A Model of Optimal Network Structure for Decentralized Nearest Neighbor Search

Alexander Ponomarenko, Irina Utkina and Mikhail Batsyn

Abstract One of the approaches for the nearest neighbor search problem is to build a network, which nodes correspond to the given set of indexed objects. In this case, the search of the closest object can be thought as a search of a node in a network. A procedure in a network is called decentralized, if it uses only local information about visited nodes and its neighbors. Networks, which structure allows efficient performing the nearest neighbor search by a decentralized search procedure started from any node, is of particular interest, especially for pure distributed systems. Several algorithms that construct such networks have been proposed in literature. However, the following questions arise: "Are there network models in which decentralized search can be performed faster?"; "What are the optimal networks for the decentralized search?"; "What are their properties?" In this paper, we partially give answers to these questions. We propose a mathematical programming model for the problem of determining an optimal network structure for decentralized nearest neighbor search. We have found an exact solution for a regular lattice of size 4×4 and heuristic solutions for sizes from 5×5 to 7×7. As a distance function, we use L_1, L_2 and L_∞ metrics. We hope that our results and the proposed model will initiate study of optimal network structures for decentralized nearest neighbor search.

Keywords Nearest neighbour search · Optimisation model · Network decentralised search

A. Ponomarenko (✉) · I. Utkina · M. Batsyn
Higher School of Economics, National Research University, Moscow, Russia
e-mail: aponomarenko@hse.ru; aponom84@gmail.com

I. Utkina
e-mail: iutkina@hse.ru

M. Batsyn
e-mail: mbatsyn@hse.ru

V. A. Kalyagin et al. (eds.), *Computational Aspects and Applications in Large-Scale Networks*, Springer Proceedings in Mathematics & Statistics 247,
https://doi.org/10.1007/978-3-319-96247-4_14

1 Introduction

The nearest neighbor search appears in many fields of computer science. A problem of building data structure for the nearest neighbor search is formulated as follows. Let D be a domain and $d : D \times D \rightarrow R_{[0;+\infty)}$ be a distance function. One needs to preprocess a finite set $X \subseteq D$ so that the search of the closest object for any given query $q \in D$ in the set X will be as fast as possible. A huge number of methods have been proposed. Of particular interest is the case when the search of nearest neighbor should run in a distributed environment without any central coordination point. For this case, a natural approach for organizing nearest neighbor search is to build a network, which nodes correspond to the given set X. In this case, the search of the closest object can be thought as a search of a node in a network. Moreover, a distributed environment, especially for p2p case, requires that all procedures that are involved in the search or indexing processes should be decentralized. This means that all procedures have only local information about visited nodes and its neighbors and do not have access to the information about the whole structure of the network.

As a rule, such an approach implies searching via *greedy walk* algorithm [1–4] or its modification [5, 6]. So, many p2p systems including DHT protocols [7–9] use the same search algorithm, but employ different distance functions and have different network structures.

In the present paper, we address the problem of optimal network structure for NNS. We emphasize that for any fixed input set, there exists an optimal network structure with respect to the chosen search algorithm. To study the properties of such networks, we present a mathematical Boolean nonlinear programming model of optimal network structure. The objective is to minimize the expected number distance computations made by the greedy walk algorithm to find the nearest neighbor for an arbitrary query starting from an arbitrary node.

As a first step, we solve this problem for the case when the input set X corresponds to the set of nodes of a two-dimensional regular lattice. We have found an exact solution for size 4×4 and heuristic solutions for sizes from 5×5 to 7×7. As a distance function we use L_1, L_2 and L_∞ metrics.

2 Mathematical Formulation

We consider a network as a graph $G(V, E)$ with vertex set $V = X = \{1, \ldots, n\}$ and edge set $E \subset V \times V$. Let $d(i, q)$ be a distance function between vertex i and query q. The neighborhood of vertex i is defined as $N(i) = \{j \in V : (i, j) \in E\}$. We denote the probability function for a query as f_q for a discrete domain and as $f(q)$ the probability density function for a query in continuous domain.

