Approximate Nearest Neighbor Search Small World Approach

Alexander Ponomarenko, Yury Mal'kov, Andrey Logvinov, Vladimir Krylov
MERA Labs LLC, Nizhny Novgorod, Russia
aponom@meralabs.com, ymalkov@meralabs.com, alogvinov@meralabs.com, vkrylov@meralabs.com

ABSTRACT

In this paper we propose a novel approach to solving the nearest neighbor search problem. We propose to build a data structure where the greedy search algorithm can be applied which is known to have logarithmic complexity in structures with navigable small world properties. The distinctive feature of our approach is that we build a non-hierarchical structure with possibility of local minimums which are circumvented by performing a series of searches starting from arbitrary elements of the structure. The performed simulation shows that the structure built using the proposed algorithms has navigable small world properties with logarithmic search complexity which is retained even for high-dimensional data.

Keywords: Similarity Search, Small World, Distributed Data Structure

1. INTRODUCTION

We present a new approach for solving nearest neighbor search problem in general metric space. This problem appears when we need to find a closest object $p \in X$ from finite set of objects $X \subseteq \mathcal{D}$ to given query $q \in \mathcal{D}$, where $\mathcal{D}$ is the set of all possible objects (data domain). Closeness or proximity of two objects $o', o'' \in \mathcal{D}$ is defined as distance function $d(o', o'')$.

In general, the search problem can be described as follows: Let $\mathcal{D}$ be a domain, d a distance measure on $\mathcal{D}$, and $(\mathcal{D}, d)$a metric space. Given a set $X \subseteq \mathcal{D}$ of n elements, preprocess or structure the data, so that proximity queries are answered efficiently.

The nearest neighbor search problem is relevant to many applications such as pattern recognition and classification [**1**], content-based image retrieval [**2**], machine learning [**3**], Recommendation systems [**4**], searching similar DNA sequence [**5**], semantic document retrieval [**6**].

In a trivial case data structure S is a simple linear list. The complexity of addition operation is $O(1)$, but searching for closest object for q requires evaluation of the metric function for every element from the set of objects X. This amounts to complexity $\theta(n)$, where n is the number of objects in X.

General way to reduce amount of distance measure calculations consists of building a set of equivalence classes, discarding some classes, and exhaustively searching the rest [**7**]. Authors also showed that two main techniques based on equivalence relations, namely, pivoting and compact partitions encompass all the existing methods. Pivot technique relies on taking k pivots and mapping the metric space onto $\mathcal{R}^k$ using the L_∞ distance and they can outperform a compact partitioning index if it has enough memory. Methods based on compact partition are more efficient for spaces with high dimensionality.

However, methods from both classes generally use either data structures with tree topology (GNAT, GHT, SAT, BST, VT, MT) or, in some cases, distance matrix (ALAESA, LAESA)

We suggest using for solving nearest neighbor problem a data structure with small word network topology presented by graph $G(V, E)$, where every object o_i from X is uniquely associated with vertex v_i from V. Thereby searching for the closest element to query q from the data set X will take the form of searching for a vertex in the graph $G(V, E)$.

Application of that approach is based on follows:

- There exist algorithms for building small world networks that have the ability to perform nearest neighbor and addition of a new object to the structure with complexity of $\log n$ [**8**].
- Small world networks have no root element.
- All operations (addition and search) use only local information and can be initiated from any element that has been added to the structure.

This gives opportunity for building decentralized similarity search oriented storage systems where physical data location doesn't depend on the content because every data object can be placed on the arbitrary physical machine and can be connected with other by links like in p2p systems. Such storage systems can provide simultaneous access to large numbers of users for performing data search and addition), have good fault tolerance and have unlimited scalability in terms of performance and capacity.

One of the basic vertex search algorithms in graphs is greedy search. This algorithm has simple implementation on the structure that has small world network topology and can be initiated from every vertex.

In order for the result of the algorithm to be the exact nearest element to the query, the network should contain the Delaunay graph as its subgraph, which is dual to the Voronoi tessellation [**9**].

However, the requirement of search in for the exact nearest neighbor can be excessive (optional) for the applications described above. So the problem for finding the exact nearest neighbor can be substituted for the approximate nearest neighbor search, since we don't need to support whole/exact Delaunay graph.

