $(1+\epsilon)$-approximately discovers B', if the bipartite matching cost between $\mathcal{T}(A)$ and B' is no more than $(1+\epsilon)$ times the minimum difference between B' and A under any translation. Clearly, with such a candidate set $\mathbb{T}$, we are able to find all instances B' which are similar to A, where the level of similarity

is controlled by some threshold on the difference of B' and the translations of A. Note that if a value of the threshold is given in advance, it is possible to further reduce the size of the candidate set by removing (during the execution of our algorithm) those translations which induce higher matching cost (see the remark in Section 6 for more details). Also, if B has some exact (or congruent) instances of A, $\mathbb{T}$ will contain all translations inducing zero-difference matchings of A.

The problem of finding the translations that match pattern A to all its exact instances could be quite challenging, as suggested by the exponential size of PMVD in [21, 4]. However, we are able to show that if approximation and implicit representation are allowed, the problem can be solved much more efficiently through identifying a small candidate set of translations with a (surprisingly) near linear size.

Particularly, we show that it is possible to build a candidate set $\mathbb{T}$ with size $O_{d,\epsilon}(n \log n)$ for A and B in $O(mn \log mn)$ deterministic time. This bound is asymptotically near optimal, since it is easy to construct an example that needs $O(n)$ different translations to yield all perfect matches of a pattern (such an example will be given later). A trade-off between the running time and the size of $\mathbb{T}$ can also be made, which provides a probabilistic algorithm to build a $\mathbb{T}$ with a slightly larger size (i.e., $O(n \log^2 n)$) but a better time complexity (i.e., $O(n \log^2 n)$) which is independent of m. Our construction is based on a weak ϵ-net technique and a space discretization technique from [7]. Our approach shows a non-obvious connection between weak ϵ-net and the pattern search problem. A fast algorithm is also provided to build a small-size ϵ-net for ranges of axis-aligned hypercubes.

In some sense, candidate set $\mathbb{T}$ can be viewed as an implicit and approximate representation of the exponential-size PMVD. Thus, it has the potential to be used in various applications of PMVD, such as moving object tracking and autonomous driving. Note that in such applications, all translations in $\mathbb{T}$ (rather than those inducing better matchings) are needed.

2 Preliminaries

In this paper, we do not distinguish a point and its corresponding vector in $\mathbb{R}^d$, i.e., a point is equivalent to a vector that points from the origin to it. In this way, a point in $\mathbb{R}^d$ naturally defines a translation (along the corresponding vector) in $\mathbb{R}^d$. Following the basic vector arithmetics, operators $+$ and $-$ can be applied to points in $\mathbb{R}^d$.

Given two point sets A_1 and A_2 of the same size, their bipartite matching is represented by a bijective mapping $\phi : A_1 \to A_2$. That is, each point $a \in A_1$ is matched to a distinct point $\phi(a) \in A_2$ and the cost C_{ϕ,A_1,A_2} of the matching is defined as $C_{\phi,A_1,A_2} = \sum_{a \in A_1} \|a - \phi(a)\|$. The difference of A_1 and A_2, denoted by $\Delta(A_1, A_2)$, is then $\Delta(A_1, A_2) = \min_\phi C_{\phi,A_1,A_2}$.

Let $A, B \subset \mathbb{R}^d$ be the two point sets in the pattern search problem, with $|B| = n, |A| = m$ and $n \geq m$. We label points in A and B, respectively, as $A = \{a_1, a_2, \ldots, a_m\}$ and $B = \{b_1, b_2, \ldots, b_n\}$. The **reference set** P of A and B is defined as follows.

▶ **Definition 1.** *Let $p_{i,j} = b_j - a_i$ for any $a_i \in A$ and $b_j \in B$. The reference set P of A and B is the muti-set that contains all $p_{i,j}$ for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.*

We use an injective mapping $\phi : \{1, 2, \ldots, m\} \to \{1, 2, \ldots, n\}$ to represent a perfect matching for A (under a certain translation) and a subset of B. $\phi(i) = j$ means that a_i is matched to b_j. The matching cost C_ϕ for a matching ϕ is defined as $\sum_{i=1}^{m} \|a_i - b_{\phi(i)}\|$.

From the definition of P, it is clear that for any $a_i \in A$, $b_j \in B$ and translation $\mathcal{T} \in \mathbb{R}^d$, $\|\mathcal{T}(a_i) - b_j\| = \|\mathcal{T} - p_{i,j}\|$. The matching cost $C_\phi(\mathcal{T})$ between $\mathcal{T}(A)$ and B for matching ϕ is then $\sum_{i=1}^{m} \|\mathcal{T} - p_{i,\phi(i)}\|$. In other words, $C_\phi(\mathcal{T})$ is the sum of distances from $\mathcal{T}$ to m points in P. For convenience, we let $P(\phi) = \{p_{1,\phi(1)}, p_{2,\phi(2)}, \ldots, p_{m,\phi(m)}\}$. Thus $C_\phi(\mathcal{T}) = \sum_{p \in P(\phi)} \|\mathcal{T} - p\|$.

In the rest of the paper, we study all possible matchings between A and B based on the relationship of $\mathcal{T} \in \mathbb{R}^d$ and P. This means that our algorithms work in the translational space of P, instead of the original space of A and B.

For any pair of point sets X and Y with the same cardinality, we use $\Delta(X, Y)$ to denote the minimum bipartite matching cost of X and Y. A set $B' \subseteq B$ is called an instance of A if $|B'| = |A|$. For any instance B', let $\mathcal{T}$ be the translation that minimizes $\Delta(\mathcal{T}(A), B')$. Then, we say that $\mathcal{T}$ *discovers* B', or B' is discoverable at $\mathcal{T}$. If there is another translation $\mathcal{T}'$ satisfying the following inequality, $\Delta(\mathcal{T}'(A), B') \leq (1+\epsilon)\Delta(\mathcal{T}(A), B')$, then, we say that B' is $(1+\epsilon)$-approximately discoverable at $\mathcal{T}'$.

3 Main Results

The main results of this paper are the following theorems which show that for any given pair of point sets A and B, and any constant $\epsilon > 0$, it is possible to efficiently construct a small-size candidate set $\mathbb{T}$ of translations in $\mathbb{R}^d$ such that any instance $B' \subseteq B$ of A is approximately discoverable at some $\mathcal{T} \in \mathbb{T}$. In other words, $\mathbb{T}$ is a small-size candidate set of translations to find all instances of A approximately.

▶ **Theorem 2.** *For any pair of point sets A and B in fixed dimensional space $\mathbb{R}^d$ with size m and n ($n \geq m$), respectively, and any small constant $0 < \epsilon < 1$, it is possible to construct, deterministically, a candidate set $\mathbb{T} \subset \mathbb{R}^d$ of size $O(n \log n)$ in $O(mn \log mn)$ time such that for any given instance $B' \subseteq B$ of A, there exists a translation $\mathcal{T} \in \mathbb{T}$ that $(1+\epsilon)$-approximately discovers B'.*

▶ **Theorem 3.** *For any pair of point sets A and B in fixed dimensional space $\mathbb{R}^d$ with size m and n ($n \geq m$), respectively, and any small constant $0 < \epsilon < 1$, it is possible to construct a candidate set $\mathbb{T} \subset \mathbb{R}^d$ of size $O(n \log^2 n)$ in $O(n \log^2 n)$ time, with success probability at least $1 - 1/n$. For any given instance $B' \subseteq B$ of A, there exists a translation $\mathcal{T} \in \mathbb{T}$ that $(1+\epsilon)$-approximately discovers B'.*

With the above theorems, we immediately have the following corollary as their application to the classical pattern matching problem. (See the appendix for the proof.)

