Efficient Case Retrieval Using Dropout Similarity Highway Multigraphs

Thomas Gabel

Faculty of Computer Science and Engineering
Frankfurt University of Applied Sciences
60318 Frankfurt am Main, Germany
tgabel@fb2.fra-uas.de

Abstract. The time required to retrieve a query's nearest neighbor may quickly become a CBR system's bottleneck when its case base contains a large volume of cases. Approximate retrieval techniques that build and employ complex index data structures can mitigate this issue by providing an acceptable tradeoff between time complexity and retrieval accuracy. In this paper, we propose a specialized multigraph-based index structure composed of multiple nested sub-case bases with individually labeled edge sets. We develop tailored algorithms for both constructing the index and utilizing it in a more efficient retrieval process, and we also evaluate our approach empirically using established benchmark datasets.

1 Introduction

A key merit of Case-Based Reasoning (CBR) is its ability to allow for "approximate reasoning" in a controlled manner [18]. This statement addresses the conceptually approximate nature of similarity- and utility-based matching. However, approximations may also come into effect on a rather technical level, when acknowledging that the time required for the matchmaking process can easily become a system's bottleneck. If it is possible to avoid an exhaustive similarity-based search over all cases and to perform similarity comparisons for only a tiny fraction of the data, while still reliably identifying approximate nearest neighbor(s), then processing datasets that are orders of magnitude larger becomes feasible. A common approach to achieve that is to create an index data structure from the case data and to exploit it when searching for the best match(es) to a new query. This is exactly the path we follow in the paper at hand. Our main contributions are (a) a structured two-stage construction process of a sophisticated graph-based index data structure using one bottom-up and one top-down pass, and (b) a straightforward adjustability of the complexity and size of the index structure using an exponential complexity decay via a simple parameterization, trading off time versus accuracy. Additionally, our approach ensures (c) linear space demand for the index and logarithmic time scaling during retrieval.

Accuracy and speed of case-based retrieval have been a topic of active research for decades. We devote Section 2 to a review of related work and foundations on approaches to efficient case retrieval. In Section 3, we propose a set

of algorithms that accomplish both, the (a) construction of a multigraph structure whose vertices are selected in a dropout manner and whose edges, speaking metaphorically, resemble a highway system based on similarities, and (b) the efficient utilization of that index structure when performing approximate case-based retrieval. Section 4 presents the results of an empirical evaluation, analyzing our approach's capabilities regarding classification accuracy as well as nearest neighbor retrieval for a variety of established benchmark datasets. Furthermore, we examine its scalability in terms of runtime and memory requirements.

2 Foundations and Related Work

The main challenge of the CBR cycle's first phase, the retrieve phase, is to select a query's nearest neighbor or k nearest neighbors. The traditional approach of linearly iterating over a case base of n cases, calculating each case's similarity to the query, and remembering the k most similar ones, quickly comes to its limits when n grows large. Traditionally, the computational burden of this task has been addressed by two predominant classes of approaches: limiting the amount of cases stored and limiting the number of similarity calculations required.

Case Base Maintenance (CBM [14]), on the one hand, controls the number of cases in the case base, thus fundamentally reducing the overall number of similarity calculations needed. Grounded in the CBR cycle's retain phase, these methods deliberately decide which cases to store, keep, or remove, aiming at a high efficacy and efficiency of the system. Following the initial proposal of the nearest neighbor rule, many authors have proposed ideas to reduce the set of stored cases [1, 20]. Since we will not address CBM in the remainder of this paper, we point to an excellent overview [11] of the history of CBM algorithms. Retrieval below linear time, on the other hand, is key to efficient CBR systems when coping with big data and large case bases. As this is more relevant to the paper at hand, we review that line of research more thoroughly in the following.

2.1 Sublinear Retrieval

Many methods exists that try to limit the number of cases whose similarity to the query must be calculated during a single retrieval. They typically build up a tree- or graph-based *index structure* before the actual retrieval takes place, then utilizing that index to efficiently guide the search towards similar cases.

Accurate Retrieval: Even with sophisticated index structures, the time complexity of nearest neighbor search degrades to linear as the data's intrinsic dimensionality increases [9]. Nevertheless, there are classical approaches that led to, e.g., query times in $O(\log n)$ under certain strong preconditions. In line with this, kd-trees yield near-logarithmic retrieval times, provided that the data dimensionality is low to moderate and the similarity measure satisfies certain conditions [24]. Cover trees [4] arrange the cases with nested trees (an idea that we will build upon, too), require a true metric as similarity measure and have mild time and

memory demands, but a strong dependence on the dimensionality of the data. In recent years, more enhanced tree-based structures and optimized versions of classical ones were proposed, including dual trees and variants specifically tailored for GPU-based acceleration [15]. All of these techniques, however, revert to $O(n)$ query time, if some of their respective preconditions are violated, which is why research in recent years has mostly focused on *approximate* retrieval algorithms.

Approximate Retrieval: Algorithms that settle for finding a query's approximate nearest neighbor may be categorized into one of the following three classes: First, multi-stage retrieval techniques start with a coarse preselection and conclude with a refinement. Among those, locality sensitive hashing approaches (LSH [2]) have gained much popularity in recent years. They start with a hashing procedure for preselection before they perform a more detailed similarity evaluation and, in so doing, share considerable conceptual relatedness with MAC/FAC retrieval approaches traditionally anchored in cognitive modelling and early CBR research [5]. Second, product quantization approaches compress the given data to compact codes of quantized subvector indices [12], enabling an efficient though approximate (due to lossy compression) retrieval. These methods are conceptually more dissimilar from classical CBR since the results they return are not actually stored instances, but rather compression-generated prototypes. The third category adopts discrete index data structures (typically trees or graphs, see Section 2.2 below) which allow to efficiently express neighborhood relationships and, thus, to evaluate the similarity of much fewer cases to the query.

In particular, approximate k-NN algorithms that rely on proximity graphs have gained popularity in recent years, driven by the more and more common need to quickly match queries in large volumes of data or in high-dimensional vector embeddings. The algorithms we are proposing in the paper at hand also fall into this category which is why we explore related work on that topic next.

2.2 Proximity Graphs

Due to the exhaustive volume of research on proximity graphs we provide a coarse overview, only. A comprehensive survey can be found in [23].