For the search algorithm to be logarithmically scalable, the small world network should have navigation property that was already discussed in [**8**]

In this paper we present the algorithm for data structure construction based on small world network topology with graph $G(V,E)$ which uses greedy search algorithm for finding approximate nearest neighbor. Graph $G(V,E)$ contains approximate Delaunay graph and has navigation property. Search algorithm has the ability to change accuracy of search without modification of the structure. Presented algorithms do not use the coordinate representation and do not presume the properties of linear spaces, because they are based only on the metric computation between objects, and therefore is applicable to data from general metric spaces.

2. RELATED WORKS

Kd-tree [**10**] and quadra trees [**11**] were among the first structures for solving exact nearest neighbor search problem. They perform well in 2-3 dimensions (search complexity is close to $O(\log n)$), but analysis of the worst case for that structures [**12**] indicates $O(d * N^{1-1/d})$ search complexity, where d is dimensionality.

Other structures which have tree topology such as variants of kd-trees, R-trees and structures based on space-filling curves are surveyed in [**13**]. They also have good performance when searching in a low-dimension ($d < 4$) metric space, but they quickly lose their effectiveness with increasing number of dimensions [**14**]. A more effective data structure for exact nearest neighbor search in $\mathfrak{R}^d$ with search complexity $O(2^d \log n)$ has been described in [**15**]. But as can be seen, search complexity has exponential dependence from the number of dimensions.

Structures such as mvp-tree [**16**], vp^s-tree and vp^{sb}-tree [**17**] use "vantage point" technique, but no analysis has been provided for search complexity in spaces with high number of dimensions.

In general, presently there are no methods for effective exact nearest neighbor search in high-dimensionality metric space. The reason behind it lies in the "curse" of dimensionality [**7**].

To avoid the curse of dimensionality while retaining the logarithmic scaling of the number of elements, it was proposed to reduce the requirements for finding the nearest neighbor, making it approximate.

Thus a large number of papers appeared which proposed to search for nearest neighbor with ε accuracy (ε-NNS). For example, Arya and Mount proposed methods with search complexity $O(\log^3 n)$, but preprocessing requires $O(n^2)$ and algorithm was applicable only to data from E^d [**18**].

Kleinberg proposed two methods [**19**] for solving ε-NNS. First method requires $O(\text{n} \log d)^{2d}$ preprocessing time and query time polynomial in d, ε and $\log n$. Another method with preprocessing polynomial in d, ε and n, but with query time $O(n + d\log^3 n)$. Also both methods are applicable only to data from E^d

The first algorithms with search complexity polynomial in d, $\log n$, ε^{-1} and polynomial preprocessing time for fixed ε were proposed by Indyk and Motwani in [**20**] and Kushilevitz, Ostrovsky and Rabani in [**21**]. Indyk and Motwani were the first ones to relax ε-ANN problem to approximate point location in equal balls (ε-PLEB). For the formulation of the problem in ε-PLEB points in metric space expand to the balls with center at this point and radius $(1+\varepsilon)r$, it is necessary to determine which ball belongs to the query q. Also in [**20**] proposed a second method, which uses the concept of locality-sensitive hashing regarding formulation of the problem ε-PLEB, with search time $\mathrm{O}(\mathrm{n}^{1/(1+\varepsilon)})$,
however requires near quadratic memory (for small ε). In addition, the first method is applicable only for E^{d}, and the second for the Hamming space.

In general, the concept of locality-sensitive hashing has become popular in the last decade to solve the ANN problem. Other works using the concept of locality-sensitive hashing are [**22**], [**23**]. But they all have the same major drawback: each algorithm is focused on a narrow class of metrics such as Hamming distance, Jakarta or l_s norms for Euclidean space. Thus it is necessary to create digests in order to decide which method to choose.

The first structure for solving ANN in E^d with topology of small world networks is Raynet [**24**]. It is an extension of earlier work by the same authors Voronet [**25**], which solved the problem of the exact NN in E^2. Originally Voronet was envisioned as a p2p network, where every node has coordinates in E^2. In Raynet every node has the coordinates in E^d. The system supports two levels of links - short for correct work of the greedy search algorithm and long - for logarithmic search. Short links correspond to edges of Delaunay graph, i.e. each object has references to objects that are neighbors of its Voronoi region. The main difference of Raynet from Voronet is that in Raynet every object doesn't know all of its Voronoi neighbors, i.e. Raynet obtains neighborhood with approximately using the Monte Carlo method.

