Using Genetic Algorithm to Balance the D-Index Algorithm for Metric Search

Tao Ban

Information Security Research Center, National Institute of Information and Communications Technology, 4-2-1 Nukui-Kita Koganei, Tokyo, 184-8795 Japan

Abstract. The Distance Index (D-index) is a recently introduced metric indexing structure which capable of state-of-the-art performance in large scale metric search applications. In this paper we address the problem of how to balance the D-index structure for more efficient similarity search. A group of evaluation functions measuring the balance property of a D-index structure are introduced to guide the construction of the indexing structure. The optimization is formulated in a genetic representation that is effectively solved by a generic genetic algorithm (GA). Compared with the classic D-index, balanced D-index structures show a significant improvement in reduction of distance calculations while maintaining a good input-output (IO) performance.

1 Introduction

Similarity search has become a heated topic of great interest regarding both research and commercial applications. Various applications now use similarity search as either an essential preprocessing step or a kernel algorithm. In this paper, we discuss general similarity search problems where the only information available among objects is pairwise distances measured by some distance function. The data domain together with the similarity measure are generally abstracted as the following metric space model:

Let $\mathbb{D}$ be the data domain, $d : \mathbb{D} \times \mathbb{D} \rightarrow \mathbb{R}$ is a distance on $\mathbb{D}$, the tuple $\mathcal{M} = (\mathbb{D}, d)$ is called a metric space, if for all $x, y, z \in \mathbb{D}$, the following conditions hold.

\[
\begin{align*}
    &d(x, y) \geq 0 & \text{non-negativity} \\
    &d(x, y) = 0 \iff x = y & \text{identity} \\
    &d(x, y) = d(y, x) & \text{symmetry} \\
    &d(x, y) + d(y, z) \geq d(x, z) & \text{triangular inequality}
\end{align*}
\]

Given a metric space, a metric query is generally defined by a query object $q$ and a similarity condition. For brevity, in this paper we only discuss the range query which is known as the most basic query type. A range query is defined by a query object $q \in \mathbb{D}$ and a radius $r \in \mathbb{R}$. The response set of $\mathbb{R}(q, r, X)$ from a finite set $X \subset \mathbb{D}$ is

\[
\mathbb{R}(q, r, X) = \{ x_i | d(q, x_i) \leq r, x_i \in X \}.
\]

Most real world applications can be modeled as metric spaces. The goal when designing a metric search algorithm is to build a data structure for a finite set $X \subset \mathbb{D}$, so that given a query object $q$, the response set can be found efficiently—both in terms
of the cutoff of distance computations as well as the reduction of input-output (IO) operations. Many metric indexing structures now available now; there are, to name a few, the metric tree approaches such as the Vantage Point tree (VPT) [1], Generalized Hyperplane tree (GHT) [2], and Metric tree (MTree) [3], and methods which exploit pre-computed distances such as AESA [4] and LEASA [5].

Similarity hashing methods known as Distance Index (D-index) [6] and its descendants incorporate multiple principles for search efficiency. With a novel clustering technique and the pivot-based distance searching strategy, D-index performs well in terms of reduction of distance calculations and offers a good IO management capability. The main idea of D-index is as follows. At individual levels, objects are hashed into separable buckets which are search-separable up to some predefined value $\rho$. Hence the structure supports easy insertion and a bounded search cost because at most one bucket per level needs to be accessed for queries with $r \leq \rho$. Furthermore, the pivot filtering strategy [4,5] is applied to significantly reduce the number of distance computations in the accessed buckets.

D-index has built a good framework for metric search especially for queries with comparatively small radii. In this paper, we try to further improve its search performance by optimizing the indexing structure. As noted in [6], a more balanced data distribution in the D-index structure improves search performance. Unfortunately, the classic D-index does not support balanced formulation. Our main idea is to use some optimization technique to guide the construction of the D-index structure. This optimization depends on the novel introduction of evaluation functions which measure the balance property of the structure. Another contribution of this paper is that the proposed optimization method allow us to automate the pivot selection procedure of D-index and obtain a well balanced indexing structure without much manual interruption. The D-index performance is further enhanced by sharing pivots among different search levels.

2 Metric Searching by D-index

In the following, we provide an overview of D-index [6].

2.1 Hashing the Dataset

In D-index, the $\rho$-split functions are defined to hash objects into search-separable clusters. An example is the $bps$ (ball-partitioning split) function. With a predefined separability parameter $\rho$, a $bps$ uniquely determines the belongingness of an object $o \in \mathbb{D}$:

$$bps^{1,\rho}(o_i) = \begin{cases} 
0 & \text{if } d(o_i, p) \leq d_m - \rho \\
1 & \text{if } d(o_i, p) > d_m + \rho \\
- & \text{otherwise}
\end{cases}$$

(6)

where $p$ is a pivot and $d_m$ the median of the distances from $p$ to all $o_i \in \mathbb{D}$. The superscript 1 denotes the order of the split function; i.e. the number of pivots involved. The subset characterized by the symbol ‘$-$’ is called the exclusion set, noted as $E$. The subsets noted by $S_{[0]}^{1,\rho}(\mathbb{D})$ and $S_{[1]}^{1,\rho}(\mathbb{D})$ are called separable sets according to the following separable property:

$$d(o_i, o_j) > 2\rho, \quad \text{for all } o_i \in S_{[0]}^{1,\rho}(\mathbb{D}), o_j \in S_{[1]}^{1,\rho}(\mathbb{D}).$$

(7)