Definition 1. *A* Proximity Graph *for a case base $\mathcal{C} = \{c_1, \ldots, c_n\} \subsetneq \mathcal{M}$ is a graph $\mathcal{P} = (V, E)$ with $V = \mathcal{C}$ the set of vertices and E the edges such that: For any similarity measure $sim : \mathcal{M}^2 \to [0, 1]$, any query $q \in \mathcal{M}$ and any $c \in V$ either c is the nearest neighbor of q within $\mathcal{C}$, i.e. $sim(q, c) = \max_{c_i \in \mathcal{C}} sim(q, c_i)$, or there exists an edge $(c, c') \in E$ with $sim(q, c') > sim(q, c)$.*

A proximity graph (PG) enables the use of an efficient greedy retrieval algorithm. Given a query q and a random start case, the search gets iteratively closer to the nearest neighbor by following only adjacent edges from $\mathcal{P}$ that lead to cases that are more similar to q than the current node, stopping when no more increase in similarity can be attained. An example for a PG in multi-dimensional vector spaces is the the Delaunay graph which guarantees precise results under

greedy search, but which is almost fully connected when the data dimensionality is high. Since the construction of PGs according to Definition 1 is costly ($O(n^2)$) and their retrieval time does not necessarily scale well, a whole body of research has focused on relaxing the proximity property from above, thus creating *approximate* proximity graphs (APG) in a much more efficient manner.

Relative neighborhood graphs (RNG [22]), for instance, have connections between two cases only, if there is no third case that is more similar to any of the former. While RNGs cut off many redundant connections, the time required for their full construction is in $O(n^3)$. In k-nearest neighbor graphs (KNNG [17]), by contrast, each case gets connected to its k nearest neighbors which yields a directed graph limiting strictly the number of cases to which each vertex is connected, but which may destroy the global connectivity of strict PGs. Numerous graph-based algorithms for the approximate search of nearest neighbors combine various components of Delaunay graphs, RNGs, and KNNGs to form more complex approaches that are scalable to very large data volumes [6, 16, 7]. The authors of [23] disentangle those developments and provide a thorough overview.

In what follows, we propose Dropout Similarity Highway Multigraphs, a graph-based index structure that yields an APG-like structure with logarithmic complexity scaling during case retrieval. We take inspiration from tree-based nearest neighbor search algorithms [4] and from graph-based ones [16] as well as from classical beam search [19] and more recent techniques like dropout [21].

3 Dropout Similarity Highway Multigraphs

Driving to a distant destination, humans typically take the nearest highway ramp, travel along the system of (interstate) highways, until more regional roads and, finally, tiny drives or alleys are needed to arrive at the target. Our approach for finding a query's (approximate) nearest neighbor is inspired by that analogy. We divide the set of cases into multiple subsets where for each case it is decided whether it gets connected to other (very similar) cases by small roads or, whether it has the privilege of being connected by some (generally longer) "highway segment" to (generally less similar) cases. This way, we form a system of "traffic" connections that we employ when searching for a query's nearest neighbor.

3.1 Outline

We aim at the construction of a labeled multigraph structure whose construction process consists of two parts (cf. Algorithm 1). In a first bottom-up stage, the vertices (so to say, the traffic hubs) of the multigraph are selected randomly in a nested order (line 1). This results in D subsets of the case base, each of which contains the next one. The details on this part are given in Section 3.3.

The second stage (line 2) deals with the edges (so to say, the highways and roads) and, in doing so, follows a top-down approach. Details on this part will be given in Section 3.4. In Section 3.2, we start by discussing a beam search adaptation for APGs that is utilized in the second stage. Algorithm 1 takes four

parameters. Besides the case base $\mathcal{C}$ for which the graph-based index structure is to be created, δ is the defining parameter for the dropout idea, whereas m restricts the maximum degree of graph vertices and Ω influences the beam search algorithm mentioned. We will discuss those parameters in Section 3.5.

CONSTRUCT_DROPOUT_SIMHIGHWAY_MULTIGRAPH($\mathcal{C}$, δ, Ω, m)
Input: case base $\mathcal{C}$, dropout factor $\delta \in [0, 1)$, maximum beam width Ω and maximum vertex degree m
Output: dropout similarity highway multigraph $\tilde{\mathcal{S}}$
1: $\mathcal{D}_{\mathcal{C}}$:= CREATE_CASE_DROPOUT_SEQUENCE($\mathcal{C}$, δ, Ω, m) *//call to Alg. 3*
2: $\tilde{\mathcal{S}}$:= CREATE_DSHM_WITH_EDGES($\mathcal{D}_{\mathcal{C}}$) *//call to Alg. 4*
3: **return** $\tilde{\mathcal{S}}$

Algorithm 1: Full Creation Procedure of a Dropout Similarity Highway

3.2 Beam Search in Approximate Proximity Graphs

As delineated above, greedy search in a *strict* proximity graph always yields the case from the case base with maximal similarity to the query. If the graph G used does not have the proximity property (cf. Definition 1), because G is just an approximation of $\mathcal{P}$ (e.g. a k-nearest neighbor graph or some approximation of that), then parts of the case base may be poorly or not at all connected to other parts, such that the search can get stuck in local optima.

An effective method to mitigate that problem is to apply a beam search approach with a large enough beam width ω [19]. In Algorithm 2, we delineate how beam search can generally be applied for a given query and an APG. Basically, a beam of width ω of the most promising cases is kept, from which all neighboring cases are considered. In so doing, the algorithm remembers which cases have already been visited and tries to finish the search as early as possible, returning the beam of the most similar cases found. For ease of presentation, we assume the similarity measure sim to be symmetric for the remainder of this paper. We also restrict our presentation to graphs with undirected edges, i.e. $(e, f) \in E \iff (f, e) \in E$.

3.3 Bottom-Up Selection of Multigraph Nodes

While the previously presented algorithm is merely a tool to be used later, we now turn to the actual construction of our multigraph index structures, starting with its vertices. The eligibility of cases to become "major hubs" follows a simple procedure that is inspired by the dropout approach known from neural network training [21] and has also some relatedness to the tree-based structures in [4].

Given a case base $\mathcal{C}$ and a dropout (or decay) factor $\delta \in [0, 1)$, a fraction of $\lfloor \delta \cdot |\mathcal{C}| \rfloor$ cases is selected randomly to form a subset of instances of "higher traffic importance". This procedure is applied repeatedly until only one case remains.