▶ **Corollary 4.** *It is to possible to generate a set of $O(n \log n)$ candidate translations in $\mathbb{R}^d$ in $O(mn \log mn)$ time such that one of them induces a $(1+\epsilon)$-approximation for the optimal translational matching between A and B.*

Another interesting conclusion from the above discussion is that if there exists an instance $B' \subseteq B$ which is identical to A under translation $\mathcal{T}$ (i.e., the bipartite matching cost between B' and $\mathcal{T}(A)$ is 0), then $\mathcal{T} \in \mathbb{T}$.

The above theorems and corollary suggest an efficient way to identify a small number of translations that enable us to obtain approximate solutions to the translational pattern matching problem. The size of the candidate set is near optimal. This can be easily seen from the following simple example in 1-D: Consider $A = \{1, 2\}$ and $B = \{1, 2, \ldots, n\}$; then all the n translations that align 1 in A to any of the $n-1$ points $\{1, 2, \ldots, n-1\}$ in B are optimal translations.

After obtaining all the candidate translations, the approximate optimal matching can then be computed by solving a min-cost partial matching problem for fixed point sets $\mathcal{T}(A)$ and B for every candidate $\mathcal{T}$.

3.1 Overview of Techniques

Our main idea for constructing a small candidate set is via space discretization. We are able to prove a locality property for the pattern search problem: if the distance between two translations $\mathcal{T}_1$ and $\mathcal{T}_2$ (viewed as points in $\mathbb{R}^d$) is close compared to their closest distance to any point in the reference set P, then for any instance $B' \subseteq B$, $\Delta(\mathcal{T}_1(A), B')$ and $\Delta(\mathcal{T}_2(A), B')$ are also close. This suggests that we can decompose $\mathbb{R}^d$ into "small" regions, so that the every region has a small diameter, comparing to its distance to P. For any $B' \subseteq B$, let $\mathcal{T}_O$ be the translation that discovers B', and $\mathcal{T}_O$ lies in some "small" region C. Then, any $\mathcal{T}' \in C$ approximately discovers B'. This means that, if we choose one arbitrary point from each "small" region to form the candidate set $\mathbb{T}$, it is guaranteed that any instance B' will be approximately discoverable by some translation in $\mathbb{T}$. The details will be shown in Section 4.

However, making all regions of the entire space "small" seems to be challenging, if not impossible. Existing space discretization techniques, such as [7, 3, 15], are only able to ensure that regions distant from the points in P are "small". For regions close to points in P (called "close" regions), new techniques are needed to select their candidate translations. In Section 5, we discuss how to choose translations from "close" regions so that every instance B' whose corresponding optimal translation falls in some "close" region can also be approximately discoverable.

In Section 4.4, we also describe how to use weak ϵ-net to "sketch" P (with size mn) using a much smaller set Q of size $O(n)$. This will allow us to significantly reduce the size of discretization (and thus the size of the candidate set) from $\widetilde{O}(mn)$ to $\widetilde{O}(n)$, which is near optimal. We are able to show that such a sketching still preserves the locality property of the pattern search problem.

4 Locality Based Discretization for Pattern Search

In this section, we present a discretization approach for the pattern search problem, based on the locality property of the problem.

4.1 Locality of Pattern Search

In the context of pattern search, locality refers to the following observation: for any instance $B' \subseteq B$ and two translations $\mathcal{T}_1$ and $\mathcal{T}_2$, if $\mathcal{T}_1$ and $\mathcal{T}_2$ are close to each other, their induced minimum bipartite matching costs between the translations of A and B' are also close. Let $\Delta(X, Y)$ denote the minimum bipartite matching cost between two point sets X and Y. The following lemma shows the locality property with respect to the distance between translations and a point in the reference set P. (See Appendix for the proof.)

▶ **Lemma 5.** *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two translations in $\mathbb{R}^d$, and $p \in P$ be the nearest neighbor of $\mathcal{T}_1$ in the reference set P. If $\|\mathcal{T}_1 - \mathcal{T}_2\| \leq \epsilon \|p - \mathcal{T}_1\|$ for some constant $0 < \epsilon < 1$, then $|\Delta(\mathcal{T}_1(A), B') - \Delta(\mathcal{T}_2(A), B')| \leq \epsilon \Delta(\mathcal{T}_1(A), B')$ for any instance $B' \subseteq B$.*

The above locality property suggests the following discretization approach to find a candidate set for the pattern search problem. The idea is to decompose $\mathbb{R}^d$ into a number of "small" regions. Each region C is "small" enough in the sense that the minimum distance r between C and the points in P is large, comparing to the diameter $D(C)$ of C, i.e., $D(C) \leq \epsilon r$ for some constant $0 < \epsilon < 1$. With such a discretization, we may then simply choose one arbitrary translation from each region to form the candidate set $\mathbb{T}$. To see that this is indeed the desired candidate set, consider any instance $B' \subseteq B$. Let $\mathcal{T}$ be the translation that discovers B', C' be the region containing $\mathcal{T}$, and $\mathcal{T}_C \in \mathbb{T}$ be the translation chosen from C. Then, by Lemma 5, we know that B' is $(1 + \epsilon)$-approximately discoverable at $\mathcal{T}_C$.

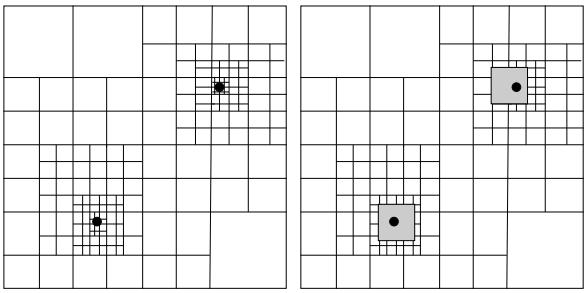

Figure 1 Illustrative figures for "small" and "close" regions. Left Figure: If every region needs to be "small", then an infinite number of regions will be generated around each point in P. Right Figure: Possible "close" regions to prevent from yielding an infinite number of "small" regions.

4.2 Space Discretization and Close Regions

Unfortunately, an exact implementation of the above space discretization is not possible. This is because the size of some regions can be infinitely small if the distance of the region to a point in P is small enough. This means that an infinite number of regions can be generated around every point in P (see Figure 1). To overcome this difficulty, a possible way is to utilize some of the known space discretization techniques, such as [7, 3, 15]. However, a common issue of such techniques is that only part of the resulting regions can be viewed as "small" (i.e., $D \leq \epsilon r$, where D is the diameter of the region and r is the minimum distance between the a point in P and the region). Such regions are distant from points in P. All other regions are in close proximity to points in P, and cannot be viewed as "small", even though their diameters might be small. We call such regions as "close" regions.

Clearly, for "small" regions, it will be sufficient (by Lemma 5) to choose one point arbitrarily from each of them and include it into the candidate set $\mathbb{T}$. The main issue is, thus, how to select candidate translations from the "close" regions. We will discuss our ideas on close regions in next section.

In this paper, we use the technique in [7] for space discretization. The main reason is that using this technique, we have some good geometric properties on close regions. This will help us ensure the correctness of our proposed approach and simplify the analysis.