Raynet is the closest work to ours in terms of general concept. But unlike Raynet, we propose a structure that works with objects from arbitrary metric spaces.

3. STRUCTURE OVERVIEW

We solve the problem of approximate nearest neighbor search formulated as follows: given objects from domain $\mathcal{D}$ with distance function $\mathrm{d}: \mathcal{D} \times \mathcal{D} \longrightarrow \mathbb{R}$. For finite set $\mathrm{X} = \{\mathrm{x}_1, \dots, \mathrm{x}_\mathrm{n}\}$, $\mathrm{X} \subset \mathcal{D}$ an effective probability search method is required to find $\mathrm{x}_\mathrm{i} \in \mathrm{X}$ which is closest to $\mathrm{q} \in \mathcal{D}$. Effective method means that search complexity must scale logarithmically with the number of elements in X. The exact search is not guaranteed, i.e. the result of the algorithm may be an element that is not true nearest neighbor, nevertheless structure and algorithms are designed to minimize the probability of this and there is a possibility to adjust it by varying the parameter of the search algorithm without changing the structure**.**

The structure of S is constructed as a small world network described by a graph $G(V, E)$, where the objects from the set X are uniquely mapped to the vertices from the set V. The set of edges E is determined by the structure construction algorithm, so as to ensure correct operation of the greedy search algorithm.

Since in the proposed structure each vertex is uniquely mapped to an element from the set X, we will use the terms "vertex", "element" or "object" interchangeably. We will use the term “friends” for vertices that share an edge. List of vertices that share a common edge with the vertex v_i is called the friend list of vertex v_i.

4. SEARCH ALGORITHM

Greedy Search

The basic search algorithm traverses the edges of the graph $G(V, E)$ from one vertex to another. The algorithm takes two parameters: `query` and the vertex $V_{enter\_point} \in V[G]$ which is the starting point of search (the entry point). Starting from the entry point at each vertex the algorithm computes the metric value from query q to each vertex from the friend list of the current vertex and then selects the vertex with minimal value of the metric. If the metric value between the query and the selected vertex is smaller than between the query and the current element, then the algorithm moves to that vertex. After that the algorithm repeats. The algorithm stops at the vertex whose friend list doesn’t contain a vertex that is closer to the query than the vertex itself. That vertex is a local minimum.

```
Greedy_Search(q: object, $v_{enter\_point}$: object)
  $v_{curr} \leftarrow v_{enter\_point}$;
  $d_{min} \leftarrow$ d(q, $v_{curr}$); $v_{next} \leftarrow$ NIL;
  foreach $v_{friend} \in v_{curr}$.getFriends() do
     if d(query, $v_{friend}$) < $d_{min}$ then
        $d_{min} \leftarrow$ d(q, $v_{friend}$);
        $v_{next} \leftarrow v_{friend}$;
 if $v_{next}$ = Nil then return $v_{curr}$;
  else return Greedy_Search(q, $v_{next}$);
```

The element which is a local minimum with respect to query q, can be either the true closest element to the query `q` from the entire set of elements of X, or a false closest..

If every element in the structure had in their friend list all of its Voronoi neighbors, then this would exclude the existence of false local minimums. Maintaining this condition is equivalent to constructing Delaunay graph, which is dual to the Voronoi diagram.

Because it is impossible to determine exact Delaunay graph [**26**] (excluding the variant of the complete graph) we cannot avoid the existence of local minimums.

But for the problem of approximate searching as defined above it is not an obstacle since approximate search does not require the entire Delaunay graph [**24**]. As shown below, the probability of finding the true nearest element tends exponentially towards 1 with increase of the average number of edges in the approximated Delaunay graph.

Multi Search

In order to be able to find the true closest element in a network with local minimums, we propose the following modification of the search algorithm. We propose to use a series of m searches initiated from random vertices and choose the result element that is closest to the query from the set of found elements. Since the greedy search `Greedy_Search(q,` $\mathrm{v}_{\mathrm{enterPoint}} \in \mathrm{V}$`)` is deterministic for each entry point $\mathrm{v}_{\mathrm{enterPoint}} \in \mathrm{V}$ it either results in a success - finding the true nearest neigbor, or with a failure - finding the element that is not the nearest neighbor of q.

Thus search of the closest element to the same query q may result in finding of the true nearest neighbor or a false nearest neigbor depending on the entry point from which the search algorithm started.