BEAM_SEARCH_APG(G, q, ω, S)
Input: approximate proximity graph $G = (\mathcal{C}, E)$ where $\mathcal{C}$ is the case base, query $q \in \mathcal{M}$, beam width ω, search start vertices $S \subset \mathcal{C}$
Output: beam $\mathcal{B}$ of most similar cases to q

```
1:  V := S, B := S, E := S          //visited cases, the beam, cases to be explored
2:  while |E| > 0
3:    e := arg max_{c∈E} sim(q,c), E = E \ {e}          //currently explored case
4:    w := min_{c∈B} sim(q,c)                  //worst case's similarity within beam
5:    if sim(q,e) < w then return B      //everything in B has been evaluated
6:    forall ν ∈ {f ∈ C|(e,f) ∈ E} do              //iterate over proximity of e
7:      if ν ∉ V then
8:        w := min_{c∈B} sim(q,c)            //worst case's similarity within beam
9:        if sim(q,ν) > w or |B| < ω then
10:         E := E ∪ {ν}, B := B ∪ {ν}
11:         if |B| > ω then                    //keep beam width under control
12:           B := B \ {arg min_{c∈B} sim(q,c)}
13:       V := V ∪ {ν}                                        //mark ν as visited
14: return B
```

Algorithm 2: Beam Search Applied to an Approximate Proximity Graph

Definition 2. *For a given dropout factor $\delta \in [0, 1)$ and a case base $\mathcal{C}$, the sequence $\mathcal{L} = (\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_D)$ is called* Sub-Case Base Sequence*, if $\mathcal{C}_0 = \mathcal{C}$ and $\mathcal{C}_{i+1}$ is a random subset of $\mathcal{C}_i$ ($\mathcal{C}_{i+1} \subsetneq \mathcal{C}_i$) with $|\mathcal{C}_{i+1}| = \lfloor \delta \cdot |\mathcal{C}_i| \rfloor$ and $|\mathcal{C}_D| = 1$.*

As far as the traffic system metaphor is concerned, we might say that later entries in a sub-case base sequence correspond to sets of the more major hubs. In the preview picture (Figure 1) those cases are plotted with a slightly larger diameter. Furthermore, any place that has access to a major highway can also have connections to roads of lower importance with rather small range reach. We will exploit that property in the sense that possible connections between the (few) major hubs allow us to get quickly to different (i.e. to dissimilar) parts of the search space. By construction it holds $D = \lceil \log_{\frac{1}{\delta}}(|\mathcal{C}|) \rceil = \lceil -\log_{\delta}(|\mathcal{C}|) \rceil$, so for smaller values of δ the sub-case base sequence is shorter.

We take the dropout idea a step further by attaching two properties to each sub-case base $\mathcal{C}_i$. First, if $c \in \mathcal{C}_i$ is a vertex in the graph, then we might be interested in limiting the number of edges it can have. Given the maximum vertex degree m (parameter to the main Algorithm 1) we define (exponentially decaying) dropout vertex maximum degrees for each sub-case base according to

$$m_i := max(D - i, \lfloor m\delta^i \rfloor). \tag{1}$$

The first entry in the max operator is added merely to ensure increasing connectivity from one sub-case base to the next, whereas the second actually realizes the dropout. Likewise, we restrict the beam width of the beam search within the different sub-case bases in just the same way. Given the maximum beam width Ω (parameter to Algorithm 1), we set sub-case base-specific beam widths to

$$\omega_i := max(1, \lfloor \Omega\delta^i \rfloor). \tag{2}$$

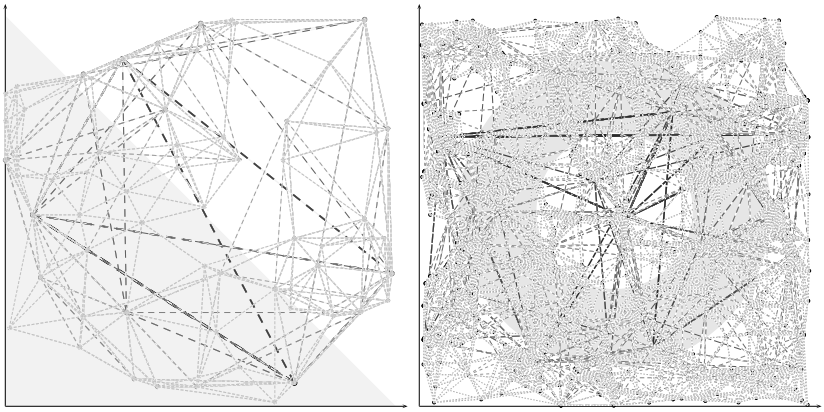

Fig. 1. Preview images of DSHMs in two two-dimensional toy domains with 100 (left) and 10000 (right) cases. Style and thickness of lines identify edge labels. There are 5 different labels in the left DSHM, and 9 in the right one. Each vertex may have up to $m = 10$ edges with the same label. Unlike in [7], both structures encode no information about the solution space (e.g. classes as indicated by the background color).

In fact, the exact values of m_i are defined proportional to the sizes of the corresponding sub-case bases. The same holds for the decaying beam widths ω_i facilitating the use of wider beams when searching through larger sub-case bases.

Definition 3. *Given a sub-case base sequence* $(\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_D)$ *for dropout factor* $\delta \in [0, 1)$*, we define a* Case Dropout Sequence $\mathcal{D}_\mathcal{C} = (C_0, \ldots, C_D)$ *as a vector of three-tuples* $C_i = (\mathcal{C}_i, m_i, \omega_i)$ *whose 2nd and 3rd component are defined according to Equations 1 and 2 for a given max vertex degree* m *and max beam width* Ω*.*

For ease of notation, we use $m_i^\mathcal{D}$ and $\omega_i^\mathcal{D}$ as a shorthand when referring to the maximum vertex degree and beam width of the ith entry of $\mathcal{D}_\mathcal{C}$. The sequence of sub-case bases $\mathcal{C}_i$ in $\mathcal{D}_\mathcal{C}$ will serve as label set $\mathcal{L}$ in the to-be-constructed labeled multigraph. As an example, for $|\mathcal{C}| = 1024$ and $\delta = \frac{1}{4}$ we obtain $|\mathcal{L}| = D = 5$ with, e.g., $|\mathcal{C}_1| = 256$ or $|\mathcal{C}_3| = 16$. Using the illustrative metaphor, there are 1024 cases in $\mathcal{C}_0$ that have connections via an alley, 256 of the cases have connections via lanes, 64 via drives, whereas 16 are connected by larger roads and the four cases within $\mathcal{C}_4$ may become connected with one another using highways.