For self completeness, below we summarize the space discretization technique in [7]. The main technique in [7] is an algorithm AIDecomposition(P, β, γ) , where P is a point set in $\mathbb{R}^d$ and $0 < \beta, \gamma < 1$ are two small constants (to be determined in later analysis). The following lemma is the main result of the algorithm.

▶ **Lemma 6.** *[7] AIDecomposition(P, β, γ) generates a partition of $\mathbb{R}^d$ in $O_{d,\beta,\gamma}(|P| \log |P|)$ time, where each region C of the partition satisfies one of the following conditions.*

1. *C is associated with a subset V of P and a point $v \in V$, such that*
   - **a.** *The diameter $D(V)$ of V is no larger than βr, where r is the closest distance between a point in C and a point in V.*
   - **b.** *$\|v - u\| \leq \gamma \|v' - u\|$ and $D(C) \leq \beta \|v' - u\|$, for any point $u \in C$ and any point $v' \in P \setminus V$*
2. *$D(C) \leq \beta r$, where r is the closest distance between a point in C and a point in P.*

The regions that satisfy condition 1 are the close regions in our previous discussion, and the regions that satisfy condition 2 correspond to the small regions. Note that for a close region C generated by AIDecomposition, it is close to the associated point set $V \subset P$,

comparing to points in $P \setminus V$, where the closeness is controlled by the parameter γ. There are also some other interesting properties of close regions generated by AIDecomposition shown in Lemma 6. These properties will prove to be useful in later analysis.

4.3 Reducing the Number of Regions

If we directly apply the above AIDecomposition technique or any other space discretization technique (such as [3, 15]) to the reference set P, at least $\Omega(mn)$ regions will be generated, since $|P| = mn$. This results in a candidate set of size $\Omega(mn)$, which could be significantly larger than $O(n)$, i.e., the maximum number of possible translations that could yield a perfect matching for $\mathcal{T}(A)$ and B. Thus, it is tempting to ask whether it is possible to construct a discretization with size only near $O(n)$.

A natural approach for size reduction is to use a smaller set Q to "sketch" P, and build a discretization for Q, instead of P. Let $\xi > 1$ and $\mu > 0$ be some given constants. We require that Q be (ξ, μ)-**dense** for P, defined as follows.

▶ **Definition 7.** *A point set $Q \subset \mathbb{R}^d$ is called (ξ, μ)-dense for P, if for any point $\mathcal{T} \in \mathbb{R}^d$, it satisfies $\mu\|p_\xi - \mathcal{T}\| \geq \|q - \mathcal{T}\|$, where p_ξ is the m/ξ-th closest point in P to $\mathcal{T}$, and $q \in Q$ is the nearest neighbor of $\mathcal{T}$ in Q.*

In other words, points in Q are "dense" enough, so that for any $\mathcal{T} \in \mathbb{R}^d$, it is possible to find a point $q \in Q$ that is closer to, or not much farther away from $\mathcal{T}$ than p_ξ. Using such a formulation for "dense" allows us to use $\|q - \mathcal{T}\|$ to effectively lower bound $\Delta(\mathcal{T}(A), B')$ for any B'. Let ϕ be the matching realizing the minimum cost bipartite matching between $\mathcal{T}(A)$ and B'. Then, we have

$$\|q - \mathcal{T}\| \leq \mu\|p_\xi - \mathcal{T}\| \leq \sum_{p \in P(\phi)} \mu\|p - \mathcal{T}\|/((1 - 1/\xi)m) = \mu\Delta(\mathcal{T}(A), B')/((1 - 1/\xi)m). \quad (1)$$

Using an argument similar to the one in Lemma 5, we have the following improved version of locality property.

▶ **Lemma 8.** *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two translations in $\mathbb{R}^d$, and $p \in Q$ be the nearest neighbor of $\mathcal{T}_1$ in Q. If $\|\mathcal{T}_1 - \mathcal{T}_2\| \leq \beta\|p - \mathcal{T}_1\|$ for constant $0 < \beta < 1$, then $|\Delta(\mathcal{T}_1(A), B') - \Delta(\mathcal{T}_2(A), B')| \leq \beta\mu(1 - 1/\xi)^{-1}\Delta(\mathcal{T}_1(A), B')$ for any instance $B' \subseteq B$.*

The above lemma enables us to use Q for the space discretization. More specifically, we run AIDecomposition(Q, β, γ) on an (ξ, μ)-dense Q (with β, ξ, μ and γ to be determined later). This yields a near linear size (in terms of the size of Q) discretization. From previous discussion, we know that for any instance $B' \subseteq B$, if the translation that discovers B' lies in a small region (note that since the discretization is based on Q instead of P, "small region" now means that their diameters are small compared to the distances to their nearest neighbors in Q), then any translation in the small region will $(1 + \epsilon)$-approximately discover B', if parameters β, ξ, μ and γ are properly chosen according to the desired approximate ratio ϵ. A formal argument will be provided later when analyzing the correctness of our algorithm. The remaining main challenge is then to deal with the case that the translation discovering B' lies in a close region. This will be covered in the next section.

4.4 Finding (ξ, μ)-dense Q for P

To conclude this section, we briefly describe how to find a (ξ, μ)-dense set Q for the discretization.

A set Q that is (ξ, μ)-dense can actually be found by constructing a weak ϵ-net of P [16]. ϵ-net is an important concept in combinatorial and computational geometry, and has been extensively studied in the past. For our problem, we use weak ϵ-net for axis-aligned hypercubes (i.e., hyper-boxes with equal edge length in every direction). Below is the definition.

▶ **Definition 9.** *Let P be any point set in $\mathbb{R}^d$, and ϵ be any small constant between 0 and 1. A point set $Q \subset \mathbb{R}^d$ is called a weak ϵ-net of P for axis-aligned hypercubes, if any axis-aligned hypercube G containing $\epsilon|P|$ or more points in P also contains at least one point in Q.*

Note that ϵ-net is a much more general concept than the above definition. In this paper, we only consider weak ϵ-net for axis-aligned hypercubes. For convenience, we will use thereafter the term "ϵ-net" without specifying "weak" and "axis-aligned hypercubes". For any constant $c > 1$, it is easy to see that a $1/cn$-net Q of P is $(c, \sqrt{d})$-dense.

A small-size ϵ-net for the reference set P can be built efficiently. (We leave the proofs of the following lemmas to the end of the paper.)

▶ **Lemma 10.** *For any point multi-set $P \subset \mathbb{R}^d$ with size mn and any constant $c > 0$, a $1/cn$-net of P with size $O_d(n)$ can be generated in $O_d(nm \log nm)$ time.*

▶ **Lemma 11.** *A $1/cn$-net of the reference set P with size $O_d(n \log n)$ can be generated in $O_d(n \log n)$ time for any constant $c > 0$ with probability at leat $1 - 1/n$.*

5 Handling Close Regions

In last section, we discuss how to use (ξ, μ)-dense Q to obtain a space discretization and how to select translations from small regions. In this section, we show how to pick translations from close regions (*i.e.* regions that satisfy condition 1 in Lemma 6) so that they will be good approximations for all those translations that fall in close regions and discover some instances of B.

