JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 24 No. 10, pp. 79-89, October 2019
https://doi.org/10.9708/jksci.2019.24.10.079

Efficient Processing of *k*-Farthest Neighbor Queries for Road Networks

Taelee Kim*, Hyung-Ju Cho*, Hee Ju Hong**, Hyogeun Nam**,
Hyejun Cho**, Gyung Yoon Do**, Pilkyu Jeon**

*Student, Department of Software, Kyungpook National University, Sangju, Korea
*Professor, Department of Software, Kyungpook National University, Sangju, Korea
**Student, Daegu Science High School, Daegu, Korea

[Abstract]

While most research focuses on the k-nearest neighbors (kNN) queries in the database community, an important type of proximity queries called k-farthest neighbors (kFN) queries has not received much attention. This paper addresses the problem of finding the k-farthest neighbors in road networks. Given a positive integer k, a query object q, and a set of data points P, a kFN query returns k data objects farthest from the query object q. Little attention has been paid to processing kFN queries in road networks. The challenge of processing kFN queries in road networks is reducing the number of network distance computations, which is the most prominent difference between a road network and a Euclidean space. In this study, we propose an efficient algorithm called FANS for k-FArthest Neighbor Search in road networks. We present a shared computation strategy to avoid redundant computation of the distances between a query object and data objects. We also present effective pruning techniques based on the maximum distance from a query object to data segments. Finally, we demonstrate the efficiency and scalability of our proposed solution with extensive experiments using real-world roadmaps.

▸ **Key words**: *k*-Farthest neighbor query, Road network, Minimum distance, Maximum distance

[요 약]

본 연구에서는 도로 네트워크에서 k-최원접 이웃 검색을 위한 효율적인 FANS(k-FArthest Neighbor Search) 알고리즘을 제안한다. 양의 정수 k, 질의 객체 q, 일련의 데이터 객체 집합 P가 주어지면, k-최원접 이웃 질의는 질의 객체 q에서 가장 멀리 있는 k개의 데이터 객체를 찾는다. 데이터베이스 분야에서 대부분의 연구는 k-최근접 이웃 질의에 중점을 두고 있어서, k-최원접 이웃 질의라는 중요한 근접 질의 유형은 별다른 관심을 받지 못했다. 이 논문에서는 도로 네트워크에서 가장 멀리 있는 이웃을 찾는 문제를 다룬다. 도로 네트워크에서 k-최원접 이웃 질의를 처리하는 연구는 거의 없었다. 도로 네트워크에서 k-최원접 이웃 질의를 처리해야 하는 문제는 최단 경로 거리를 계산하는 횟수를 줄이는 것인데, 이는 도로 네트워크와 유클리드 공간의 질의 처리에서 가장 중요한 차이다. 질의 객체와 데이터 객체 사이의 최단 경로 거리에 대한 중복 계산을 줄이기 위하여 공유 계산 전략을 사용한다. 질의 객체에서 데이터 세그먼트까지 최대 거리를 기반으로 효과적으로 후보군을 제거하는 방법은 제시한다. 마지막으로 실제 도로 지도를 사용한 광범위한 실험을 통해 제시된 방법의 효율성과 확장성을 보여준다.

▸ **주제어**: k-최원접 이웃 검색, 도로 네트워크, 최소 거리, 최대 거리

• First Author: Taelee Kim, Corresponding Author: Hyung-Ju Cho
*Taelee Kim (rhlxp0602@naver.com), Dept. of Software, Kyungpook National University
*Hyung-Ju Cho (hyungju@knu.ac.kr), Dept. of Software, Kyungpook National University
**Hee Ju Hong (heejujeenakit@gmail.com), Daegu Science High School
**Hyogeun Nam (rms0605@naver.com), Daegu Science High School
**Hyejun Cho (reginajoe@naver.com), Daegu Science High School
**Gyung Yoon Do (ehruddbs1024@naver.com), Daegu Science High School
**Pilkyu Jeon (jkjpk0620@gmail.com), Daegu Science High School
• Received: 2019. 09. 11, Revised: 2019. 09. 27, Accepted: 2019. 10. 07.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

