# K-Nearest Neighbor Search in Peer-to-Peer Systems

Hoda Mashayekhi, Jafar Habibi
Computer Engineering Department
Sharif University of Technology
Tehran, Iran
mashayekhi@ce.sharif.edu, jhabibi@sharif.edu

*Abstract*— **Data classification in large scale systems, such as peer-to-peer networks, can be very communication-expensive and impractical due to the huge amount of available data and lack of central control. Frequent data updates pose even more difficulties when applying existing classification techniques in peer-to-peer networks. We propose a distributed, scalable and robust classification algorithm based on k-nearest neighbor estimation. Our algorithm is asynchronous, considers data updates and imposes low communication overhead. The proposed method uses a content based overlay structure to organize data and moderate the number of query messages propagated in the network. Simulation results show that our algorithm performs efficiently in large scale networks.**

***Keywords- Classification; K-Nearest Neighbors; Content Addressable Network; Peer-to-peer systems.***

## I. INTRODUCTION

Huge amounts of data are available in large scale systems such as peer-to-peer (P2P) networks. Data mining in these systems can utilize the distributed resources of data and computation. Many applications such as smart recommendations, query answering, failure determination, and market analysis can benefit from the hidden information and patterns in the distributed data. Due to large computation overhead, scalability and privacy issues, it is impractical to collect the data at different nodes in a central server. In an alternative approach, each node can execute the data mining algorithm to produce a local model of its data. Some approaches transmit these models or representatives of the data to a central server for integration [3][4]. However, pure distributed data mining approaches do not require a central server [2][5]. Datta et al. present an overview of data mining in P2P networks [10].

Well known classification algorithms like decision tree induction [6], nearest-neighbors [7], Bayesian methods [8] and Support Vector Machines [9] require data to reside on a central site. Majority of the available classification techniques have tried to minimize computation costs and number of disk accesses. Whereas different requirements exist in P2P systems, such as minimizing communication overhead, preserving privacy, etc. Special attempts have been made to design distributed classification algorithms for large scale dynamic P2P environments [11].

In this paper we introduce a distributed K-Nearest Neighbors (KNN) [7] algorithm. We apply our proposed algorithm in the Content Addressable Network (CAN) [1], which is a well-accepted P2P overlay network supporting multidimensional data. In our design neighbor peers collaborate to find the k-nearest neighbors of the query. This collaboration is guided by the CAN structure to form a local algorithm [5]. We generally analyze the complexity of our algorithm. The simulation results show that our algorithm can efficiently perform classification in P2P networks.

The rest of the paper is organized as follows. In Section 2, we review the related work. Section 3 describes the system model and basic assumptions. Section 4 elaborates the distributed classification algorithm. Simulation results are presented in Section 5, and finally, Section 6 presents conclusion and future work.

## II. RELATED WORK

Various proposals exist for classification of data in distributed systems [11]. Many of the proposed nearest neighbor algorithms assume a centralized collection of data. Seidl et al. propose a multi-step k-nearest neighbor search algorithm to overcome the shortcomings of single step algorithms [12]. Their algorithm tries to minimize the number of exact object distance evaluations and is suitable for high dimensional and adaptable distance functions. Roussopoulos et al. provide a branch and bound algorithm for processing k-nearest neighbor queries utilizing the R-tree structure [13].

The majority of distributed nearest neighbor classification methods gather data at a central site before executing the classification algorithm. Li et al. use k-nearest neighbor classification in distributed sensor networks for collaborative signal processing [14]. The actual classification is performed in a central node which collects the data of the other nodes in its region. A monitoring algorithm for k-nearest neighbor queries over moving objects is proposed by Yu et al [15]. They propose two methods based on indexing objects or queries, in a two dimensional space. Song et al. propose methods for finding k nearest neighbors of a moving query point. Their algorithm uses R-tree structure [16].