Below, we assume that γ, β, μ and ξ are chosen such that $\gamma < (16\sqrt{d}+1)^{-1}$, $4\sqrt{d}\beta < 1$ and $\xi = 12d$. Let $B' \subseteq B$ be an instance of A and $\mathcal{T}_O$ be the translation that discovers B' and lies in some close region C. Denote by $V \subset Q$ and $v \in V$, respectively, the subset of points in Q and its representative associated with C, as indicated in condition 1 of Lemma 6. Let ϕ_O be the matching between $\mathcal{T}_O(A)$ and B' that realizes the minimum bipartite matching cost. Let G_v be the smallest axis-aligned box containing V. We consider 2 cases: $|G_v \cap P(\phi_O)| \geq 2m/3$ or $|G_v \cap P(\phi_O)| < 2m/3$.

▶ **Lemma 12.** *If $|G_v \cap P(\phi_O)| \geq 2m/3$, $\mathcal{T}_O = v$.*

Proof. Assume by contradiction that $\mathcal{T}_O \neq v$. For any matching ϕ and any translation $\mathcal{T}$, we use notation $C_\phi(\mathcal{T})$ to denote the matching cost between $\mathcal{T}(A)$ and B' under ϕ. In the following, we analyze how the value of $C_{\phi_O}(\mathcal{T})$ changes, where variable $\mathcal{T}$ is initially $\mathcal{T}_O$, and then changed to v. Note that $C_{\phi_O}(\mathcal{T}) = \sum_{p \in P(\phi_O)} \|p - \mathcal{T}\| = \sum_{p \in G_v \cap P(\phi_O)} \|p - \mathcal{T}\| + \sum_{p \in P(\phi_O) \setminus G_v} \|p - \mathcal{T}\|$.

To estimate the change of $\sum_{p \in G_v \cap P(\phi_O)} \|p - \mathcal{T}\|$, we note that for any $p \in G_v \cap P(\phi_O)$, $\|p - v\| \leq \sqrt{d} D(V) \leq \sqrt{d}\beta \|\mathcal{T}_O - v\|$, where the last inequality is from Condition 1 of Lemma 6 and the fact that $\mathcal{T}_O$ is in C. From triangle inequality, we have $\|p - \mathcal{T}_0\| \geq \|v - \mathcal{T}_0\| - \|v - p\| \geq (1 - \sqrt{d}\beta)\|v - \mathcal{T}_0\|$. Thus, we get $\|p - \mathcal{T}_0\| - \|p - v\| \geq (1 - 2\sqrt{d}\beta)\|v - \mathcal{T}_0\|$. This means that moving $\mathcal{T}$ from $\mathcal{T}_O$ to v reduces the value of $\sum_{p \in G_v \cap P(\phi_O)} \|p - \mathcal{T}\|$ by at least $(2m/3)(1 - 2\sqrt{d}\beta)\|v - \mathcal{T}_O\|$.

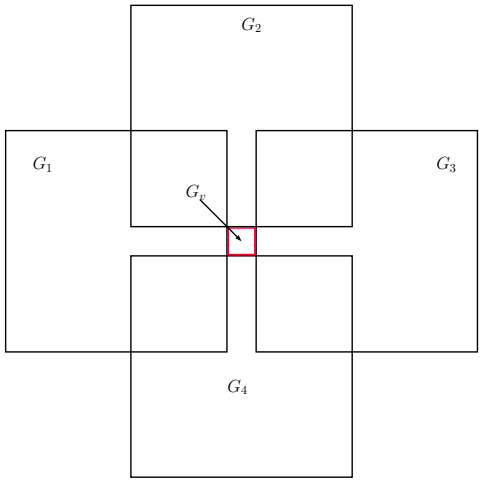

Figure 2 Illustration of the arrangement of $\{G_1, G_2, \ldots, G_{2d}\}$ and G_v.

For the term of $\sum_{p\in P(\phi_O)\setminus G_v}\|p - \mathcal{T}\|$, we know (from triangle inequality and the assumption that $|P(\phi_O) \setminus G_v| \leq m/3$) that its change is smaller than $(m/3)\|v - \mathcal{T}_O\|$. Since $1 - 2\sqrt{d}\beta > 1/2$ (by the assumption that $4\sqrt{d}\beta < 1$), we get $(2m/3)(1 - 2\sqrt{d}\beta)\|v - \mathcal{T}_O\| > (m/3)\|v - \mathcal{T}_O\|$ when $\mathcal{T}_O \neq v$. Therefore, we have $C_{\phi_O}(v) < C_{\phi_O}(\mathcal{T}_O)$ (from previous discussion). However, this results in a contradiction, since from definition, $\mathcal{T}_O$ discovers B' and thus should have the minimum matching cost between A and B' under any translation.

Thus, the lemma follows. ◀

For the case $|G_v \cap P(\phi_O)| < 2m/3$, we have the following lemma.

▶ **Lemma 13.** *If $|G_v \cap P(\phi_O)| < 2m/3$, then for any $\mathcal{T}' \in C$, $\Delta(\mathcal{T}'(A), B') \leq (1 + 48\beta\sqrt{d})\Delta(\mathcal{T}_O(A), B')$.*

Proof. For any matching ϕ and any translation $\mathcal{T}$, we use notation $C_\phi(\mathcal{T})$ to denote the matching cost between $\mathcal{T}(A)$ and B' under ϕ. We prove this lemma by showing that $C_{\phi_O}(\mathcal{T}') \leq (1 + 48\beta\sqrt{d})C_{\phi_O}(\mathcal{T}_O)$; the lemma then follows, since $C_{\phi_O}(\mathcal{T}') \geq \Delta(\mathcal{T}'(A), B')$.

Let P' be the closest $5m/6$ points to $\mathcal{T}_O$ in $P(\phi_O)$. Since $|G_v \cap P(\phi_O)| < 2m/3$, we have $|P' \setminus G_v| \geq m/6$.

For analysis purpose, imagine that we "attach" $2d$ axis-aligned boxes $\{G_1, G_2, \ldots, G_{2d}\}$ to each face of G_v, with centers aligned in G_v (see Figure 2 for an example in 2D), and each box has equal edge length r, where r is the smallest positive number such that P' is contained in the union of $\{G_1, G_2, \ldots, G_{2d}\}$ and G_v. Let $F = G_v \cup G_1 \cup G_2 \ldots \cup G_{2d}$.

By the fact that $|P' \setminus G_v| \geq m/6$, we know that one box of $\{G_1, G_2, \ldots, G_{2d}\}$ contains more than $m/12d$ points in P. Since $\xi = 12d$ and Q is a $1/n\xi$-net of P, we also know that the box contains a point q_t from Q. Thus, F contains a point q_t in $Q \setminus V$.

From Lemma 6, we have $\|\mathcal{T}_O - v\| \leq \gamma\|\mathcal{T}_O - q_t\|$. Thus,

$$\|v - q_t\| \geq (1/\gamma - 1)\|\mathcal{T}_O - v\|. \tag{2}$$

Let L_v denote the edge length of G_v. Then, we have $L_v \leq D(V) \leq \beta\|\mathcal{T}_O - v\|$. Since $\beta < 1/4\sqrt{d} < (1/4\sqrt{d})(1/\gamma - 1)$, from (2) we get

$$L_v \leq \|v - q_t\|/4\sqrt{d}. \tag{3}$$

Let L denote the length of F: $L = 2r + L_v$. L is clearly no smaller than $\|v - q_t\|/\sqrt{d}$ in order to contain both v and q_t. From (3), we have $L_v \leq r$ and $r \geq 3\|v - q_t\|/8\sqrt{d}$. By (2) and the assumption that $1/\gamma - 1 \geq 16\sqrt{d}$, we get

$$r \geq 3(1/\gamma - 1)\|\mathcal{T}_O - v\|/8\sqrt{d} \geq 6\|\mathcal{T}_O - v\|. \tag{4}$$

Let O_v denote the center of G_v. Then, $\|O_v - v\| \leq \sqrt{d}L_v \leq \sqrt{d}D(V) \leq \sqrt{d}\beta\|\mathcal{T}_O - v\| \leq \|\mathcal{T}_O - v\|/4$. Thus, from triangle inequality we have

$$\|O_v - \mathcal{T}_O\| \leq 5\|\mathcal{T}_O - v\|/4. \tag{5}$$

Combining (4) and (5) gives us $\|O_v - \mathcal{T}_O\| \leq 5r/24$.