Since we can choose the entry point at random, there is a probability p of finding the true closest to the particular element q (but not to all elements). Moreover, this probability is always nonzero, because it is always possible to choose the exact nearest neighbor as the entry point, which subsequently will be returned by the greedy search algorithm.

If probability to find true closest in one search attempt is p then probability to find the same element in m search attempts is $1 - (1 - p)^m$, so failure probability decreases exponentially with the number of search attempts. Thus, we can improve search precision, increasing the parameter m - number independent searches.

```
Multi_Search(object q, integer: m)
 results: SET[objects];
 for (i ⟵ 0; i < m; i++) do
   enter_point ⟵ getRandomEnterPoint();
   local_min ⟵ Greedy_Search(query, enter_point)
   if local_min ∉ results then
      results.add(result);
 return results;
```

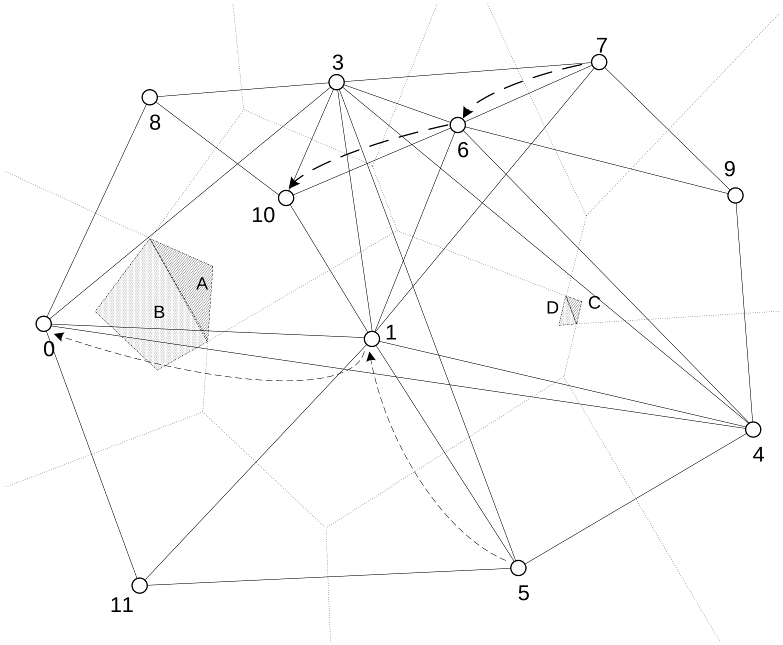

Fig 1

If $m = n$, where n is the number of elements in the structure, the algorithm becomes exhaustive search.

If the graph of the network has small-world properties, then it is possible tp choose a random vertex in a number of random steps proportional to $\log n$, which doesn't affect overall logarithmic search complexity.

Therefore the overall complexity of the search will increase no more than m times.

5. DATA ADDITION ALGORITHM

Since we build an approximation of the Delaunay graph, there is much freedom in the choice of construction algorithm. For example in [**24**] it is proposed to build approximate Delaunay graph which minimizes the volume of Voronoi region for a fixed number of edges for each vertex in the graph. In [**27**] it is proposed to connect new element with k closest objects which are already in the structure. It is based on the idea that intersection of the set of elements which are Voronoi neighbors and the k closest elements is large. In [**27**], [**28**] authors also have shown theoretically and confirmed by experimental results that graph which constructed by proposed algorithm has properties of small world network if elements arrive in random order.

We propose a modified variant of this algorithm which is distinguished by the fact that that the search for k nearest elements uses a series of searches.

The algorithm takes three parameters: the object to be added to the structure and two positive integers `k` and `init_attempts`. First, the algorithm determines a set of local minima, using the procedure `Multi_Search`, which produces a series of independent searches on init_attempts of randomly selected elements from the set of objects that already have been added to the structure. After that algorithm determines neighborhood `u`, which contains all neighbors of each found local minimums. Set `u` is sorted in ascending order by distance from the object `new_object` to be added. After that `new_object` is connected with the first `K` nearest elements from the set of `u`.

```
Nearest_Neighbor_Add(object: new_object,
integer: k, integer: init_attempts)
  SET[object]: localMins ⟵
MultiAttempts_Search(new_object,
init_attempts);
  SET[object]: u ⟵ ∅; //neighborhood;
  **foreach** object: local_min ∈ localMins
**do**
    u ⟵ u ∪ local_min.getFriends();
  *sort the set* u *so to satisfy the
condition* d(u[i], new_object) < d(u[i+1],
new_object)
  **for** (i ⟵ 0; i < k; i++) **do**
    u[i].connect(new_object);
    new_object.connect(u[i]);
```