### A. K-nearest Neighbor Classification

Nearest neighbor classifiers [7] are based on learning by analogy. They compare a given query tuple with training data tuples that are similar to it. Each data tuple is described by m attributes. So each tuple represents a point in the m-dimensional space. Given a query tuple with an unknown class attribute, the KNN classifier searches the m-

dimensional space for k data tuples that are closest to the query tuple according to a distance function. The query tuple is assigned the most common class among these k nearest data tuples. To determine the nearest tuples, a suitable distance metric such as Euclidean distance should be used.

### B. Content Addressable Network

CAN [1] considers a virtual m-dimensional Cartesian logical coordinate space on a m-torus. The entire coordinate space is dynamically partitioned among all the peers in the network, such that each peer owns a distinct zone within the overall space. A hash function is used to map each (key, value) pair in the network to a point in the coordinate space. The pair is then stored at the node that owns the zone containing the corresponding point. Retrieval of any entry corresponding to a key is similar; after applying the hash function to the desired key, the related value is retrieved from the appropriate zone owner.

Whenever a new node joins the network, it chooses a random point in the coordinate space and joins the zone containing that point. The zone owner splits its zone along a dimension, chosen on a round robin fashion, and transfers the data belonging to half of the zone to the new node. Also when a node leaves the network, it assigns its zone to one of its neighbor nodes. After any join or leave, the owners of neighbor zones should be informed of the new situation. Also control messages are propagated when a new data point is added to the network. Some other control messages are discussed in [1].

For efficient routing of queries, any node keeps pointers to neighboring zones along each dimension. Any message is always forwarded to the neighbor node which is closer to the destination zone.

## III. System Model

A network of size N is assumed, where each peer stores a portion of the available data in the network. Each data tuple, d, is represented by a m-dimensional attribute vector $(d_1, d_2, \dots, d_m)$. The query is also described by a similar vector. The peers form an m-dimensional CAN overlay over the coordinate space. Attribute vectors are used, instead of the hash function, to map data tuples to points in the coordinate space. To handle dynamics in network topology and also changes in peers' data, the mechanisms introduced in the CAN proposal are employed [1].

Two zones are considered adjacent if they share at least one point in their boundary. Note that this definition defers from the definition of neighboring zones in the CAN proposal [1]. Regarding the latter definition, adjacent zones reside at most two hops away from each other. The distance of any data tuple to an arbitrary zone is the length of the shortest line between the data tuple and the zone boundary. The zone owned by a peer and the data tuples mapped to that zone, are called the local zone and local data of that peer respectively. When a query is initiated, the zone containing the query tuple is called the query local zone and the owner of this zone is referred to as query owner. Also the term region refers to a collection of at least one zone in the CAN overlay.

## IV. The Distributed Classification Algorithm

As a consequence of using attribute vectors to map data tuples to the coordinate space, similar data tuples will reside in approximate zones. The basic idea of our algorithm is that the nearest neighbors of a query are discovered, with high probability, in the query local zone and its approximate zones. This probability is directly proportional to the density of data objects in the zones and inversely proportional to the value of k in the KNN algorithm.

The CAN overlay can be leveraged to find the nearest neighbors of a query in an iterative manner. Figure 1 shows the distributed k-nearest neighbors algorithm. The symbols used in the algorithm are summarized in table 1. When a query q is initiated by a peer, it is routed to the query local zone and the query owner initiates the KNN algorithm. In any iteration, a search region (SR) consisting of zones which may contain the k nearest neighbors of q is investigated. Also a candidate data set (CS) is maintained in different iterations. This set contains the data tuples that may be among the k nearest neighbors of q. The k nearest neighbors of q will eventually be excluded from this set and added to the final answer set.

The main part of the distributed algorithm for finding the k nearest neighbors is iteration on three tasks:

*1)* Searching the current search region for nearest data tuples (lines 7-16 of algorithm 1).

*2)* Updating the final answer set (lines 17-23 of algorithm 1).

*3)* Determining the next search region (lines 24-29 of algorithm 1)

The above tasks are repeated until the final answer set size is k or the search region for the next iteration is empty. Note that for correct behavior of the algorithm it is necessary that the structure of the zones containing k nearest neighbors of q remain unchanged. Also as query owner stores contact information of zone owners in the search region, if these information changes additional routing is required to complete the algorithm.