From the definition of r, we know that there must exist a point $p' \in P'$ on the boundary of F. From the arrangement of $\{G_1, G_2, \ldots, G_{2d}\}$ and G_v, we have $\|O_v - p'\| \geq r/2$. Thus, $\|\mathcal{T}_O - p'\| \geq r/2 - 5r/24 > r/4$ (by triangle inequality).

Also, from the fact that $L_v \leq r$, we know that F can be covered by a box centered at O_v and with edge length $3r$. Since $q_t \in F$, we have $\|O_v - q_t\| \leq 3\sqrt{d}r/2$. Combining this with the fact that $\|O_v - \mathcal{T}_O\| \leq 5r/24$, we get $\|\mathcal{T}_O - q_t\| \leq 3\sqrt{d}r/2 + 5r/24 \leq 2\sqrt{d}r$. Thus, we have $\|\mathcal{T}_O - q_t\|/\|\mathcal{T}_O - p'\| \leq 8\sqrt{d}$.

In summary, from the above discussion and Lemma 6, we have the following.

1. There exists $q_t \in Q$ such that the following inequality holds $\|\mathcal{T}_O - q_t\|/\|\mathcal{T}_O - p'\| \leq 8\sqrt{d}$, where $p' \in P(\phi_O)$ satisfies the condition that less than $5m/6$ points in $P(\phi_O)$ are closer to $\mathcal{T}_O$ than it.
2. Inequality $\|\mathcal{T}_O - \mathcal{T}'\| \leq \beta\|\mathcal{T}_O - q_t\|$ holds for any $\mathcal{T}' \in C$.

Following a similar argument given in Lemma 5, we can show that $|C_{\phi_O}(\mathcal{T}_O) - C_{\phi_O}(\mathcal{T}')| \leq 48\beta\sqrt{d}C_{\phi_O}(\mathcal{T}_O)$. This concludes the proof. ◀

Now for any $\epsilon > 0$, if we set $48\beta\sqrt{d} \leq \epsilon$, then by Lemmas 12 and 13, we know that if $\mathcal{T}_O$ lies in a close region C generated by AIDecomposition, either v discovers B' (if $|G_v \cap P(\phi_O)| \geq 2m/3$), or an arbitrary $\mathcal{T}' \in C$ $(1+\epsilon)$-approximately discovers B' (if $|G_v \cap P(\phi_O)| < m/3$). Therefore, we may put v and an arbitrary point in C into the candidate set. This ensures that B' is $(1+\epsilon)$-approximately discoverable by at least one of these two points when $\mathcal{T}_O$ lies in C.

6 The Algorithm and Analysis

In this section, we summarize the discussion so far and provide the algorithm to generate the candidate set of translations. The following Algorithm 1 shows in details how to generate the candidate set $\mathbb{T}$.

Depending on the method chosen to construct the ϵ-net in Step 3 (Lemma 10 or Lemma 11), the size of Q is $O(n)$ or $O(n \log n)$, the size of the discretization (in terms of number of regions) generated in step 4 is $O(n \log n)$ or $O(n \log^2 n)$, and the total running time of the algorithm is $O(mn \log mn)$ or $O(n \log^2 n)$. No matter which algorithm is chosen to construct $\mathbb{T}$, we have the following lemma.

▶ **Lemma 14.** *For any instance $B' \in B$ of A, there exists at least one translation $\mathcal{T} \in \mathbb{T}$, such that $\mathcal{T}$ $(1+\epsilon)$-approximately discovers B'.*

Algorithm 1 Generate-Candidate-Set.

Input: Point sets A and B of $\mathbb{R}^d$ with $|A| \leq |B|$. Approximate factor $0 < \epsilon < 1$.
Output: A set $\mathbb{T}$ of translations in $\mathbb{R}^d$, such that each instance B' of A is $(1+\epsilon)$-approximately discoverable.

1: Initialize $\mathbb{T}$ to be $\emptyset$.
2: Initialize constants β, γ, ξ, such that: $\xi = 12d$, $48\beta\sqrt{d} \leq \epsilon$, $\gamma < (16\sqrt{d}+1)^{-1}$, $4\sqrt{d}\beta < 1$ and $\sqrt{d}\beta(1-1/\xi)^{-1} \leq \epsilon$.
3: Build a $1/\xi n$-net Q for P, where P is the reference set.
4: Run AIDecomposition(Q, β, γ) to generate a discretization which decomposes $\mathbb{R}^d$ into close regions and small regions.
5: For each small region C, pick an arbitrary point from C and put it into $\mathbb{T}$.
6: For each close region C, suppose it is associated with point set V and $v \in V$. Pick an arbitrary point p in C. Put both v and p into $\mathbb{T}$.
7: Output $\mathbb{T}$ as the result.

Proof. Let $\mathcal{T}_O$ be the translation that discovers B'.

If $\mathcal{T}_O$ lies in a small region C, let $\mathcal{T}_C \in \mathbb{T}$ be the point chosen from C in step 5 of Algorithm 1. Let q be the nearest neighbor of $\mathcal{T}_O$ in Q. By Lemma 6, we know that $\|\mathcal{T}_C - \mathcal{T}_O\| \leq \beta\|q - \mathcal{T}_O\|$. By Lemma 8 and the fact that Q is $(\xi, \sqrt{d})$-dense, we have $\Delta(\mathcal{T}_C(A), B') \leq (1+\sqrt{d}\beta(1-\xi)^{-1})\Delta(\mathcal{T}_O(A), B') \leq (1+\epsilon)\Delta(\mathcal{T}_O(A), B')$ (The last inequality comes from choice of parameters in Algorithm 1). Thus, we know that B' is $(1+\epsilon)$-approximately discoverable at $\mathcal{T}_C \in \mathbb{T}$.

If $\mathcal{T}_O$ lies in a close region C, let $v \in V \subset Q$ be the representative point associated with C as stated in Lemma 6. Let $\mathcal{T}_C \in \mathbb{T}$ be the point chosen from C in step 6 of Algorithm 1. Then, by Lemmas 12 and 13, we know that either v discovers B', or $\Delta(\mathcal{T}_C(A), B') \leq (1+48\beta\sqrt{d})\Delta(\mathcal{T}_O(A), B') \leq (1+\epsilon)\Delta(\mathcal{T}_O(A), B')$. This means that either v or $\mathcal{T}_C$ $(1+\epsilon)$-approximately discovers B'.

This completes the proof. ◀

From the above analysis, we immediately have our main results, Theorems 2 and 3.