3.4 Top-Down Construction of Multigraph Edges

Next, we extend PGs by identifiable labeled edges in the graph structure. This means we allow for multiple edges between vertices, but make them distinguishable by attaching a label out of a set $\mathcal{L}$ of D different labels to each edge.

3.4.1 Notation

We define a *Dropout Similarity Highway Multigraph (DSHM)* as a labeled undirected multigraph as follows:

CREATE_CASE_DROPOUT_SEQUENCE($\mathcal{C}$, δ, Ω, m),
Input: case base $\mathcal{C}$, dropout factor $\delta \in [0, 1)$
maximum vertex degree m, maximum beam width Ω
Output: case dropout sequence $\mathcal{D}_\mathcal{C}$

```
1:  D := 0, C_0 := C, D_C := (C_0) with C_0 := (C, m, Ω)      //first element of D_C
2:  while |C_D| > 1                                          //no more dropout possible
3:    C_{D+1} := ∅
4:    while |C_{D+1}| < ⌊δ · |C_D|⌋                          //create next sub-case base
5:      a := random case from C_D \ C_{D+1},  C_{D+1} := C_{D+1} ∪ {a}
6:    D := D + 1
7:  for i := 0 to D do
8:    m_i := max(D − i, ⌊mδ^i⌋), ω_i := max(1, ⌊Ωδ^i⌋)        //use Eq. 1+2
9:    C_i := (C_i, m_i, ω_i)                                 //next entry in D_C
10:   D_C := (C_0, ..., C_{i−1}, C_i)          //append C_i to the existing sequence D_C
11: return D_C
```

Algorithm 3: Case Dropout Sequence Generation

Definition 4. *Let $\mathcal{C}$ be a case base and $\mathcal{D}_\mathcal{C} = (C_0, \ldots, C_D)$ its case dropout sequence with length D. Then, a* Dropout Similarity Highway Multigraph (DSHM) *$\tilde{\mathcal{S}} = (\tilde{V}, \tilde{E}, \tilde{L})$ is defined as a labeled undirected multigraph with $\tilde{V} = \mathcal{C}$ the set of vertices, $\tilde{E}$ a multiset of bidirectional edges, and $\tilde{L} : \tilde{E} \to \mathcal{L}$ a labeling function which maps to each edge a label out of a finite set $\mathcal{L}$ of D unique labels.*

We use the notation $\tilde{\mathcal{S}}|_l$ to indicate the sub-multigraph of $\tilde{\mathcal{S}}$ for some specific edge label $l \in \mathcal{L}$. Note that, due to the restriction to edge label l, $\tilde{\mathcal{S}}|_l$ represents an approximate PG (cf. Definition 1), no longer a DSHM. We formally obtain

$$\tilde{\mathcal{S}}|_l = (V_l, E_l) \text{ where } E_l = \{e \in \tilde{E} | \tilde{L}(e) = l\} \text{ and } V_l = \mathcal{C}_l \tag{3}$$

Metaphorically, we might say that, for instance, the induced sub-multigraph $\tilde{\mathcal{S}}|_l$ of $\tilde{\mathcal{S}}$ for label $l =$ "*interstate highway*" refers to an excerpt of the entire traffic system that covers cities connected to interstate highways, only.

3.4.2 Creation of a DSHM and its Edges

The second stage of creating a full DSHM is delineated in Algorithm 4. This procedure takes as input a case dropout sequence $\mathcal{D}_\mathcal{C} = (C_0, \ldots, C_D)$, which is a vector of triples $C_l = (\mathcal{C}_l, m_l^\mathcal{D}, \omega_l^\mathcal{D})$ whose first elements are sub-case bases of $\mathcal{C}$. As pointed out earlier, the set of edge labels considered (line 1) corresponds to those sub-case bases $\mathcal{L} := \{\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_D\}$. Recall the meaning of the 2nd and 3rd component of C_l: $\omega_l^\mathcal{D}$ indicates the applied beam width when performing beam search using edges with a specific label l on some subgraph $\tilde{\mathcal{S}}|_l$ of the DSHM, and $m_l^\mathcal{D}$ stands for the maximal degrees that vertices are allowed to have w.r.t. a specific edge label l. Note that the degree of a vertex c with respect to some edge label l is defined as $deg_l(c) = |\{(u, v) \in \tilde{E} | \tilde{L}((u, v)) = l, c \in \{u, v\}\}|$.

The algorithm starts by defining a DSHM $\tilde{\mathcal{S}}$ whose vertices are $\mathcal{C}_0$, i.e. the full case base $\mathcal{C}$, and which has no edges. It then (line 3) proceeds "top-down"

CREATE_DSHM_WITH_EDGES($\mathcal{D}_\mathcal{C}$)
Input: case dropout sequence $\mathcal{D}_\mathcal{C} = (C_0, \ldots, C_D)$ with length D
Output: similarity highway multigraph $\tilde{\mathcal{S}}$

1: $\mathcal{L} := \{\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_D\}$ *//sub-case bases form the set of labels*
2: $\tilde{\mathcal{S}} := (\mathcal{C}_0, \tilde{E}, \tilde{L})$ with $\tilde{L} : \tilde{E} \to \mathcal{L}$ and $\tilde{E} = \emptyset$ *//initial DSHM without edges*
3: **for** $l = D$ **downto** 0 *//iterate over labels: from "highways" to "alleys"*
4: **forall** $c \in \mathcal{C}_l$ **do** *//all cases in current "type of road"*
5: $\mathcal{B} := \mathcal{C}_D$ *//start beam search with "main traffic hub"*
6: **for** $r = D$ **downto** l *//find neighbors of c in partially built DSHM*
7: $\mathcal{B} :=$ BEAM_SEARCH_APG(c, $\tilde{\mathcal{S}}|_r$, $\omega^\mathcal{D}_{r-l}$, $\mathcal{B}$) *//call to Alg. 2*
8: **while** $|\mathcal{B}| > 0$ **and** $deg_l(c) < m^\mathcal{D}_l$ *//process all neighbors in beam*
9: $b := \arg\max_{b \in \mathcal{B}} sim(c, b)$ *//best in beam*
10: $\tilde{E} := \tilde{E} \cup \{(c, b)\}$, $\tilde{L}((c, b)) := \mathcal{C}_l$ *//add bidirectional edge + set label*
11: $\mathcal{B} := \mathcal{B} \setminus \{b\}$ *//remove processed neighbor from beam*
12: **if** $deg_l(b) > m^\mathcal{D}_l$ **then** *//max degree of b exceeded*
13: $\tilde{E} := \tilde{E} \setminus \arg\min_{(b,x) \in \tilde{E}, \tilde{L}((b,x)) = \mathcal{C}_l, x \in \mathcal{C}_l} sim(x, c)$
14: **return** $\tilde{\mathcal{S}}$