In the next subsections each task is explored in depth. In the following sections the subscript of the symbols in table 1, is a representative of the iteration number.

### A. Searching the current search region

The first task of algorithm 1 consists of sending query messages by the query owner to all the zone owners in the current search region. The search region for the first iteration is the query local zone, thus in the first iteration the query owner will send a query message to itself. At the beginning of any iteration if the search region is empty, then $k - |KN|$ data tuples with minimum distance to q are extracted from CS and added to KN. The algorithm is then terminated. The candidate data set and the final answer set are empty at the beginning of the algorithm.

All of the nodes that receive a query message should search their local zones for candidate data tuples that may be among the k nearest neighbors of q. A subset of their local data satisfying the conditions of lemma 1 is sent back to the query local zone owner which will insert them in the

TABLE I. DESCRIPTION OF SYMBOLS

| Symbol | Description |
|---|---|
| KN | The final answer set containing k nearest neighbors |
| CS | The set containing candidate data tuples |
| SR | The search region |
| SZ | Previously searched zones |
| BDS | The set containing distances to adjacent zones |
| RZ | The set containing received information of adjacent zones |

candidate data set. They will also compute the distance between q and all of their adjacent zones, and send the set of adjacent zone owners' addresses along with the computed distances to the query local zone owner. This information will be utilized later in the algorithm.

**Lemma 1.** Let $D_{z'}$ denote the data of zone $z'$. The owner of zone $z'$ which has received the query message in iteration i, will reply back with a subset $SD_{z'}$ of its local data, such that $|SD_{z'}| \leq (k - |KN_{i-1}|)$. Also the following property holds:

$$\forall d \in SD_{z'}. \forall d' \in D_{z'} \backslash SD_{z'}. Dist(q, d') \geq Dist(q, d) \quad (1)$$

If $|CS_{i-1}| > (k - |KN_{i-1}|)$, then we also have:

$$\forall d \in SD_{z'}. Dist(q, d) \leq \max\{Dist(q, d') | d' \in CS_{i-1}\} \quad (2)$$

First note that size of $SD_{z'}$ is at most equal to number of desired data tuples, $k - |KN_{i-1}|$, so that no extra data is transmitted. Inequality (1) emphasizes that the data tuples in $SD_{z'}$ have minimum distance to q among all the data tuples in $D_{z'}$. Also if size of $CS_{i-1}$ is greater than number of desired data tuples, no data tuple which has greater distance to q than all the data tuples in $CS_{i-1}$, can be among the k nearest neighbors of q, thus (2) should hold.

After receiving replies from all of the nodes in the search region, the query owner will set $CS_i = \cup_{z' \in RZ_i} SD_{z'} \cup CS_{i-1}$.

### *B. Updating the final answer set*

After obtaining $CS_i$ in the previous task, the query owner should now examine $CS_i$ to extract suitable data tuples and add them to the final answer set.

**Lemma 2.** Let $D_i$ be the set of data tuples mapped to region $SR_i$. If C is the maximum subset of $CS_i$ such that

$$\forall d \in C. Dist(q, d) < \min\{Dist(q, z') | z' \text{is adjacent to } SR_i \wedge z' \notin \cup_{j=1}^{i} SR_j\},$$

then $KN_i = KN_{i-1} \cup C$ and $CS_i = CS_i \backslash C$. Note that if $|KN_{i-1} \cup C| > k$, then k points with minimum distance to q will be kept.

Using lemma 2, any candidate data tuple which is closer to q than to any adjacent zone of $SR_i$ which is not investigated before, is extracted from $CS_i$ and added to $KN_i$. We have the following statement: $\forall d \in \cup_{j=1}^{i} D_j . \forall d' \in CS_i \; Dist(q, d) \geq Dist(q, d')$. If this inequality does not hold for a pair of data tuples d and $d'$ such that $d \in \cup_{j=1}^{i} D_j$ and $d' \in CS_i$, then d can replace $d'$ in the set of candidate data tuples. Also the following property holds for any data tuple d belonging to an adjacent zone:

$$\forall d \in C. \forall d' \in z' | (z' \text{is adjacent to } SR_i \wedge z' \notin \cup_{j=1}^{i} SR_j). (Dist(q, d') \geq Dist(q, z') \geq Dist(q, d)\}. \quad (3)$$

So d is one of the k nearest neighbors of q.