2.1 Decentralized Search Algorithm—Greedy Walk

The goal of the search algorithm is to find the vertex (target vertex) in the graph G which is the closest to the query, going from one vertex to another through the set of edges E of G. The search is based on the information related to the vertices. During the search process, the algorithm can calculate the distance between the query and the vertices which it knows. Below is the pseudo code of the greedy walk algorithm.

```
GreedyWalk(s∈V, q∈V)// s -starting vertex, q - query
  c ← argmin_{y∈N(s)} (d(y,q))
  if d(c,q) < d(s,q) then
     return GreedyWalk(c, q)
  else
     return s
```

Starting from vertex s the algorithm calculates the value of the distance function $d(y, q)$ between query q and every neighbor y of s. After that, the algorithm is recursively called for vertex c closest to the q. The algorithm stops at the vertex, which neighborhood contains no vertices closer to the query than itself. The greedy walk algorithm can be also considered as a process of routing a search message in a network. At each step, the node (vertex) which has received a message (message holder) passes it to the neighbor closest to the query according to the function d.

2.2 Mathematical Programming Model

By no means, all graphs have proper structure for searching via greedy walk. In our model, we require from the structure of graph G that search of any vertex by the greedy walk will reach the target vertex starting from an arbitrary vertex. In general, this requires that the graph needs to have the Delone graph as a subgraph. Similar to the Kleinberg model [1], in this paper, we consider a particular case when vertices are nodes of a regular lattice with integer coordinates. In this case, the Delone graph is just the set of the edges of the regular lattice.

The complexity of the search algorithm is measured as the number of different vertices for which the distance to the query has been calculated. We take this number as an objective function. Equations (1–9) define Boolean nonlinear programming formulation for optimal graph structure.

Decision variables

$$x_{ij} = \begin{cases} 1, \text{ if edge } (i,\ j) \text{ belongs to the solution} \\ 0, \text{ otherwise} \end{cases} \quad (1)$$

$$y_{ij}^k = \begin{cases} 1, \text{ if vertex } k \text{ belongs to the greedy walk from } i \text{ to } j \\ 0, \text{ otherwise} \end{cases} \tag{2}$$

Objective function

$$\min \frac{1}{n} \sum_{i=1}^{n} \sum_{q \in D} O(i, j_q) f_q \text{ (discrete domain)} \tag{3a}$$

$$\min \frac{1}{n} \sum_{i=1}^{n} \int_D O(i, j_q) f(q) dq, \text{ (continuous domain)}, \tag{3b}$$

where

$$j_q = \arg \min_{j=\overline{1,n}} d(j, q) \tag{4}$$

$$O(i, j_q) = \left| \left\{ l \in V : \exists k \, x_{lk} = 1 \text{ and } y_{ij_q}^k = 1 \right\} \right| \tag{5}$$

Constraints

$$x_{ii} = 0 \quad \forall i \in V \tag{6}$$

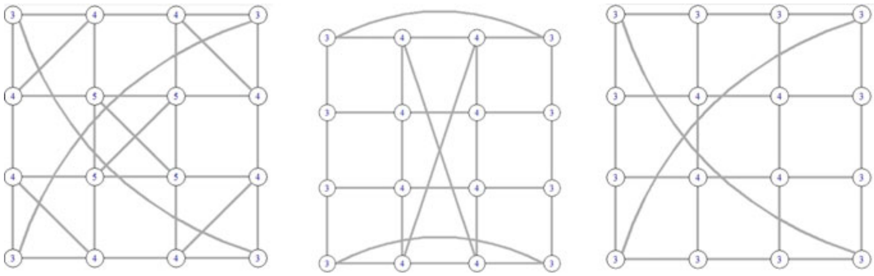
$$y_{ij}^i = y_{ij_q}^j = 1 \quad \forall i, j_q \in V \tag{7}$$

$$\sum_{k=1}^{n} x_{lk} y_{ij_q}^k \geq y_{ij_q}^l \quad \forall i, j_q, l \in V \tag{8}$$

$$l^* = \arg \min_{l \in V:\, x_{kl}=1} (d(l, q)) \Rightarrow y_{ij_q}^{l^*} \geq y_{ij_q}^k \quad \forall q \in D \; \forall i, k \in V \tag{9}$$

Decision variables x_{ij} (1) determine the adjacency matrix of the optimal graph, which we want to find. Indicator variables y_{ij}^k (2) are used to calculate the number of the operations $O(i, j_q)$ performed during the search process from vertex i to vertex j_q, which is the closest vertex (target vertex) to the query q (4). In our case, it is the number of different vertices for which the distance to the query has been calculated. This is equal to the cardinality of the union set of neighborhoods of vertices k for which $y_{ij}^k = 1$ (5).