Algorithm 4: Creation of a Similarity Highway Multigraph's Edges

through the case dropout sequence, starting with C_D and progressing downwards to level C_0. In iteration l the algorithm considers element $C_l = (\mathcal{C}_l, m^\mathcal{D}_l, \omega^\mathcal{D}_l)$ of $\mathcal{D}_\mathcal{C}$. Within each such iteration, each case from $\mathcal{C}_l$ is processed (line 4) which is why a number of cases are touched multiple times. Since $\mathcal{C}_{l+1} \subset \mathcal{C}_l$, each $c \in \mathcal{C}_{l+1}$ is processed once more when iterating over all elements of $\mathcal{C}_l$ (and again for $\mathcal{C}_{l-1}$ etc.). This may appear as a tremendous overhead, but due to the nestedness of the case dropout sequence the amount of work on individual cases is

$$\sum_{l=0}^{D} |\mathcal{C}_l| \underset{|\mathcal{C}_{l+1}| = \delta \cdot |\mathcal{C}_l|}{=} \sum_{l=0}^{D} \delta^l \cdot |\mathcal{C}_0| \underset{\delta \in [0,1)}{\simeq} \frac{|\mathcal{C}|}{1-\delta}$$

and, thus, the overhead can simply be controlled by adjusting δ.

For each $c \in \mathcal{C}_l$ (line 4) two steps are made. First (lines 5-7), a graph-based nearest neighbor beam search for c is made, exploiting the so far created multigraph structure $\tilde{\mathcal{S}}$ in conjunction with the beam search Algorithm 2. Iteratively, a number of beam searches on sub-multigraphs of $\tilde{\mathcal{S}}$ are made, each time utilizing a different edge label $r \in \mathcal{L}$, starting with $r = D$ and proceeding top-down till reaching label l—the label for which the DSHM is currently extended. Each call to BEAM_SEARCH_APG receives the currently considered case c as query, the induced subgraph $\tilde{\mathcal{S}}|_r$ from $\tilde{\mathcal{S}}$ with respect to label r, a beam width $\omega^\mathcal{D}$ increasing for smaller values of r, and, most importantly, the result beam $\mathcal{B}$ of the preceding call as starting nodes for the next call to the beam search. This way, the search for nearest neighbors of c steps through $\tilde{\mathcal{S}}$ in a label-wise manner, finding in each step r the nearest neighbors to c within $\mathcal{C}_r$ and, using those as the seed for the search in the (by factor $\frac{1}{\delta}$) larger sub-case base $\mathcal{C}_{r-1}$.

The remaining lines (8-13) process the results from the previous search and extend $\tilde{\mathcal{S}}$. As long as there are still elements in the result set $\mathcal{B}$ and if the degree

of the current case c w.r.t. label $\mathcal{C}_l$ is below the allowed maximum (which decays w.r.t. the sub-case base size of $\mathcal{C}_l$), the most similar case b from $\mathcal{B}$ is determined (line 9) and an undirected edge between c and b, which is labeled by $\mathcal{C}_l$, is added to the edges of $\tilde{\mathcal{S}}$ (line 10). Then, b is removed from $\mathcal{B}$. Finally, line 12 checks whether the recent extension of $\tilde{E}$ has caused b to have more edges with label $\mathcal{C}_l$ than permitted (the maximum $m_l^{\mathcal{D}}$ is taken from the l-th entry in the case dropout sequence, i.e. from $C_l = (\mathcal{C}_l, m_l^{\mathcal{D}}, \omega_l^{\mathcal{D}})$). If that is the case, then line 13 removes a $\mathcal{C}_l$-labeled edge between b and the case least similar to c.

RETRIEVE_USING_DSHM(q, $\tilde{\mathcal{S}}$, Ω, k)
Input: query q, dropout similarity highway multigraph $\tilde{\mathcal{S}} = (\mathcal{D}_{\mathcal{C}}, \tilde{E}, \tilde{L})$,
maximum beam width Ω, number k of nearest neighbors to retrieve
Output: set of k nearest neighbors
1: $\mathcal{B} := \mathcal{C}_D$ *//where $\mathcal{D}_{\mathcal{C}} = (C_0, \ldots, C_D)$*
2: **for** $l = D$ **downto** 0 *//knowing that length of $\mathcal{D}_{\mathcal{C}}$ is D*
3: $\omega := max(1, \lfloor \Omega \left(\frac{|\mathcal{C}_{l+1}|}{|\mathcal{C}_l|} \right)^l \rfloor)$ *//$|\mathcal{C}_{l+1}|/|\mathcal{C}_l| = \delta$*
4: **if** $l = 0$ **and** $\omega < k$ **then** $\omega := k$ *//assert that at least k nghbs. are found*
5: $\mathcal{B} :=$ BEAM_SEARCH_APG(q, $\tilde{\mathcal{S}}|_l$, ω, $\mathcal{B}$) *//call to Alg. 2*
6: **return** $\arg\max_{B \subset \mathcal{B}, |B|=k} \sum_{b \in B} sim(q, b)$

Algorithm 5: k-NN Retrieval Using a Dropout Similarity Highway Multigraph

3.4.3 Utilizing a DSHM for k-Nearest Neighbor Retrieval

Once, a full DSHM has been constructed, its usage during case-based retrieval for a new query q is straightforward. Algorithm 5 reveals that—very similar to lines 5-7 in Algorithm 4—a label-wise top-down beam search approach is taken, where the results from retrieving within one induced subgraph of the DSHM $\tilde{\mathcal{S}}$ are carried further as the seed to the search in the subgraph for the next label.