**Algorithm 1**. Finding k nearest neighbors of q
z: query local zone
$O_z$: query owner
$O_z$. Find_k_nearest_neighbors(q,k)
1: KN = ∅, SR = z, SZ = ∅, CS = ∅, BDS = ∅
2: **while** $|KN| < k$ **do**
3: **if** |SR|is zero **then**
4: add $k - |KN|$ tuples from CS with minimum distance from q, to KN
5: terminate the algorithm
6: **end if**
7: RZ= ∅
8: send query message to owners of zones in SR with parameters ( q, k- |KN|, |CS|, max{Dist(q, d)| d ∈ CS})
9: RZ = ∅ // the received candidate adjacent zones
10: **while** not received reply from a zone owner in SR **do**
11: receive message from a zone owner o in region SR
12: CS= CS ∪ received data tuples
13: RZ = RZ ∪ received adjacent zones
14: BDS = BDS ∪ received boundary distances
15: **end while**
16: SZ = SZ ∪ SR
17: minBDS = min{Dist(q, b)|b ∈ BDS}
18: **for** i=1 to |CS| **do**
19: **if** $Dist(q, d_i \in CS) < minBDS$ **then**
20: KN = KN ∪ {$d_i$} //if |KN| = k replace one of the tuples in $KN_q$with $d_i$ if the overall sum of distances is improved
21: CS = CS\{$d_i$}
22: **end if**
23: **end for**
24: maxD = max{Dist(q, $d_i$)|$d_i$ ∈ CS}
25: **if** (|CS| < (k − |KN|)) **then**
26: SR = RZ\{z|z ∈ SZ}
27: **else**
28: $SR = \{z' | z' \in RZ \wedge Dist(q, z') < maxD\} \backslash \{z | z \in SZ\}$
29: **end if**
**end while**

Figure 1. The distributed KNN algorithm

### *C. Determining the next search region*

If $|KN_i| < k$, the search area should be expanded. Lemma 3 is used to construct the search region for the next iteration.

**Lemma 3.** In any iteration we have:

$$\forall i > 1. \forall j < i. SR_i \cap SR_j = \emptyset \quad (4)$$

Also for any zone $z'$ which is adjacent to $SR_i$ and $z' \notin \bigcup_{j=1}^{i} SR_j$ we have:

$$\text{If } |CS_q| < k - |KN_i| \text{ then } z' \in SR_{i+1}$$

$$\text{else } (\exists d \in CS_q. \text{Dist}(q, d) > Dist(q, z') \Rightarrow z' \in SR_{i+1} \quad (5)$$

If less than $k - |KN_i|$ data tuples are available in the candidate data set, then any zone adjacent to the current search region which is not investigated before should be included in the next search region, as it may contain closer points to the query. Else, the distance between q and adjacent zones should be calculated. Only those zones are included in the next search region that the distance between them and q is less than the distance of q to at least one of the candidate data tuples. Consequently these zones may contain data tuples that are closer to the query. The distance between q and zones adjacent to the current search region is calculated by zones in $SR_i$ and sent to the query owner in task 1. Also addresses of the adjacent zone owners, is sent to the query owner so that in subsequent iterations, it can contact the adjacent zones directly. Note that $SR_{i+1}$ does not contain any zones searched in the previous iterations.

If the total number of data tuples in the network is greater than k, then the termination of the algorithm is guaranteed. Figure 2 illustrates examples of non expandable (a) and expandable (b) search regions in a 2-dimensional CAN overlay for k=5. In Figure 2 (b), the distance from q to the zones z2-z4 is less than the distance from q to data tuple 1. Therefore these zones should be investigated in the second iteration.

## V. Analysis and Experimental Results

In this section we present a general analysis on the message complexity of our algorithm in a static network.