본 논문에서는 도로 네트워크에서 k-최원접 이웃 질의를 효율적으로 검색하는 방법을 제시한다. k-최원접 이웃 검색은 k-최근접 이웃 검색과 논리적으로 반대가 되는 연산이다[1-6]. 사람이나 자동차와 같은 운송수단은 도로 위를 움직이는데, 기존 연구들은 최단 경로 거리 대신에, 직선거리를 이용하여 공간 질의 처리에 집중하고 있다. 직선거리를 사용하는 기존 방법들은 최단 경로 거리를 사용하는 공간 질의 처리 문제를 해결할 수 없다. 본 연구에서는 도로 네트워크에서의 최단 경로 거리를 기반으로 k-최원접 이웃 질의 문제를 다룬다. k-최원접 이웃 질의 문제는 계산 기하학, 인공 지능, 패턴 인식 및 정보 검색과 같은 많은 실세계 문제와 관련되어 있다. 예를 들어, 최원접 이웃 질의는 질의 객체 q를 중심으로 모든 데이터 객체를 포함하는 원의 최소 반지름을 구하는데 사용될 수 있다. 또는 특공대 팀이 임무를 수행할 때, 팀장은 자신을 중심으로 모든 팀원이 1km 거리 안에서 임무를 수행하기를 원한다고 가정하자. 만약 k-최원접 이웃 질의를 사용하면, 팀장은 팀원들의 위치를 계속 확인하며 가장 멀리 떨어져 있는 팀원에게 멀리 이동하지 않도록 주의하면서 임무를 수행할 수 있다.

Fig. 1. Difference between kNN and kFN Queries

그림 1과 같이 정수 $k = 2$, 질의 객체 q, 그리고 데이터 객체 집합 $P = \{p_1, p_2, p_3, p_4, p_5\}$이 주어져 있다고 가정하자. k-최근접 이웃 질의는 질의 객체 q에서 가장 가까운 두 개의 데이터 객체 p_1과 p_2를 찾는다. 반면에, k-최원접 이웃 질의는 질의 객체 q에서 가장 먼 곳에 있는 두 개의 데이터 객체 p_4과 p_5를 찾는다. 현재까지 도로 네트워크에서 움직이는 데이터 객체를 고려한 k-최원접 이웃 질의를 효율적으로 계산하는 방법에 관한 연구는 수행되지 않았다. 도로 네트워크에서 k-최원접 이웃 질의를 처리하기 위한 간단한 방법은 질의 객체 q에서 각각의 데이터 객체까지 최단 경로 거리를 계산한 후에, 이들 중에서 가장 멀리 떨어진 k개의 데이터 객체를 찾는 것이다. 이 방법은 q에서 가장 먼 곳에 있는 k개의 데이터 객체를 찾기 위하여, q에서 최단 경로 거리를 기반으로 모든 데이터 객체를 정렬하는 연산이 필요하다. 이 방법은 간단하지만 불필요한 최단 경로 거리 계산을 요구하기 때문에 매우 비효율적이고, 데이터 객체들이 많은 경우에 사용할 수가 없다. k-최원접 이웃 검색 방법은 k-최근접 이웃 검색 방법과 원칙적으로 추구하는 목적이 다르다. 따라서, 본 연구에서는 도로 네트워크에서 효율적인 k-최원접 이웃 검색(k-FArthest Neighbor Search)을 위한 FANS 알고리즘을 제안한다. FANS 알고리즘은 인접한 데이터 객체들을 그룹화하고, 질의 객체 q에서 각 그룹까지의 최대 거리를 계산하여 불필요한 후보 데이터 객체들을 빠르게 제거함으로써, 불필요한 최단 경로 거리 계산을 줄일 수 있다. 본 논문에서는 질의 객체와 데이터 객체들이 도로를 따라서 자유롭게 움직인다고 가정하기 때문에, 질의 객체와 데이터 객체에 대한 사전 거리 계산 방식을 허용하지 않는다. 다음은 본 논문의 성과를 요약한 것이다.

- 본 논문에서는 도로 네트워크에서 k-최원접 이웃 검색(k-Farthest Neighbor Search)을 위한 효율적인 FANS 알고리즘을 제안한다.
- 인접한 데이터 객체들을 그룹화해서 처리하는 공유 연산 방식을 사용하여, 정답이 될 수 없는 데이터 객체에 대한 최단 경로 거리 계산을 허용하지 않는다. 또한, 질의 객체에서 데이터 그룹까지 최대 거리를 계산하고, 정답이 될 수 없는 데이터 객체들을 빠르게 제거하는 방법을 제시한다.
- FANS 알고리즘의 효율성을 검증하기 위하여 실제 지도 데이터를 사용해서 다양한 조건에서 비교 실험을 수행한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 조사한다. 3장에서는 사전지식을 소개하고 k-최원접 이웃 질의를 정의한다. 4장에서는 데이터 객체를 그룹화하는 방법과 최소 거리와 최대 거리 계산 방법을 설명한다. 5장에서는 도로 네트워크에서 k-최원접 이웃 질의를 효율적으로 처리하기 위한 FANS 알고리즘을 제안한다. 6장에서는 기존 알고리즘과 FANS 알고리즘을 다양한 조건에서 비교 실험한 결과를 보여준다. 7장에서는 본 논문의 결론을 제시한다.