The maximum dropout beam width handed over as a parameter Ω may very well differ from the eponymous parameter used during DSHM construction. This way, it is, for example, possible, to do the retrieval on the basis of broader beams and, thus, to trade accuracy for computational efficiency. When reaching label $l = 0$ during search, line 4 additionally asserts that at least k entries will be contained in the result set by setting a lower bound to ω.

3.5 Discussion

Our algorithms are defined by a dropout-inspired divide-and-conquer approach, where the dropout factor δ serves as a key parameter. It determines the length of the case dropout sequence, the exponentially decaying maximum vertex degrees, and the beam widths in the multigraph structure. We consider $\delta = 0.5$ a reasonable default, as it halves the sizes of sub-case bases sequentially. The maximum vertex degree m affects the space requirements of the multigraph (proportional to

m) and must be sufficiently large to ensure connectivity within $\mathcal{C}_0$. Empirically, we found that the maximum beam width Ω during the construction influences graph quality and should not be set below m, making $\Omega = m$ a reasonable choice. The structure of a case dropout sequence implies that most cases are represented multiple times and have edges with varying labels. An efficient implementation represents the sub-case base sequence and all edges between cases using pointers. For a given $\delta \in [0, 1)$ and n cases the space requirements are in $O(\frac{mn}{1-\delta})$. By definition, the length D of the case dropout sequence is $\log_{\frac{1}{\delta}}(n)$. When taking the step from C_{l+1} to C_l the starting case(s) for searching through $\mathcal{C}_l$ are the result of a preceding search in a sparser case base and, thus, a good approximation to the outcome already. The backtracking nature of beam search adds a constant factor to the overall effort which is thus in $O(log(n))$ for the retrieval. As far as the construction of a DSHM is concerned, up to $2n$ retrievals in a partial DSHM must be made, yielding an overall time complexity of $O(n \log(n))$.

4 Experimental Evaluation

The main focus of our experiments is on the accuracy of approximate retrieval, its speed and memory demand, and the corresponding trade-off between them.

We selected the same established classification benchmark datasets as in [7] from the UCI Machine Learning Repository [13] omitting, however, very small datasets with less than 300 instances. We also used a larger nine-dimensional dataset (500k instances) from robotic soccer simulation (RSS) to classify an opponent agent's next action (e.g., kick, dash, or turn, cf. [8]), along with the widely used SIFT 1M dataset, which contains one million 128-dimensional image feature descriptors [10]. While the smaller UCI datasets evaluated DSHM's basic functionality, the million-scale datasets analyzed its scaling properties.

In all classification experiments, we used a knowledge-poor default similarity measure according to the local-global principle [3]. For each domain, we randomly split the data into 80% for constructing the DSHM and 20% for testing. To account for the stochastic nature of our algorithms, we repeated experiments 100 times, except for the largest volumes of data, and averaged performance indicators. The main metrics are classification error and speedup σ, defined as the ratio of total cases to those actually compared during retrieval with a DSHM using Algorithm 5. Thus, σ directly quantifies the speedup over linear search.

The SIFT domain includes a ground truth test set with 10k queries and their 100 true nearest neighbors among the one million cases. To align with the literature, we used a similarity measure that is compatible to the Euclidean distance and focused on 1-recall as performance indicator—the fraction of queries where our approach found the true nearest neighbor.

4.1 Classification Datasets

In our first experiments, we evaluated DSHMs with $\delta = 0.5$ for three different maximum vertex degrees m and beam widths Ω, comparing them to the best-performing linear k-NN retriever (optimal k was found to be $k^* = 13$). We

determined classification error ε on the test set and the speedup σ achieved over sequential search. Table 1 lists the results plus an average over all of them. For reference, the last four columns show error values for related algorithms [7].

Domain	k-NN	DSHM with $k = 13$ and $m = \Omega := p$						BF_t	$IB2_e$	BG_r	LBM_λ
	$k^* = 13$	$p = 15$		$p = 30$		$p = 100$		$t^* = 325$	$e^* = 94$	$r = 50$	$\lambda^* = 37$
	ε	ε	σ	ε	σ	ε	σ	ε	ε	ε	ε
Balance	17.12	16.56	*3.01*	16.80	*1.60*	**16.24**	*1.01*	19.13	45.75	25.18	**15.69**
BanknoteAuth	0.30	1.71	*15.68*	0.25	*9.31*	**0.10**	*2.87*	0.15	0.17	0.14	**0.12**
Car	9.18	10.52	*7.12*	9.50	*3.56*	9.62	*1.31*	**3.93**	14.57	**5.76**	6.59
Contraceptive	52.00	52.92	*8.29*	53.02	*3.94*	**51.49**	*1.43*	52.67	53.90	54.81	**51.11**
Ecoli	16.12	14.17	*3.92*	**14.02**	*2.58*	**13.88**	*1.10*	19.98	23.13	21.99	15.24
Flare	**17.52**	18.26	*11.51*	18.35	*5.87*	18.30	*1.81*	21.21	33.82	21.16	**17.12**
Haberman	**26.20**	**28.19**	*3.91*	**28.19**	*2.24*	**28.19**	*1.06*	35.17	38.12	36.66	33.39
MammogrMass	20.10	20.54	*7.03*	19.27	*4.36*	**18.97**	*1.68*	21.82	27.42	23.64	**19.14**
Monks	6.11	10.23	*2.62*	6.39	*1.50*	5.46	*1.00*	**0.00**	13.03	1.03	**0.22**
Pima	26.58	**25.16**	*5.64*	**24.90**	*3.09*	**24.90**	*1.31*	28.14	34.00	32.47	26.38
TicTacToe	**2.33**	3.56	*5.07*	3.24	*2.41*	2.98	*1.07*	**1.07**	16.02	2.46	4.61
Transfusion	**22.30**	22.34	*9.02*	**22.28**	*5.28*	22.48	*1.86*	29.03	33.51	30.98	24.08
VertebralCol	21.26	20.16	*3.49*	**20.00**	*2.13*	**20.00**	*1.14*	22.53	25.59	26.19	22.37
Wholesale	29.24	**29.20**	*4.60*	**29.20**	*2.77*	**29.20**	*1.25*	42.91	49.19	46.31	35.75
Yeast	**41.85**	42.83	*9.47*	42.43	*5.04*	**42.06**	*1.81*	43.92	51.60	47.98	42.17
Average	20.55	21.09	*6.70*	**20.53**	*3.72*	**20.26**	*1.45*	22.78	30.65	25.12	20.93

Table 1. Classification error ε (in %, averaged over 100 runs) and speedup σ (relative to linear retrieval) for DSHMs with varying maximum vertex degrees m and maximum beam widths Ω, compared to the best linear k-NN ($k^* = 13$) and four retrieval methods from [7]. Top two algorithms per domain are highlighted.