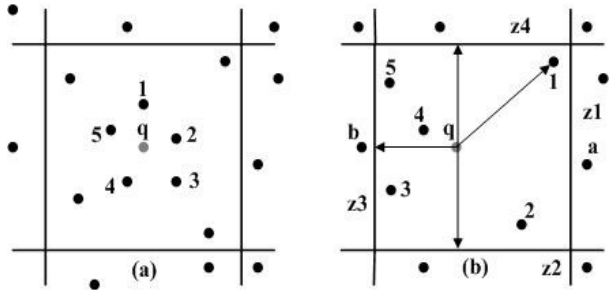


Figure 2. Examples of (a) nonexpandable and (b) expandable search regions for k=5.

Recall the three steps of the algorithm. Note that only step 1 imposes communication in the network. In any iteration the query owner sends a message to all zone owners in the search region. As the search regions in different iterations do not have any zones in common, no zone receives more than one message from the query owner. Also in other than the first iteration the query owner receives addresses of adjacent zones from the zones in the search region. So it can contact them directly without routing the message in the CAN overlay. So if we could determine the maximum number of zones investigated in the algorithm, the number of messages could be calculated.

Define the diameter of an m-dimensional hypercube as the largest distance between any two of its vertices. Assume that the minimal hypercube centered at the q, which contains at least k data tuples other than q, has diameter l. So the distance from q to any of these k data tuples is at most l. To determine the maximum number of zones searched by the algorithm, consider the minimal m-dimensional hypercube centered at q, which contains the union of all search regions in different iterations. By extending lemma 3, it is induced that the edge size of this hypercube should be at most $2l + \varepsilon$, where $\varepsilon \to 0$, so that the algorithm investigates all the zones that may contain the k nearest neighbors. This observation shows that the communication overhead of our algorithm is independent of the network size and it is considered a local algorithm [5]. Having the average number of data tuples contained in any zone of the CAN overlay, the minimum number of zones that are investigated can be determined.

We conducted a simulation to evaluate our proposed algorithm in a CAN network. We used two different synthetically generated data sets containing 20000 data tuples in our experiments. Figure 3 shows the snapshots of the test data used to evaluate the algorithm, which are generated using the uniform and 5 Gaussian distributions with pre selected centers and standard deviation. Each simulation is executed 10 times for random queries and the average value is displayed. Similarity is measured using Euclidean distance.

We have also compared our algorithm with the KNN algorithm executed on P2PR-tree [17]. We extended the NN algorithm proposed in [18] for KNN by modifying the pruning rule for k data tuples, and implemented it on top of P2PR-tree. As each node in the P2PR-tree keeps information about the top level of the tree, it can initiate the KNN algorithm. Different parameters used in P2PR-tree are set as follows: number of blocks and groups in each block is set to 10, number of routers of each node is set to 1, $G_{Max}$ and $SG_{Max}$ are set to 50 and minimum number of nodes in a subgroup is set to 20. Tree vertices other than blocks and leaves have two children. The tree node splitting algorithm is introduced in [20]. We consider a power law distribution of data among the peers. Each peer is assigned a maximum of 100 data tuples based on the Sarioiu distribution [19].

As our algorithm uses CAN overlay network as its base, it is useful to consider the message complexity incurred by maintaining this overlay. To better reveal the efficiency of the algorithm, we have compared the message complexity of our algorithm with the simple distributed KNN algorithm in which queries are broadcasted in the network. In the latter algorithm an unstructured P2P network is considered. In such a network, a new node sends join requests to some