II. Related Works

1. k-nearest neighbor queries

도로 네트워크에서의 k-최근접 이웃 질의는 광범위하게 연구되었다. Papadias *et al.*은 IER(Incremental

Euclidean Restriction) 기법과 INE(Incremental Network Expansion) 기법을 개발했다[7]. IER 기법은 도로 네트워크에서 두 지점 사이의 최단 경로 거리는 직선거리보다 작지 않다는 원리를 활용하였다. INE 기법은 질의 위치에서 Dijkstra 알고리즘과 유사한 방식으로 네트워크 확장을 수행하고, 데이터 객체를 발견하는 순서에 따라 조사한다. Shahabi *et al.*은 기존의 유클리드 기반 알고리즘을 사용하여 가장 가까운 객체를 근사적으로 검색하기 위해 도로 네트워크를 제약이 없는 고차원의 유클리드 공간으로 변환하는 내장 기법을 개발하였다[8]. Kolahdouzan and Shahabi는 도로 네트워크에 있는 데이터 객체를 중심으로 네트워크 보로노이 다각형(Network Voronoi Polygons, NVP)을 만들기 위하여, 1차 네트워크 보로노이 다이어그램을 사용한다[9]. 이 방식은 도로 네트워크에 있는 데이터 객체에 대한 검색 문제를 유클리드 공간에 있는 데이터 객체 문제로 변환하고, NVP를 색인하였다. 여기에서 NVP를 미리 계산하여 온라인 네트워크 거리 연산을 최소화한다. Huang *et al.*은 k-최근접 이웃 질의 검색 문제를 해결하기 위하여, 모든 데이터 객체들을 주요한 정점들과 연결하여 질의 처리를 수행하는 섬(Island) 접근 방법을 사용하였다[10]. Samet *et al.*은 특정 간선을 포함하는 최단 경로를 가진 모든 정점을 빠르게 탐색하기 위하여 해당 간선에 이름을 부여하는 거리 탐색(Distance Browsing, DisBrw)방법을 제안했다[11]. DisBrw 기법은 두 객체 간의 최단 경로를 찾기 위하여, 최단 경로 쿼드 트리(shortest-path quadtree)를 탐색하며, 간선의 이름을 사용하여 불필요한 간선들을 탐색하지 않는다.

Lee *et al.*은 움직이는 공간 질의 객체를 위하여 ROAD 시스템을 개발했다[12]. ROAD 시스템은 상호 연결된 지역의 서브 네트워크를 계층화하여 거대한 도로 네트워크를 구성하며, 각각의 서브 네트워크는 네트워크 탐색 가속화와 신속한 객체 검색을 위해 최단 경로와 객체 추상화 정보를 추가한다. Zhong *et al.*은 R-tree[13]의 개념을 활용하여, 도로 네트워크에서 사용할 수 있는 높이 균형 색인 방법으로 G-tree를 개발했다[14]. G-tree는 그래프 분할을 사용하여 주어진 도로 네트워크를 재귀적으로 분할하고, 분할된 그래프 계층을 사용하여 최단 경로 거리를 빠르게 계산한다. Abeywickrama *et al.*은 도로 네트워크에서 최단 경로 거리 계산과 k-최근접 이웃 질의 처리 알고리즘들에 대한 비교 실험 평가를 수행하였다[15]. 실험 결과에서는 G-tree[14]가 INE[7], DisBrw[11], ROAD[12] 보다 우수한 성능을 보여주었다. 최근접 이웃 검색과 최원접 이웃 검색은 찾고자 하는 데이터 객체의 조건이 달라서, 최근접 이웃 검색을 위한 기법들은 최원접 이웃 검색 문제 해결에 사용할 수 없다. 즉, 최근접 이웃 검색은 질의 객체에서 가장 가까운 데이터 객체를 찾는 것이고, 최원접 이웃 검색은 질의 객체에서 가장 먼 곳에 있는 데이터 객체를 찾는 것이다.