Figure 2 visualizes the impact of a DSHM construction algorithm's main parameters. Its left chart plots ε (averaged over 15 UCI domains) as $m = \Omega$ increases from 5 to 100 ($\delta = 0.5$). There is a strong correlation between the error ε and the speedup σ (dashed line). Notably, for $m = \Omega \geq 28$ the linear k^*-NN retriever ($k^* = 13$) is outperformed while still yielding a speedup of ≈ 4.

The right chart in Figure 2 examines $250k$ RSS cases, highlighting the impact of varying δ. Here, m and Ω were set to 100. For DSHM-based 1-NN retrieval on test cases, classification error plateaus around $\delta = 0.3$. Beyond ~ 0.6, speedup degrades, and memory usage (measured by edge count) increases substantially.

4.2 Scaling Analysis

After analyzing our approach's basic functioning, we now examine its performance on larger, real-world datasets. Case-based classification should improve with more case data. This is confirmed by the solid graphs in Figure 3. In 1-NN classification for the 3-class RSS dataset, DSHM-based retrieval surpasses linear retrieval in accuracy—while achieving a tremendous speedup. Having constructed a DSHM for $\sim 400k$ cases, the classification of a new query q requires only 1295 similarity calculations between q and cases/vertices from $\mathcal{C}$, which is a speedup of ~ 314. Although retrieval time depends on hardware, we clearly observe logarithmic scaling with case data, leading to a speedup growing linearly

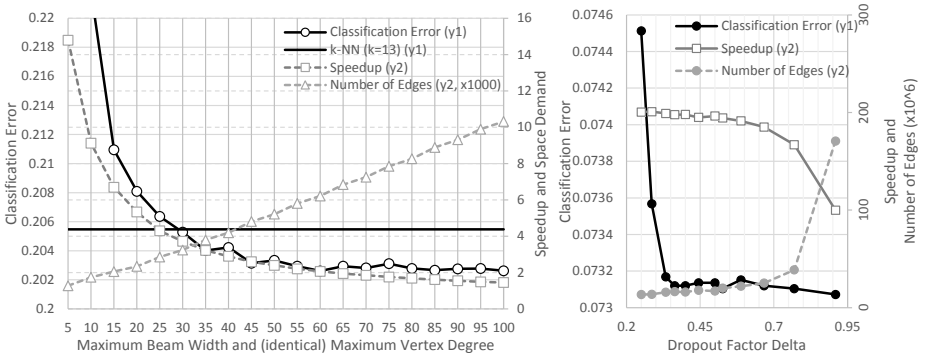

Fig. 2. Left: For a fixed dropout factor $\delta = 0.5$, the impact of the maximum vertex degree m and maximum beam width Ω is analyzed, averaged over 15 UCI domains. Right: For 250k RSS cases, $m = \Omega = 100$ is used with varying δ. Around $\delta = 0.5$, a strong trade-off between accuracy and speedup is achieved.

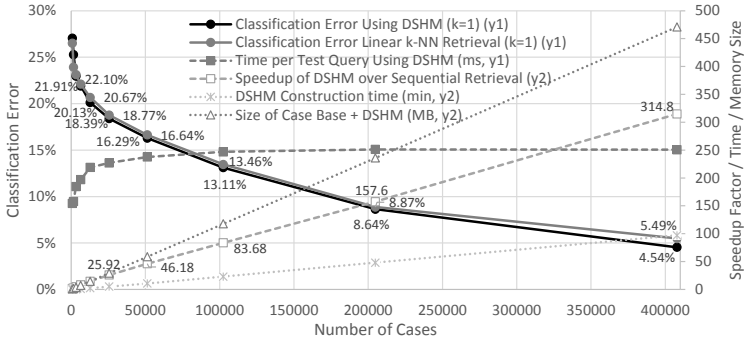

Fig. 3. Settling for fixed maximum vertex degrees ($m = 100$), maximum beam width ($\Omega = 100$) and dropout rate ($\delta = 0.5$), this series of experiments highlights the scaling behavior of DSHMs. We use DSHMs for classification on the basis of a 1-NN retrieval in the RSS domain and compare it to sequential retrieval. For $n = 400k$ cases, a speedup of more than 300 is achieved at linear space requirements.

with $|\mathcal{C}|$. However, and this is a downside, memory consumption (index size) also increases linearly with the number of cases.

Unlike classification, which benefits from more cases, pure retrieval tasks (where cases lack a solution part) becomes harder with larger case volumes when focusing on 1-recall. Let $\tilde{S}$ be the DSHM for case base $\mathcal{C}$. The ground truth over $\mathcal{C}$ for a query q is defined as $g_q = \arg\max_{c \in \mathcal{C}} sim(q, c)$. Let $\mathcal{R}$ be the result of the retrieve Algorithm 5 for $\tilde{\mathcal{S}}$, q, and $k = 1$. We then say that the DSHM-based retrieval *recalls* successfully if $\mathcal{R} = g_q$. The fraction of queries (from a ground truth set) meeting this condition is denoted as 1-recall. We stick to this performance measure, noting that relaxations to $k > 1$ (e.g., 10-recall) are also common in the literature. For a fixed maximum capacity determined by $m = 100$,

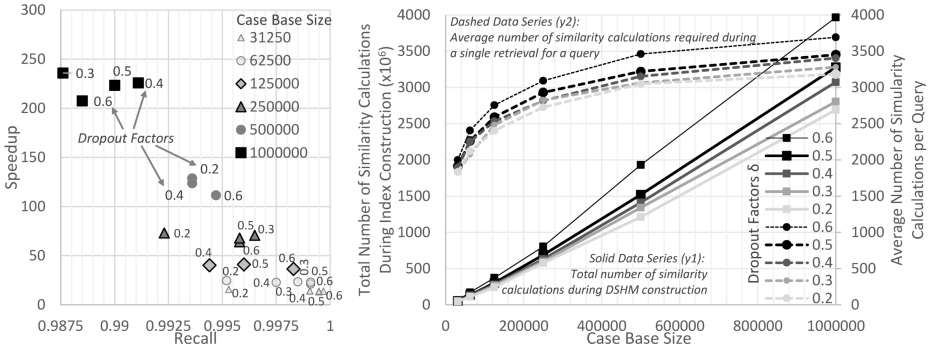

Fig. 4. Analysis for SIFT data with DSHM construction on up to 10^6 cases and testing on 10k ground truth queries. Left: Tradeoff between recall and speedup for $m = 100 = \Omega = 100$, with varying dropout factors δ and data sizes. Right: Same setting with $\Omega = 70$, showing time demand in terms of the number of required similarity calculations.