7 Constructing ϵ-net for Hypercubes for P

To conclude this paper, we introduce efficient algorithms to construct a weak ϵ-net for the reference set P with axis-aligned hypercubic ranges. From the well known ϵ-net theorem, we know that a random sample of size $O((d'/\epsilon)\log(d'/\epsilon) + \log n/\epsilon)$ from P, where d' is the VC-dimension of the range space defined by hypercubes in $\mathbb{R}^d$ (it is known that $d' \leq 2d$), is an ϵ-net with probability at least $1 - 1/n$. There are several previous results on ϵ-net for simple shapes like axis aligned rectangles, halfspace and disks in 2 or 3 dimension [20, 8, 2], which provide methods to build smaller size ϵ-nets. [19] provides a mathematical construction of ϵ-nets for axis-aligned hypercubes of size $O(1/\epsilon)$, which is optimal in size, although its efficient (*i.e.* in near $O(|P|)$ time) algorithmic implementation is unknown. [12] introduces a method to construct ϵ-nets for axis-aligned rectangles in any fixed dimension, which can be applied to generate the (ξ, μ)-dense subset Q. The running time of this method is $O(|P|\log^d|P|)$.

We show that if we further restrict the shapes to hypercubes, we are able to obtain an optimal size (*i.e.* $O(1/\epsilon)$) weak ϵ-net more efficiently. In the following we show how to deterministically construct a linear size $1/n$-net Q for any multi-set P of size $O(mn)$, *i.e.*, a point set Q of size $O(n)$ such that if an axis-aligned hypercube G contains m points in

P, it then contains at least one point in Q. The time of the construction is $O(nm \log nm)$ (Recall that the size of P is mn). Clearly, the same space and time complexity bounds for a $1/cn$-net Q for any constant c are also achievable, thus proving Lemma 10. We also note that by applying the ϵ-net theorem, it is possible to construct such a Q of larger size ($O(n \log n)$), but in shorter ($O(n \log n)$) time, with high probability. This allows us to make a trade-off between the size of Q and the time complexity of the construction. We leave the discussion of the alternative construction to the end of section, and focus on the deterministic linear size ϵ-net construction in the following.

The construction is based on the quad-tree decomposition technique, which recursively partitions the regions inside the quad-tree boxes, and uses a 2^d-way tree structure to represent the partition. To build a quad-tree for P, we first start with a bounding box G which contains all points in P and is the root of the quad-tree. We then decompose G into 2^d smaller boxes with equal size, with each of them being a child of G. For each child box, we recursively perform the same decomposition. The recursion stops when a box contains no more than 1 point in P. The quad tree decomposition for P can be performed within $O(|P| \log |P|)$ time by maintaining a sorted list of P for each of the d axes [6], and compressing the tree properly to handle empty boxes during the decomposition.

Figure 3 below shows and example of quad-tree decomposition.

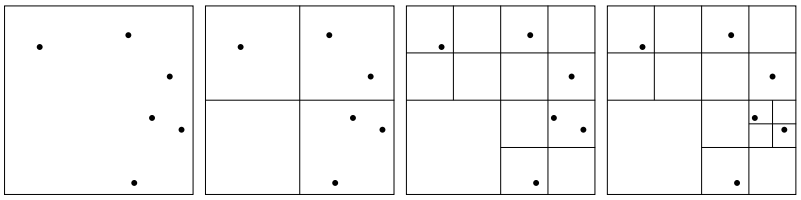

Figure 3 Example of quad-tree decomposition.

A quad-tree decomposition may produce a large number of empty boxes, when a large number of points are aggregated in some region (see left of Figure 4 for an example). To resolve this issue, when decomposing a box G in the quad-tree decomposition, we first perform a quad-tree compression, which directly computes the smallest quad-tree box G_C that contains all the points in $P \cap G$ (see the right side of Figure 4). Then the quad-tree decomposition can continue on G_C. This will avoid generating many unnecessary empty boxes. Note that this compression step is not required if points are not concentrated, *i.e.*, if decomposing G yields at least 2 nonempty boxes (i.e., containing points in P). In this case, we decompose G in the standard fashion.

With this compression step, the running time of the quad-tree decomposition is still $O(|P| \log |P|)$ [6].

Algorithm 2 and Algorithm 3 describe our quad-tree decomposition-based method for producing a weak ϵ-net Q. The decomposition scheme is a modification of the standard quad-tree decomposition. The main routine Algorithm 2 outputs the ϵ-net Q, together with a set U of boxes which is for analysis purpose. Algorithm 3 is the body for the recursion.

The Algorithm 2 and 3 are essentially trimmed versions of the standard quad-tree decomposition (by not decomposing some of the boxes in the process). Given a box G that contains multiple points in P, instead of simply decomposing it into 2^d sub-boxes and recursively building the quad-tree on them (which could generate boxes with few points in it and thus results in a quad-tree with high complexity), Algorithm 3 iteratively performs

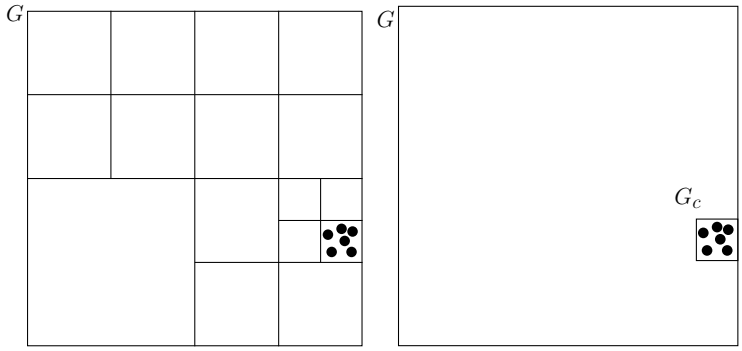

Figure 4 Example of quad-tree compression. It is possible that points in a box are aggregated at some location. Directly applying the quad-tree decomposition will generate many empty boxes. We can directly compute a box to contain all these points without really performing the decomposition.

Algorithm 2 Construct-ϵ-Net.

Input: A set $P \subset \mathbb{R}^d$
Output: An ϵ-net Q. A set U of $\mathbb{R}^d$ boxes.

1: Initialize Q,U as empty sets. Initialize G as a box that contains P.
2: Start recursion by running Decompose-Single-Box subroutine on G

the quad-tree decomposition on *only one* sub-box which contains the maximum number of points in P, and tries to identify a box G' with the following properties. When the iteration (from step 4 to step 8) stops (at step 4 or 7), G' satisfies the following 2 conditions

1. There are less than $m/2^{d+1}$ points in $P \cap G \setminus G'$.
2. (a) All points in $P \cap G'$ have the same location, OR (b) There are at least $m/2^{d+1}$ points in $P \cap G \setminus G''$, where G'' is the child box of G' with the most number of points in P.

Only in case 2(b) we perform the recursion on the boxes generated by the decomposition of G'. See Figure 5 for illustration. In addition, we do not decompose G when there are only a small number ($\leq m/2^{d+1}$) of points in it. Since the algorithm is a trimmed version of the standard quad-tree decomposition, the running time is thus $O(|P| \log |P|) = O(mn \log mn)$.

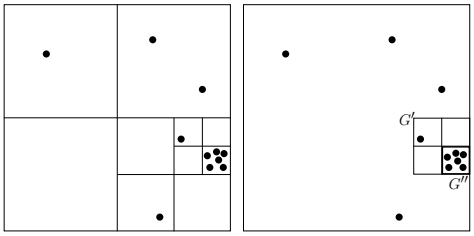

Figure 5 To achieve better performance, Algorithm 3 uses an iteration to find out G' with the desired properties. Recursion continues only (on sub-boxes of G') if $P \cap G \setminus G''$ contains an enough number of points, where G'' is the quad-tree child box of G' with the most number of points in P. This can greatly reduce the number of boxes generated.