2. k-farthest neighbor queries

유클리드 공간에서 최원접 이웃 검색을 이용한 공간 질의 처리에 관한 많은 연구가 수행되었다. Curtin *et al.*은 유클리드 공간에 분포한 데이터를 사용하여 후보 객체 집합을 선택하는 근사적인 k-최원접 이웃 검색 알고리즘과 최원접 이웃 검색 문제의 난이도를 평가하기 위한 정보 이론 엔트로피 방법을 개발했다[1]. 또한, 최근에는 유클리드 공간과 도로 네트워크에서 집합적 k-최원접 이웃 질의 문제를 연구했다[6, 16]. 객체 집합 P와 질의 집합 Q가 주어졌을 때, 집합적 k-최원접 이웃 질의는 Q의 모든 질의 지점까지의 거리의 합이 가장 크게 되는 데이터 객체들을 집합 P에서 검색한다. 역 최원접 이웃 질의 또한 유클리드 공간과 도로 네트워크에서 연구되었다. Yao *et al.*은 유클리드 공간에서 역 최원접 이웃 질의 문제를 처음으로 연구했다[2]. 역 최원접 이웃 검색을 효율적으로 처리하기 위하여 R-tree[13]를 사용한 점진적 최원접 셀과 convex hull을 사용한 최원접 셀 알고리즘을 제안했다. Wang *et al.*은 유클리드 공간에서 임의의 k 값에 대한 역 k-최원접 이웃 질의 처리 방법을 개발했다[17]. Xu *et al.*은 랜드마크와 분할 기법을 사용하여 도로 네트워크에서 일원적(monochromatic) 역최원접 이웃 질의와 이원적(bichromatic) 역최원접 이웃 질의에 대한 효율적인 알고리즘을 개발했다[3]. 유클리드 공간의 최원접 이웃 질의 처리에 사용되는 기법들(예: R-tree와 convex hull)은 최단 경로 거리를 사용하는 도로 네트워크의 최원접 이웃 질의에 쉽게 적용할 수 없다.

본 논문에서 제안된 방법은 다음과 같은 이유로 기존 연구와 차별성을 가진다. 첫째, 본 연구는 도로 네트워크에서 k-최원접 이웃 질의를 효율적으로 처리하기 위한 첫 번째 시도이다. 둘째, 본 연구에서는 질의 객체와 데이터 객체가 자유롭게 도로 네트워크를 따라서 움직인다고 가정하기 때문에, 질의 객체와 데이터 객체 사이의 사전 거리 계산 방법을 사용하지 않는다. 셋째, 제안된 알고리즘은 기존의 최단 경로 거리 계산 방법[12, 14, 18]을 사용하여 쉽게 구현할 수 있다. 이것은 실무에서 매우 유용한 특성이다.

III. Preliminaries

정의 1. (*k*-최원접 이웃 질의) 양수 정수 k, 질의 객체 q, 그리고 데이터 객체 집합 P가 주어졌을 때, k-최원접 이웃 질의는 P에 속하는 데이터 객체들에서 질의 객체 q로부터 가장 먼 곳에 있는 k개의 데이터 객체들의 집합 P_k를 찾는다.

정의 2. (도로 네트워크) 도로 네트워크는 가중치가 있는 비방향성 그래프 $G=\langle V,E,W\rangle$로 나타낸다. 이때, V는 정점 집합, E는 간선 집합, W는 거리 행렬을 나타낸다. 간선은 이동 시간과 같은 네트워크 거리를 의미하고, 음수가 아닌 가중치를 갖는다.

정의 3. (교차 정점, 중간 정점, 말단 정점) 정점들은 차수에 따라 세 가지 분류한다. 정점의 차수가 3보다 크거나 같으면, 교차 정점(intersection vertex)이라고 한다. 정점의 차수가 2이면, 중간 정점(intermediate vertex)이라고 한다. 정점의 차수가 1이면, 말단 정점(terminal vertex)이라고 한다.

정의 4. (정점 시퀀스, 데이터 세그먼트) 정점 시퀀스 $\overline{v_l v_{l+1}\cdots v_m}$는 두 정점 v_l과 v_m을 연결하는 경로이다. 여기에서, v_l과 v_m은 교차 정점이거나 말단 정점이다. 나머지 다른 정점들 $v_{l+1},\cdots,v_{m-1}$은 중간 정점이다. 정점 시퀀스의 길이는 정점 시퀀스에 있는 모든 간선들의 가중치들의 합이 된다. 본 논문에서 정점 시퀀스에 있는 데이터 객체들을 연결한 선분을 데이터 세그먼트라고 부른다.