the left part of Figure 4 shows exemplarily, i.e. for individual runs, recall results for varying case base sizes and dropout factors. For moderate amounts of data nearly perfect recall is achieved with a speedup of 10-20. Generally, larger values of δ imply higher recall at lower speed and at the cost of more edges in the DSHM due to its longer case dropout sequence. The right part of Figure 4 highlights the logarithmic scaling of DSHM-based retrieval time with case base size. For $1M$ cases and $\delta = 0.5$, only ≈ 3500 similarity calculations are needed while achieving a recall > 0.99. Moreover, the chart plots the total number of similarity calculations required to fully construct a DSHM. Since those similarity comparisons make up for the most of the time effort except for a constant factor, these graphs expose the linearithmic time complexity (cf. Section 3.5) of the DSHM construction procedure.

5 Conclusion

Retrieval efficiency in Case-Based Reasoning remains an important topic even after decades of research. In this paper, we have introduced the concept of dropout similarity highways multigraphs which we suggest to use as an efficient index structure during approximate case-based retrieval. The core idea of these multigraph structures is that multiple nested sub-case bases with their own labeled edge connection sets exist. These are exploited in a top-down manner to more quickly approach regions in the case base with more similar cases.

We have proposed appropriate index construction and retrieve algorithms and evaluated them using a selection of benchmark classification problems as well as two large-scale datasets in the million range. The methods we presented are applicable to various problem types, including classification and information retrieval. In both areas, we found that the performance, in terms of classification error and recall versus time complexity, is very good.

References

1. Aha, D., Kibler, D., Albert, M.: Instance-Based Learning Algorithms. Machine Learning 6, 37–66 (1991)
2. Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I., Schmidt, L.: Practical and Optimal LSH for Angular Distance. In: NeurIPS. pp. 1225–1233. Montreal (2015)
3. Bergmann, R., Richter, M., Schmitt, S., Stahl, A., Vollrath, I.: Utility-Oriented Matching: A New Research Direction for CBR. In: (GWCBR) (2001)
4. Beygelzimer, A., Kakade, S., Langford, J.: Cover Tree for Nearest Neighbor. In: Proceedings of ICML 2006. pp. 97–104. ACM Press, Pittsburgh, USA (2006)
5. Forbus, K., Gentner, D., Law, K.: MAC/FAC: A Model of Similarity-Based Retrieval. Cogn. Sci. 19(2), 141–205 (1995)
6. Fu, C., Xiang, C., Wang, C., Cai, D.: Fast Approximate NN Search With The Navigating Spreading-out Graph. Proc. VLDB Endow. 12(5), 461–474 (2019)
7. Gabel, T., Sommer, F.: Case-Based Learning and Reasoning Using Layered Boundary Multigraphs. In: ICCBR 2022. pp. 193–208. Springer, Nancy, France (2022)
8. Gabel, T., Sommer, F.: Instance-Based Opponent Action Prediction in Soccer Simulation Using Boundary Graphs. In: RoboCup 2022. Springer, Bangkok (2022)
9. Indyk, P., Motwani, R.: Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In: ACM Symposium on the Theory of Computing (1998)
10. Jégou, H., Douze, M., Schmid, C.: Product Quantization for Nearest Neighbor Search. IEEE Trans. Pattern Anal. Mach. Intell. 33(1), 117–128 (2011)
11. Juarez, J., Craw, S., Lopez-Delgado, J., Campos, M.: Maintenance of Case Bases: Current Algorithms after Fifty Years. In: Proceedings of the 27th Intl. Joint Conference on Artificial Intelligence (IJCAI). pp. 5457–5463. Stockholm (2018)
12. Kalantidis, Y., Avrithis, Y.: Locally Optimized Product Quantization for Approximate Nearest Neighbor Search. In: CVPR. pp. 2329–2336. IEEE, Columbus (2014)
13. Kelly, M., Longjohn, R.: The UCI ML Repository, https://archive.ics.uci.edu
14. Leake, D., Wilson, D.: Categorizing Case-Base Maintenance: Dimensions and Directions. In: Proc. of 4th Europ. Workshop on CBR. pp. 196–207. Dublin (1998)
15. Li, S., Amenta, N.: Brute-Force k-Nearest Neighbors Search on the GPU. In: 8th International SISAP Conference. pp. 259–270. Springer, Glasgow, UK (2015)
16. Malkov, Y., Yashunin, D.: Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. IEEE Trans. Pattern Anal. Mach. Intell. 42(4), 824–836 (2020)
17. Paredes, R., Chávez, E.: Using the k-Nearest Neighbor Graph for Proximity Searching in Metric Spaces. In: 12th International SPIRE Conference. vol. 3772, pp. 127–138. Springer, Buenos Aires, Argentina (2005)
18. Richter, M., Weber, R.: Case-Based Reasoning. A Textbook. Springer (2013)
19. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson (2020)
20. Smyth, B., McKenna, E.: Building Compact Competent Case-Bases. In: Proceedings of ICCBR-99. pp. 329–342. Springer (1999)
21. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. J. Mach. Learn. Res. 15(1), 1929–1958 (2014)
22. Toussaint, G.: The Relative Neighbourhood Graph of a Finite Planar Set. Pattern Recognit. 12(4), 261–268 (1980)
23. Wang, M., Xu, X., Yue, Q., Wang, Y.: A Comprehensive Survey and Experimental Comparison of Graph-Based Approximate Nearest Neighbor Search. Proc. VLDB Endow. 14(11), 1964–1978 (2021)
24. Wess, S., Althoff, K., Derwand, G.: Using k-d Trees to Improve the Retrieval Step in CBR. In: Proceedings EWCBR 1993. pp. 167–181. Springer, Germany (1993)