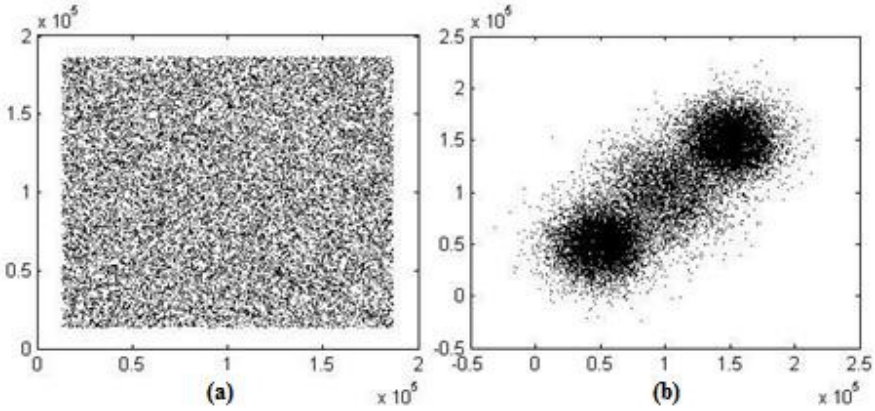


Figure 3. (a) Two dimensional uniform data, (b) Two dimensional mixture of Gaussian data sets

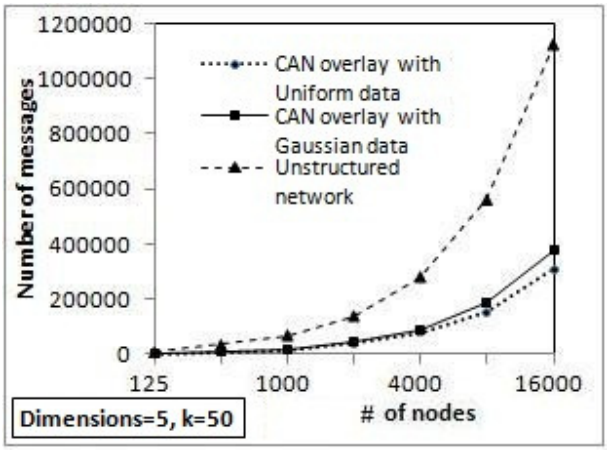


Figure 4. Number of messages incurred in the CAN network and unstructured network

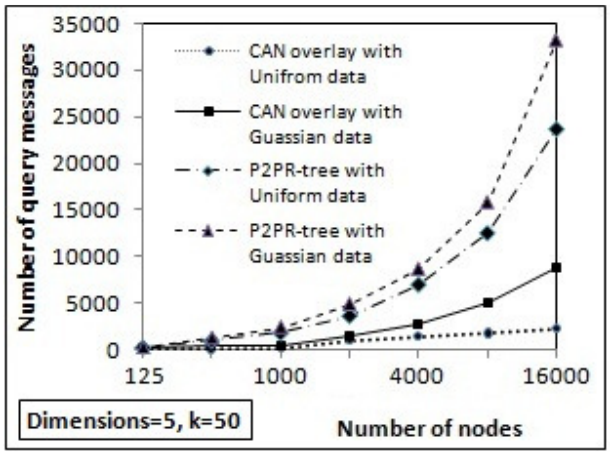


Figure 5. Query messages incurred by algorithms when number of nodes is increased

randomly chosen nodes. But when leaving the network no message overload is incurred.

Figure 4 shows the average number of messages per request in a dynamic CAN network compared to an unstructured network. A request can be a join, leave, or KNN query request and also addition of new data to the network. As seen in the figure, although our algorithm imposes high control traffic load in the network, it outperforms executing the KNN algorithm in the unstructured network, due to the low query traffic.

Figure 5 shows the message complexity of our algorithm and the KNN algorithm in P2PR-tree under different network sizes. As observed, the number of query messages for Gaussian data is larger than uniform data. In the former case, there are sparse areas in the network with few data tuples. If the query resides in these areas, more query messages should be propagated in the network. Also, Figure 5 exposes the efficiency of our algorithm compared to KNN in P2PR-tree.