Fig 1 shows the structure which is constructed by Nearest_Neighbor_Add algorithm for points from E2. Circles denote the elements. The numbers near thems correspond to the addition order. Solid lines show the links (edges) between elements. Dotted lines correspond to the borders of Voronoi tessellation. Delaunay graph edges between elements 0 and 10, 1 and 9 are missing. The structure obtained by the algorithm with parameters m = 3 and attemptsNumber = 5. Element with number 0 is a local minimum, which is not the closest to queries that fall into the shaded area "A", respectively 10 for the "B", 9 for D and 1 for the region"C". Hatched lines show the paths of the two search algorithm runs for query q in the region "B". The algorithm run which starts from vertex 7 stops on the element 10 which is local minimum, but not the closest to the query q. However, the algorithm run which starts from vertex 5 finds the true closest vertex to the query q.

6. EXPERIMENT RESULTS

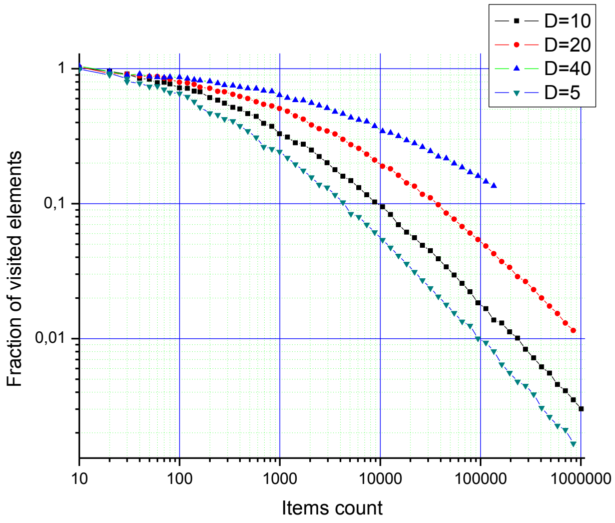

Fig 2

We have implemented the algorithms presented above in order to validate our assumptions about the logarithmic search complexity dependence of the total number of elements.

We used randomly selected points from the E^D as test dataset. L_2 (Euclidean distance) was selected as proximity function

n elements were added to the structure. We chose the number of search attempts m so that the probability of finding the true closest element to the query was not less than 95%. The number of metric calculations was measured. The graph shows the percent of scanned elements (vertical) with an increase in the number of added elements in the structure (horizontal).

The graph (**Fig 2**) shows that with the increase of number of elements in the structure, the percentage of visited elements decreases, and the curve becomes a straight line with angle 45 degrees. This gives us grounds to speak of logarithmic complexity of the search on the number of scanned elements.

The graph shows that the curve for higher dimensions behaves similarly. From this we can make the assumption that there is no exponential dependence from the dimension of space. But it requires more careful study.

7. CONCLUSION

We have proposed a method of organizing data into a small world topology data structure suited for approximate nearest neighbor search in metric space.

We have created a modified k nearest neighbor connection algorithm which is one of the possible algorithms for construction of small world data structures with navigation properties.

Simulation results confirm logarithmic dependency of search complexity from the number of elements in the structure.

All proposed algorithms use only local information on each step and can be initiated from any vertex.

All elements in the structure are of the same type, there is no central or root element.

Thus, all mentioned structure properties are a basis for using the structure for building totally decentralized data storage systems.

8. REFERENCES

[1] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21-27, Jan. 1967.

[2] M. Flickner, et al., "Query by image and video content: the QBIC system ," *Computer*, vol. 28, no. 9, pp. 23-32, Sep. 1995 .

[3] Salzberg, S. Cost, and Steven, "A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features," *Machine Learning*, vol. 10, no. 1, pp. 57-78, 1993.

[4] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, New York, USA, 2001, pp. 285-295.

[5] Rhoads, W. Rychlik, and R. Unknown, "A computer program for choosing optimal oligonudeotides for filter hybridization, sequencing and in vitro amplification of DNA," *Nucletic Acids Research*, vol. 17, no. 21, pp. 8543-8551, Oct. 1989.

