Paper Title

Jaouhara Chanchaf
UM6P

Karima Echihabi
UM6P

Algorithm 2: HeuristicKnnSearch

Input: A query vector q, k, N_{root}.
Output: k Nearest Vectors to q.

```
1  Queue Knn[k] ← {(null, ∞_1), ..., (null, ∞_k)};
2  Node N ← N_root;
3  while !N.IsLeaf() do
4      SP = N.SplitPolicy();
5      N = N.RouteToChildNode(q, SP);
6  (v', bsf) = Knn[k];
7  foreach v ∈ N.Vcetors() do
8      if D(v, q) ≤ bsf then
9          Knn.SortedInsert(v, D(v, q));
10     (v', bsf) = Knn[k];
11 return Knn;
```

Algorithm 3: Kashif: InitWorkerThread

Input: *JobArray*, *k*, *JobCounter*, *Barrier*, *KnnResults*, *AllWorkersDone*.

```
1  while True do
2      JobIdx = SyncFetchAndAdd(JobCounter);
3      if JobIdx > JobArray.Size() then
4          while !AtomicLoad(AllWorkersDone) do
5              Thread reaches Barrier;
6          break; // no more jobs to do, and all workers
                  finished their work
7      q = JobArray[JobIdx];
8      ExactKnnSearch(q, k, KnnResults[JobIdx], Barrier);
9      if JobIdx == JobArray.Size() then
           // completed last job
10         AtomicStore(AllWorkersDone, True);
```

Algorithm 4: ExactKnnSearch

Input: A query vectors q, k, knnResults, Barrier, N_{root}.

```
   // local knn results, ordered by distance
1  Queue Knn[k] ← {(null, +∞_1), ..., (null, +∞_k)};
2  Queue pq ← {}; // priority queue
   /* last NN returned by incremental search is at
      position LastIncrResult - 1                    */
3  Integer LastIncrResult ← 0;
4  Integer UpdateStart ← 0; // where new results start
5  Integer UpdateEnd ← 0; // where new results end
6  Boolean NewIncrement ← False;
   /* perform heuristic search and update knn results
      in global array                                */
7  KnnResults ←HeuristicKnnSearch(q, k);
   // initialize priority queues
8  pq.Add(N_root);
9  while !pq.Empty() do
10     NewIncrement = False;
11     while !NewIncrement and !pq.Empty() do
12         N ← pq.Pop();
13         for pos ← LastIncrResult to k − 1 do
14             (v', bsf) ← Knn[pos];
15             if D_lb(N, q) > bsf then
16                 LastIncrResult ← pos + 1;
17                 NewIncrement ← True;
18         if NewIncrement then
19             UpdateStart ← UpdateEnd;
20             UpdateEnd ← LastIncrResult;
21         if N.IsLeaf() then
22             (v', bsf) ← Knn[k];
23             foreach v ∈ N.Vcetors() do
24                 if D(v, q) < bsf then
25                     Knn.SortedInsert(v, D(v, q));
26         else
27             foreach N' in N.ChildNodes() do
28                 if D_lb(N', q) < bsf then
29                     pq.Add(N');
       // copy new results to global kNN Queue
30     CopyResults(Knn, KnnReasults, UpdateStart, UpdateEnd);
       // Wait for other threads to finish current
          increment
31     Thread blocks on Barrier;
```

ABSTRACT

300 word description of the project

PVLDB Reference Format:
Jaouhara Chanchaf and Karima Echihabi. Paper Title, 15(10): XXX-XXX, 2022.
doi:XX.XX/XXX.XX

1 PROPOSED APPROACH

Algorithm 1: KASHIF: INITTHREADPOOL

Input: $JobArray = \{q_1, ..., q_n\}$, k, r_{th}, m the number of threads, $JobCounter$, $AllWorkersDone$.
Output: $KnnResults$.

```
1  Thread Array Workers[m];
2  Barrier Barrier;
3  Boolean AllWorkersDone ← False;
4  Shared Queue KnnResults[n][k] ←
     {{+∞_1, ..., +∞_k}_q1, ..., {+∞_1, ..., +∞_k}_qn};
5  for i ← 0 to m − 1 do
6      Workers[i].InitWorkerThread(JobArray, k,
7      JobCounter, Barrier, KnnResults, AllWorkersDone);
   // get incremental results
8  while !AtomicLoad(AllWorkersDone) and do
9      Coordinator threads waits for new results on Barrier;
       if Recall(KnnResults, GroundThruth) ≥ r_th) then
10         StopAllWorkers();
11         Return KnnResults;
```


ACKNOWLEDGMENTS

We sincerely thank X, Y and Z.

REFERENCES

[1] K. Echihabi, K. Zoumpatianos, T. Palpanas, and H. Benbrahim. The Lernaean Hydra of Data Series Similarity Search: An Experimental Evaluation of the State of the Art. *PVLDB*, 12(2), 2018.

[2] PhDComics. Graduate Student Work Output. https://phdcomics.com/comics/archive.php?comicid=124, 2022.

Proceedings of the VLDB Endowment, Vol. 15, No. 10 ISSN 2150-8097.
doi:XX.XX/XXX.XX