The performance of our algorithm when the nodes form a regular m-dimensional mesh is shown in Figure 6 and Figure 7. In such networks, number of adjacent zones of a particular zone is proportional to number of dimensions. So as observed in Figure 6, when number of dimensions is 10, the query traffic is much less than the same configuration in

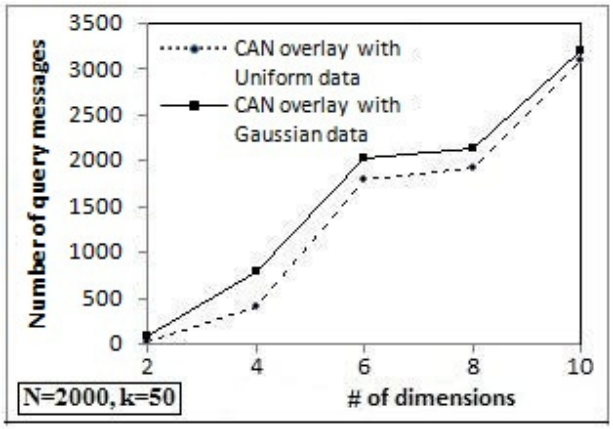


Figure 6. Effect of changing number of dimensions on number of query messages

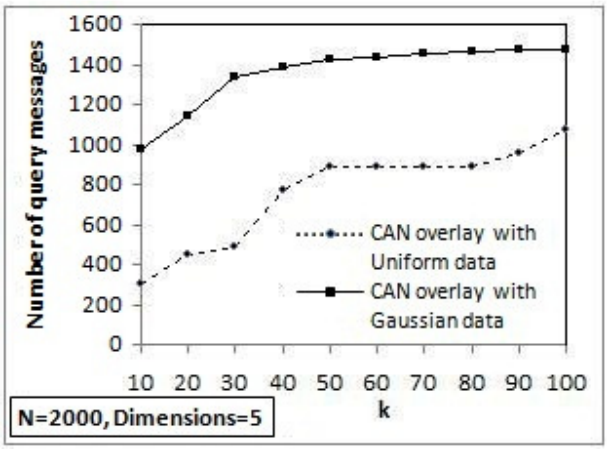


Figure 7. Effect of changing the value of parameter k on number of query messages

Figure 5. Obviously when number of dimensions increases, more query messages are propagated through the network. Figure 7 shows the effect of changing the parameter k in the KNN algorithm, on number of query messages. As observed by increasing this parameter, number of query messages increases. The parameter k has a more visible effect when the data tuples are distributed uniformly in the network. This is due to the fact that the average number of nodes' data is the same in the uniform configuration. Therefore by increasing the parameter k, eventually more zones should be investigated.

## I. CONCLUSION

In this paper we studied classification of data objects in P2P networks. We presented a new distributed algorithm for finding the k nearest neighbors of a query in P2P networks. Our algorithm uses a content addressable network to organize data and moderate the number of query messages propagated in the network. Using pruning rules, the number of nodes that should be prompted to find the k nearest neighbors is decreased. Simulation results show the effectiveness of our algorithm in different configuration. We have presented comparisons with the KNN algorithm executed in an unstructured P2P network and in a network with P2PR-tree structure. Extending our algorithm to dynamically adapt the query answer, when nodes leave and join the network or k is updated, remains as future work. Also examining the effectiveness of our algorithm with real world non-numerical data sets can further expose its strengths and weaknesses.

## REFERENCES


[1] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," Proc. 2001 conference on Applications, technologies, architectures, and protocols for computer communications, ACM Press, 2001, pp. 161-172, doi: 10.1145/383059.383072.

[2] M. Li, G. Lee, W. C. Lee, and A. Sivasubramaniam, "PENS: An algorithm for density-based clustering in peer-to-peer systems," Proc. 1st international conference on Scalable information systems, ACM Press, 2006, Article No. 39, doi: 10.1145/1146847.1146886.

[3] J. da Silva, C. Giannella, R. Bhargava, H. Kargupta, and M. Klusch, "Distributed Data Mining and Agents," Eng. Applications of Artificial Intelligence, vol. 18 (7), 2005, pp. 791-807, doi: 10.1016/j.engappai.2005.06.004.

[4] E. Januzaj, H. P. Kriegel, and M. Pfeifle, "DBDC: Density Based Distributed Clustering," Proc. Ninth Int'l Conf. Extending Database Technology (EDBT '04), Springer Berlin / Heidelberg Press, LNCS, vol. 2992, 2004, pp. 88-105, doi: 10.1007/b95855.