Since we want to find the optimal graph in general case (for any starting vertex and any query), our objective is to minimize the average number of operations required for the search algorithm to reach a target vertex (3a, 3b, 4, 5). Constraint (6) guarantees that there are no loops in the graph and constraint (7) requires greedy walk (i, j) to start from vertex i and stop at vertex j. Constraint (8) links variables x_{ij} and y_{ij}^k and requires that the search algorithm (the greedy walk) will go through one of vertex l neighbors, if it goes through this vertex l. Constraint (9) describes the greedy strategy of the greedy walk algorithm: if vertex k belongs to the greedy walk from vertex i to



(a) L_2, $f \approx 7.093$ (b) L_1, $f \approx 7.039$ (c) L_∞, $f \approx 7.203$

Fig. 1 Exact solutions found by our branch-and-bound algorithm for regular lattice 4×4

vertex j_q ($y^k_{ij_q} = 1$) then its neighbor l^*, closest to the query q among all its neighbors l, should also belong to this greedy walk ($y^{l^*}_{ij_q} = 1$).

The presented model is applicable for an arbitrary metric space. In the next section, we present the results for a particular case when vertices are the nodes of a two-dimensional regular lattice and the distance functions are L_1, L_2, or L_∞.

3 Computational Experiments and Results

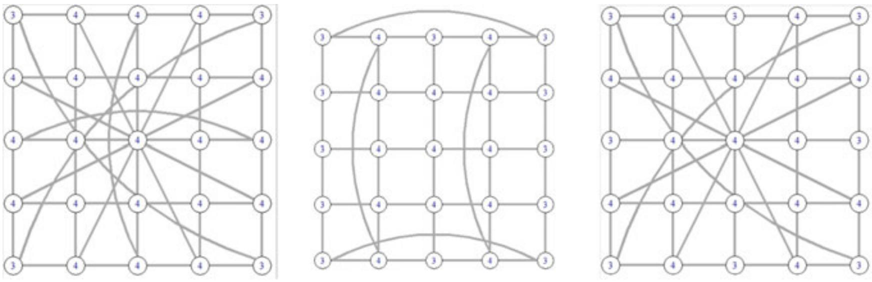
In this work, we suppose that the input set corresponds to the nodes of a two-dimensional regular lattice, and we have a domain such that all nodes have the same probability to be the nearest neighbor for a query. In this case, the nearest neighbor search can be thought as a node discovery procedure, which means that we need to find the given node in the network.

Obviously, we can find the optimal graph structure, if we check all possible configurations of the set of edges. However, the number of all possible configurations grows as $2^{n(n-1)/2}$.

To find an exact solution, we have implemented a branch-and-bound algorithm. The exact solutions found by algorithm for regular lattice 4×4 are presented in Fig. 1. The solutions found by our heuristic are presented in Figs. 2, 3, and 4.

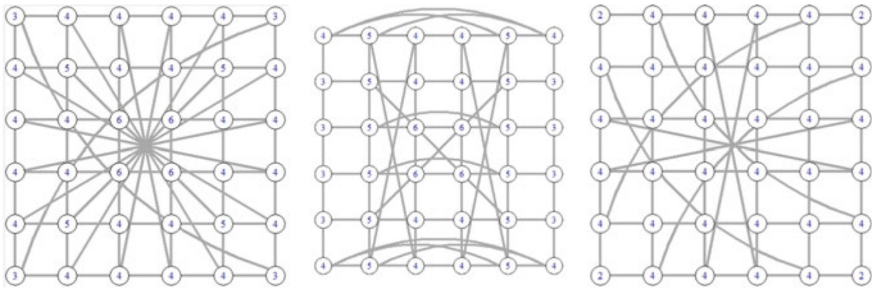
4 Conclusion and Future Work

We have proposed a Boolean nonlinear programming model to determine an optimal graph structure, which minimizes the complexity of the nearest neighbor search by the greedy walk algorithm. We have found an exact solution for a regular lattice of size 4×4, and presented the results found by our heuristic for sizes from 5×5 to 7×7 with the three most popular distances: L_1, L_2, and L_∞.