It is quite clear that the size of Q is $O(n)$. The Decompose-Single-Box procedure stops immediately once the condition $|G \cap P| \leq m/2^{d+1}$ is satisfied. The procedure also makes sure that for any G'' of the 2^d child boxes generated for G (if the decomposition and recursion occur), inequality $|G \setminus G''| \geq m/2^{d+1}$ holds. This implies that the size of the recursion tree, and thus the size of U and Q, is $O(mn/(m/2^{d+1})) = O(n)$.

Algorithm 3 Decompose-Single-Box.

Input: A box G

Output: A sub-quad tree with G as the root.

1: Add G into U. Add all the vertices of G to Q.
2: If G contains $\leq m/2^{d+1}$ points in P, return.
3: Initialize variable G' to be the box G.
4: If all points in P that lie in G' coincide at point p. Put p and all vertices of G' into Q. Put G' into U. Return.
5: Update variable G' to be the resulting box from the quad-tree compression of the current G', if necessary (See Appendix A.3).
6: Decompose G' equally into 2^d sub-boxes. Let G'' be the one that contains the most number of points in P.
7: If $G \setminus G''$ contains $\geq m/2^{d+1}$ points in P, recursively call Decompose-Single-Box on the 2^d sub-boxes generated in the above step. Then return.
8: Otherwise, update variable G' to be G''. Go to step 4.

▶ **Lemma 15.** *Set Q generated by the above algorithm is a weak $1/n$-net for P,* i.e., *if any axis-aligned hypercube G_m contains at least m points in P, then G_m contains at least 1 point in Q.*

Proof. Let U_m denote the subset $U_m \subseteq U$ of hyperboxes G' in U such that the interior of G_m intersects G'.

For each coordinate axis e of $\mathbb{R}$, let $F(e)$ be the set of faces f of boxes in U_m, such that f is perpendicular to e and intersects the interior of G_m. Let the *cutting number* $x(e)$ of e be the possible number of distinct coordinate values of faces in $F(e)$ in the e axis.(See Figure 6 for an example.) We consider two cases.

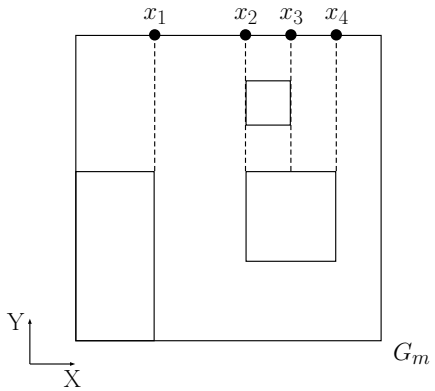

Figure 6 Example of cutting number for X axis in an configuration of interior G_m. In this example there are 3 boxes of U_m intersecting G_m. There are 4 different x values for faces of these boxes that lies in G_m and are perpendicular to the X axis. Thus the cutting number for X axis in this example is 4.

Case 1. $x(e) \leq 1$ for any axis e. Then boxes in U partitions G_m into no more than 2^d regions, since in every direction G_m is "cut" by boxes in U at most once. Since G_m contains at least m points in P, one of the regions will contain strictly more than $m/2^d$ points. However, from Algorithm 3, we know that all of the regions formed by U can

only have no more than $m/2^{d+1}$ points, with the exception that for some regions, all its contained points in P have the same location p. Thus, G_m intersects such a region and contains p. Since $p \in Q$ from Algorithm 3 (see Step 4), this case is proved.

Case 2. $x(e) \geq 2$ **for some axis** e**.** Since the quad-tree decomposition always divides boxes equally, the following fact is clear.

Fact. *Let f_1, f_2 be the faces of boxes G_1 and G_2 in U, respectively, such that they are facing the same direction e. If the distance between f_1 and f_2 in the direction of e is $l > 0$, one of G_1 and G_2 has edge length $\leq l$.*

If $x(e) \geq 2$, then there exist faces f_1 and f_2 facing the direction of e and intersect the interior of G_m. If these 2 faces belongs to the same box $G_f \in U$, then G_f is smaller than G_m in size. Thus, one of the vertices p_f of G_f must lie in G_m. From Algorithm 3, we know that $p_f \in Q$. Therefore, $G_m \cap Q \neq \emptyset$. If f_1 and f_2 belong to different boxes, say G_1 and G_2 in U, from the above fact, we know that the edge length of one of the boxes will be no larger than the distance between f_1 and f_2 in the direction of e. Since the box size is smaller than G_m, one of its vertices is in G_m. This again leads to the fact that $G_m \cap Q \neq \emptyset$. This completes the proof. ◀

Alternative Construction Using ϵ-net Theorem. Recall that the ϵ-net Theorem allows us to build a $1/cn$-net of P with size $O(n \log n)$ by using $O(n \log n)$ samples, where $c > 0$ is a constant. Also note that, it is not necessary to explicitly compute P (whose size is mn) before conducting the sampling. From the definition of P (in Section 2, Definition 1), a random sample from P can be obtained by first sampling a from A, b from B, and then computing $b - a$ as the sample. This allows us to build a $1/cn$-net of P in $O(n \log n)$ time.

References

1. Helmut Alt and Leonidas J Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In *Handbook of computational geometry*, pages 121–153. Elsevier, 2000.
2. Boris Aronov, Esther Ezra, and Micha Sharir. Small-Size \eps-Nets for Axis-Parallel Rectangles and Boxes. *SIAM Journal on Computing*, 39(7):3248–3282, 2010.
3. Sunil Arya, Theocharis Malamatos, and David M Mount. Space-time tradeoffs for approximate nearest neighbor searching. *Journal of the ACM (JACM)*, 57(1):1, 2009.
4. Rinat Ben-Avraham, Matthias Henze, Rafel Jaume, Balázs Keszegh, Orit E Raz, Micha Sharir, and Igor Tubis. Minimum partial-matching and Hausdorff RMS-distance under translation: combinatorics and algorithms. In *European Symposium on Algorithms*, pages 100–111. Springer, 2014.
5. Sergio Cabello, Panos Giannopoulos, and Christian Knauer. On the parameterized complexity of d-dimensional point set pattern matching. In *International Workshop on Parameterized and Exact Computation*, pages 175–183. Springer, 2006.
6. Paul B Callahan and S Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, 1995.
7. Danny Z Chen, Ziyun Huang, Yangwei Liu, and Jinhui Xu. On Clustering Induced Voronoi Diagrams. *SIAM Journal on Computing*, 46(6):1679–1711, 2017.
8. Kenneth L Clarkson and Kasturi Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58, 2007.
9. Hu Ding, Ronald Berezney, and Jinhui Xu. k-prototype learning for 3d rigid structures. In *Advances in Neural Information Processing Systems*, pages 2589–2597, 2013.
10. Hu Ding, Branislav Stojkovic, Ronald Berezney, and Jinhui Xu. Gauging association patterns of chromosome territories via chromatic median. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1296–1303, 2013.

11 Hu Ding and Jinhui Xu. FPTAS for minimizing earth mover's distance under rigid transformations. In *European Symposium on Algorithms*, pages 397–408. Springer, 2013.