[5] R. Wolff and A. Schuster, "Association Rule Mining in Peer-to-Peer Systems," IEEE Transactions on Systems, Man and Cybernetics - Part B, vol. 34 (6), 2004, pp. 2426-2438.

[6] J. R. Quinlan, "Induction of Decision Trees," Machine Learning, vol. 1(1), 1986, pp. 81-106, doi: 10.1023/A:1022643204877.

[7] B. V. Dasarathy, "Nearest Neighbor (NN) Norms: NN Pattern Classification techniques," IEEE Computer Society Press, 1991.

[8] T. M. Mitchell, "Machine learning," McGraw Hill Higher Education, 1997.

[9] B. Boser, I. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," Proc. Fifth Annual Workshop on Computational Learning Theory, ACM Press, 1992, pp. 144-152, doi: 10.1145/130385.130401.

[10] S. Datta, K. Bhaduri, C. Giannella, R. Wolff, and H. Kargupta, "Distributed Data Mining in Peer-to-Peer Networks," IEEE Internet Computing Special Issue on Distributed Data Mining, vol. 10 (4), 2006, pp. 18-26, doi: 10.1109/MIC.2006.74.

[11] S. J. Stolfo, A. L. Prodromidis, S. Tselepis, W. Lee, D. W. Fan, and P. K. Chan, "JAM: Java Agents for Meta-Learning over Distributed Databases," Proc. Third International Conference on Knowledge Discovery and Data Mining, AAAI press, 1997, pp. 74–81, doi: 10.1.1.44.5728.

[12] T. Seidl and H. P. Kriegel, "Optimal Multi-Step k-Nearest Neighbor Search," Proc 1998 ACM SIGMOD international conference on Management of data, ACM Press, 1998, pp. 154-165, doi: 10.1145/276304.276319.

[13] N. Rousopoulos, S. Kelley, and F. Vincent, "Nearest Neighbor Queries," Proc. 1995 ACM SIGMOD international conference on Management of data, ACM Press, 1995, pp. 71-79, doi: 10.1145/223784.223794.

[14] D. Li, K. D. Wong, Y. Hu, A. M. Sayeed, "Detection, Classification and Tracking of objects in Distributed Sensor Networks," IEEE Signal Processing Magazine, vol. 19 (2), 2002, pp. 17-29, doi: 10.1.1.58.2340.

[15] X. Yu, K. Pu, and N. Koudas, "Monitoring k-Nearest Neighbor Queries Over Moving Objects," Proc. 21st International Conference on Data Engineering, IEEE Press, 2005, pp. 631-642, doi: 10.1109/ICDE.2005.92.

[16] Z. Song and N. Roussopoulos, "K-Nearest Neighbor Search for Moving Query Point," Proc. 7th International Symposium on Advances in Spatial and Temporal Databases, Springer-Verlag Press, 2001, pp. 79-96, doi: 10.1.1.108.8564.

[17] A. Mondal, Y Lifu, and M Kitsuregawa, P2PR-tree: An R-tree-based Spatial Index for Peer-to-Peer Environments, Proc. Current Trends in Database Technology - EDBT 2004 Workshops, Springer Berlin / Heidelberg Press, LNCS, vol. 3268/2005 (516), 2005, DOI: 10.1007/978-3-540-30192-9_51

[18] S. Berchtold, C. Bohm, D. Keim, and H. Kriegel. A cost model for nearest neighbor search in high-dimensional data space. In Proc. ACM Symp. on Principles of Database Systems, ACM Press, 1997, pp. 78-86, doi: 10.1145/263661.263671.

[19] S. Saroiu, P. K. Gummad, SD. Gribble, A measurement study of peer-to-peer file sharing systems, Proc of Multimedia Computing and Networking, 2002.

[20] A. Guttman, R-trees: a dynamic index structure for spatial searching, Proc of the ACM SIGMOD Conference on Management of Data, ACM Press, 1984, pp. 47-57, doi: 10.1145/602259.602266.