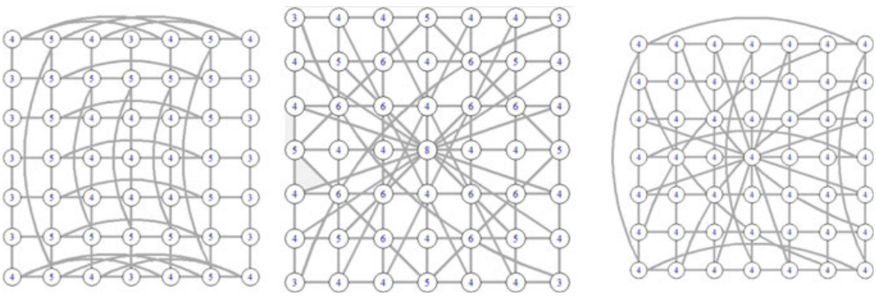
(a) L_2, $f \approx 8.974$ (b) L_1, $f = 8.784$ (c) L_∞, $f \approx 9.036$

Fig. 2 Solutions found by our heuristic for regular lattice 5×5



(a) L_2, $f \approx 10.509$ (b) L_1, $f \approx 10.485$ (c) L_∞, $f \approx 10.756$

Fig. 3 Solutions founded by heuristic for a regular lattice 6×6



(a) L_1, $f \approx 12.054$ (b) L_2, $f \approx 12.136$ (c) L_∞, $f \approx 12.328$

Fig. 4 Solutions found by our heuristic for regular lattice 7×7

However, we realize that the most important characteristic which should be studied is the asymptotical behavior of the objective function. Therefore, our future work will be focused on improving the efficiency of our exact and heuristic algorithms. We also have plans to develop models describing optimal network structures for approximate nearest neighbor search. We hope that this work will draw attention to the study of graph structures optimal for decentralized nearest neighbor search.

Acknowledgements This research is conducted in LATNA Laboratory, National Research University Higher School of Economics and supported by RSF grant 14-41-00039.

References

1. Kleinberg, J.: The small-world phenomenon: an algorithmic perspective. In: Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, pp. 163–170. ACM, May 2000
2. Beaumont, O., Kermarrec, A.-M., Marchal, L., Riviere, E.: VoroNet: a scalable object network based on Voronoi tessellations. In: Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International, pp. 1–10. IEEE (2007)
3. Malkov, Y., Ponomarenko, A., Logvinov, A., Krylov, V.: Scalable distributed algorithm for approximate nearest neighbor search problem in high dimensional general metric spaces. In: Similarity Search and Applications, pp. 132–147. Springer, Berlin, Heidelberg (2012)
4. Beaumont, O., Kermarrec, A.M., Rivière, É.: Peer to peer multidimensional overlays: approximating complex structures. In: Principles of Distributed Systems, pp. 315–328. Springer, Berlin, Heidelberg (2007)
5. Malkov, Y., Ponomarenko, A., Logvinov, A., Krylov, V.: Approximate nearest neighbor algorithm based on navigable small world graphs. Inf. Syst. **45**, 61–68 (2014)
6. Ruiz, G., Chávez, E., Graff, M., Téllez, E.S.: Finding near neighbors through local search. In: Similarity Search and Applications, pp. 103–109. Springer International Publishing (2015)
7. Maymounkov, P., David, M.: Kademlia: a peer-to-peer information system based on the xor metric. In: Peer-to-Peer Systems, pp. 53–65. Springer, Berlin, Heidelberg (2002)
8. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM Comput. Commun. Rev. **31**(4), 149–160 (2001)
9. Rowstron, A., Druschel, P.: Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Middleware 2001, pp. 329–350. Springer, Berlin, Heidelberg, Nov 2001