12 Esther Ezra. A note about weak ϵ-nets for axis-parallel boxes in d-space. *Information Processing Letters*, 110(18-19):835–840, 2010.

13 Martin Gavrilov, Piotr Indyk, Rajeev Motwani, and Suresh Venkatasubramanian. Combinatorial and experimental methods for approximate point pattern matching. *Algorithmica*, 38(1):59–90, 2004.

14 Michael T Goodrich, Joseph SB Mitchell, and Mark W Orletsky. Practical methods for approximate geometric pattern matching under rigid motions:(preliminary version). In *Proceedings of the tenth annual symposium on Computational geometry*, pages 103–112. ACM, 1994.

15 Sariel Har-Peled. A replacement for Voronoi diagrams of near linear size. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 94–103. IEEE, 2001.

16 David Haussler and Emo Welzl. ϵ-nets and simplex range queries. *Discrete & Computational Geometry*, 2(2):127–151, 1987.

17 Matthias Henze, Rafel Jaume, and Balázs Keszegh. On the complexity of the partial least-squares matching Voronoi diagram. In *Proc. 29th European Workshop on Computational Geometry*, pages 193–196, 2013.

18 Daniel P Huttenlocher, Klara Kedem, and Micha Sharir. The upper envelope of Voronoi surfaces and its applications. *Discrete & Computational Geometry*, 9(3):267–291, 1993.

19 Janardhan Kulkarni and Sathish Govindarajan. New ϵ-net constructions. In *Proceedings of the 22nd Annual Canadian Conference on Computational Geometry, Winnipeg, Manitoba, Canada*, pages 159–162. Citeseer, 2010.

20 Jiří Matoušek, Raimund Seidel, and Emo Welzl. How to net a lot with little: Small ε-nets for disks and halfspaces. In *Proceedings of the sixth annual symposium on Computational geometry*, pages 16–22. ACM, 1990.

21 Günter Rote. Partial least-squares point matching under translations. In *Proc. 26th European Workshop on Computational Geometry*, pages 249–251. Citeseer, 2010.

22 Nitasha Sehgal, Andrew J Fritz, Jaromira Vecerova, Hu Ding, Zihe Chen, Branislav Stojkovic, Sambit Bhattacharya, Jinhui Xu, and Ronald Berezney. Large-scale probabilistic 3D organization of human chromosome territories. *Human molecular genetics*, 25(3):419–436, 2015.

A Appendix

A.1 Proof of Corollary 4

Proof. Suppose that translation $\mathcal{T}$ and $B' \subseteq B$ realize the minimum cost bipartite matching with A. Let C_{OPT} be the minimum bipartite matching cost of B' and $\mathcal{T}(A)$. C_{OPT} is then the optimal minimum bipartite matching cost between A and B under translations. Since B' is a C_{OPT}-instance of A, there exists $\mathcal{T}' \in \mathbb{T}$ such that the matching cost between B' and $\mathcal{T}'(A)$ is no larger than $(1+\epsilon)C_{OPT}$. Thus, $\mathcal{T}'$ induces a $(1+\epsilon)$-approximation for the optimal translational matching between A and B. ◀

A.2 Proof of Lemma 5

Proof. Let ϕ_1 (or ϕ_2) be the corresponding bipartite matching which gives rise to the minimum cost between B' and $\mathcal{T}_1(A)$ (or $\mathcal{T}_2(A)$). Then, $\Delta(\mathcal{T}_1(A), B') = \sum_{q \in P(\phi_1)} \|q - \mathcal{T}_1\|$, and $\Delta(\mathcal{T}_2(A), B') = \sum_{q \in P(\phi_2)} \|q - \mathcal{T}_2\|$. Note that

$$\begin{aligned}
\sum_{q\in P(\phi_1)} \|q-\mathcal{T}_2\| &= \sum_{q\in P(\phi_1)} \|q-\mathcal{T}_1-\mathcal{T}_2+\mathcal{T}_1\| \\
&\le \sum_{q\in P(\phi_1)} \|\mathcal{T}_2-\mathcal{T}_1\| + \sum_{q\in P(\phi_1)} \|q-\mathcal{T}_1\| \\
&= m\|\mathcal{T}_2-\mathcal{T}_1\| + \Delta(\mathcal{T}_1(A), B') \\
&\le m\epsilon\|p-\mathcal{T}_1\| + \Delta(\mathcal{T}_1(A), B') \\
&\le \epsilon\Delta(\mathcal{T}_1(A), B') + \Delta(\mathcal{T}_1(A), B') \\
&= (1+\epsilon)\Delta(\mathcal{T}_1(A), B'),
\end{aligned}$$

where the first inquality comes from the triangle inequality, and the third inequality comes from the fact that p is the nearest neighbor of $\mathcal{T}_1$ in P, which implies that $m\|p-\mathcal{T}_1\| \le \sum_{q\in P(\phi_1)}\|q-\mathcal{T}_1\| = \Delta(\mathcal{T}_1(A), B')$. From the assumption that ϕ_2 is the minimum cost bipartite matching between $\mathcal{T}_2(A)$ and B', we know that the value $\sum_{q\in P(\phi_1)}\|q-\mathcal{T}_2\|$, which is the matching cost between $\mathcal{T}_2(A)$ and B' under ϕ_1, must be no smaller than $\Delta(\mathcal{T}_2(A), B')$. Therefore, from the above inequality, we have $\Delta(\mathcal{T}_2(A), B') \le (1+\epsilon)\Delta(\mathcal{T}_1(A), B')$.

Following a similar argument, we also have

$$\begin{aligned}
\Delta(\mathcal{T}_1(A), B') &\le \sum_{q\in P(\phi_2)} \|q-\mathcal{T}_1\| \\
&= \sum_{q\in P(\phi_2)} \|q-\mathcal{T}_2-\mathcal{T}_1+\mathcal{T}_2\| \\
&\le \sum_{q\in P(\phi_2)} \|\mathcal{T}_2-\mathcal{T}_1\| + \sum_{q\in P(\phi_2)} \|q-\mathcal{T}_2\| \\
&= m\|\mathcal{T}_2-\mathcal{T}_1\| + \Delta(\mathcal{T}_2(A), B') \\
&\le m\epsilon\|p-\mathcal{T}_1\| + \Delta(\mathcal{T}_2(A), B') \\
&\le \frac{\epsilon}{1-\epsilon}\Delta(\mathcal{T}_2(A), B') + \Delta(\mathcal{T}_2(A), B') \\
&= (1-\epsilon)^{-1}\Delta(\mathcal{T}_2(A), B'),
\end{aligned}$$

where the fourth inequality comes from the following argument. The closest distance from a point in P to $\mathcal{T}_1$ is $\|p-\mathcal{T}_1\|$. Since $\|\mathcal{T}_1-\mathcal{T}_2\| \le \epsilon\|p-\mathcal{T}_1\|$, we know that the closest distance from a point in P to $\mathcal{T}_2$ is at least $(1-\epsilon)\|p-\mathcal{T}_1\|$. Therefore, $\Delta(\mathcal{T}_2(A), B') = \sum_{q\in P(\phi_2)}\|q-\mathcal{T}_2\| \ge m(1-\epsilon)\|p-\mathcal{T}_1\|$.

Putting everything together, we have that $(1-\epsilon)\Delta(\mathcal{T}_1(A), B') \le \Delta(\mathcal{T}_2(A), B') \le (1+\epsilon)\Delta(\mathcal{T}_1(A), B')$. Thus, the lemma follows. ◀