[6] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman, "Indexing by Latent Semantic Analysis," *J.*

Amer. Soc. Inform. Sci., vol. 41, pp. 391-407, 1990.

[7] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín, "Searching in metric space," *Journal ACM Computing Surveys (CSUR)*, vol. 33, no. 3, pp. 273-321, Sep. 2001.

[8] J. Kleinberg, "The Small-World Phenomenon: An Algorithmic Perspective," *ANNUAL ACM SYMPOSIUM ON THEORY OF COMPUTING*, vol. 32, pp. 163-170, 2000.

[9] F. Aurenhammer, "Voronoi diagrams — a survey of a fundamental geometric data structure," *ACM Computing Surveys (CSUR)*, vol. 23, no. 3, pp. 345-405, Sep. 1991.

[10] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18 , no. 9, pp. 509-517, Sep. 1975.

[11] Bentley and R. Finkel, "Quad Trees: A Data Structure for Retrieval on Composite Keys," *Acta Informatica*, vol. 4, no. 1, pp. 1-9, 1974.

[12] Wong and Lee, "Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees," *Acta Informatica*, vol. 9, no. 1, pp. 23-29, 1977.

[13] H. Samet, *The design and analysis of spatial data structures*. Addison-Wesley, Reading, MA, 1989.

[14] Mount, Arya, and Narayan, "Accounting for boundary effects in nearest-neighbor searching," *Discrete & Computational Geometry*, vol. 16, no. 2, pp. 155-176, 1996.

[15] D. Dobkin and R. Lipton, "Multidimensional Searching Problems," *SIAM J. Comput*, vol. 5, no. 2, pp. 181-186, 1976.

[16] T. Bozkaya and M. Ozsoyoglu, "Distance-based indexing for high-dimensional metric spaces," in *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, New York, USA, 1997, pp. 357-368.

[17] P. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," in *SODA '93 Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, Philadelphia, USA, 1993, pp. 311-321.

[18] S. Arya and D. Mount, "Approximate nearest neighbor queries in fixed dimensions," in *SODA '93 Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, Philadelphia, PA, USA, 1993, pp. 271-280.

[19] J. Kleinberg, "Two algorithms for nearest-neighbor search in high dimensions," in *STOC '97 Proceedings of the twenty-ninth annual ACM symposium on Theory of computing* , New York, USA, 1997, pp. 599-608.

[20] Motwani, P. Indyk, and Rajeev, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *STOC '98 Proceedings of the thirtieth annual ACM symposium on Theory of computing*, New York, USA, 1998, pp. 604-613.

[21] E. Kushilevitz, R. Ostrovsky, and Y. Rabani, "Efficient search for approximate nearest neighbor in high dimensional spaces," in *STOC '98 Proceedings of the thirtieth annual ACM symposium on Theory of computing*, New York, NY, USA, 1998, pp. 614-623.

[22] A. Gionis, P. Indyk, and R. Motwani, "Similarity Search in High Dimensions via Hashing," in *VLDB '99 Proceedings of the 25th International Conference on Very Large Data Bases*, San Francisco, USA, 1999, pp. 518-529.

[23] A. Andoni and P. Indyk, "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions," in *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, Berkeley, California , 2006, pp. 459-468.

[24] O. Beaumont, A.-M. Kermarrec, and É. Rivière, "Peer to peer multidimensional overlays: approximating complex structures," in *Proceedings of the 11th international conference on Principles of distributed systems*, Berlin, Heidelberg, 2007, pp. 315-328.

[25] O. Beaumont, A.-M. Kermarrec, L. Marchal, and E. Riviere, "VoroNet: A scalable object network based on Voronoi tessellations," in *International Parallel and Distributed Processing Symposium*, Long Beach, USA , 2007, p. 20.

[26] G. Navarro, "Searching in metric spaces by spatial approximation," in *String Processing and Information Retrieval Symposium*, Cancun, Mexico, 1999, pp. 141-148.

[27] Krylov, Logvinov, Ponomarenko, and Ponomarev, "Metrized Small World Properties Data Structure," in *SEDE*, Los Angeles, California USA, 2008.

[28] V. Krylov, A. Ponomarenko, A. Logvinov, and D. Ponomarev, "Single-attribute Distributed Metrized Small World Data Structure‖," in *IEEE International Conference on Intelligent Computing and Intelligent Systems (CAS)*, 2009.