Table 1. Symbols and Their Meanings

Symbol	Definition
k	Number of requested FNs
q	Query point
P_k	Set of k data points farthest from a query point q
$dist(p_1,p_2)$	Length of the shortest path connecting two points p_1 and p_2 in the road network
$len(p_1,p_2)$	Length of the segment connecting two points p_1 and p_2, such that both p_1 and p_2 are located in the same vertex sequence
$\overline{v_l v_{l+1}\cdots v_m}$	Vertex sequence where v_l and v_m are not intermediate vertices and the other vertices, $v_{l+1},\cdots,v_{m-1}$, are intermediate vertices
$\overline{p_i p_{i+1}\cdots p_j}$	Data segment that consists of data points $p_i,p_{i+1},\cdots,p_j$ in a vertex sequence
$mindist(q,\overline{p_i p_j})$	Minimum distance from a query point q to a data segment $\overline{p_i p_j}$
$maxdist(q,\overline{p_i p_j})$	Maximum distance from a query point q to a data segment $\overline{p_i p_j}$

본 논문에서 사용되는 기호들에 대하여 표 1에서 설명한다. 그림 2는 도로 네트워크의 데이터 객체 p_1과 p_2 사이의 최단 경로 거리와 세그먼트 길이의 차이를 설명한다. 간선의 숫자는 인접한 지점 간의 거리를 의미한다(예: $dist(v_1,v_2)=6$). p_1에서 p_2까지 최단 경로는 $p_1\rightarrow v_2\rightarrow v_5\rightarrow p_2$이며, p_1에서 p_2까지 거리는 $dist(p_1,p_2)=6$이다. 정점 시퀀스 $\overline{v_2v_3v_4v_5}$에서 p_1과 p_2를 연결하는 데이터 세그먼트는 $\overline{p_1v_3v_4p_2}$이며, 이 데이터 세그먼트의 길이는 $len(p_1,p_2)=9$이다. 데이터 객체 p_1과 p_2가 같은 정점 시퀀스에 위치할 때만, 데이터 세그먼트 길이 $len(p_1,p_2)$을 정의할 수 있다.

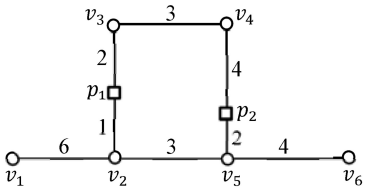


Fig. 2. Example where $dist(p_1,p_2)=6$ and $len(p_1,p_2)=9$

IV. Processing kFN queries in Road Networks

1. Grouping adjacent data objects

그림 3은 도로 네트워크에서 k-최원접 이웃 질의 처리를 위한 예제이다. 본 논문에서는 이 예제를 사용하여 설명한다. 그림 3에는 p_1에서 p_8까지 총 8개의 데이터 객체가 있다. 데이터 객체 집합 $P=\{p_1,p_2,...,p_8\}$가 주어졌을 때, 질의 객체 q에서 가장 멀리 있는 2개 데이터 객체를 찾는 k-최원접 이웃 질의(k = 2)를 다룬다.

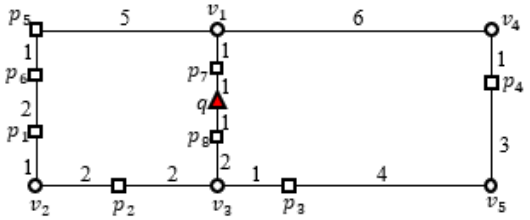


Fig. 3. Example of a kFN Query in a Road Network

그림 4는 정점 시퀀스에서 데이터 객체를 그룹화하는 예를 보여준다. 정점 시퀀스 $\overline{v_1v_2v_3}$에 위치한 4개 데이터 객체들 p_1,p_2,p_5,p_6은 데이터 세그먼트 $\overline{p_2p_1p_6p_5}$로 그룹화되고, 정점 시퀀스 $\overline{v_1v_4v_5v_3}$에 위치한 2개 데이터 객체들 p_3,p_4은 데이터 세그먼트 $\overline{p_3p_4}$로 그룹화된다. 정점 시퀀스 $\overline{v_1v_3}$에 위치한 두 객체 p_7,p_8은 데이터 세그먼트 $\overline{p_7p_8}$로 그

룹화된다. 결과적으로, 데이터 객체 집합 $P=\{p_1,p_2,...,p_8\}$는 데이터 세그먼트 집합 $\overline{P}=\{\overline{p_2p_1p_6p_5},\overline{p_3p_4},\overline{p_7p_8}\}$로 변환된다.

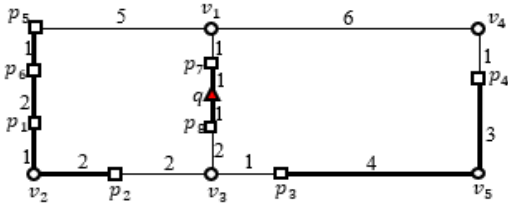


Fig. 4. Grouping Adjacent Data Objects

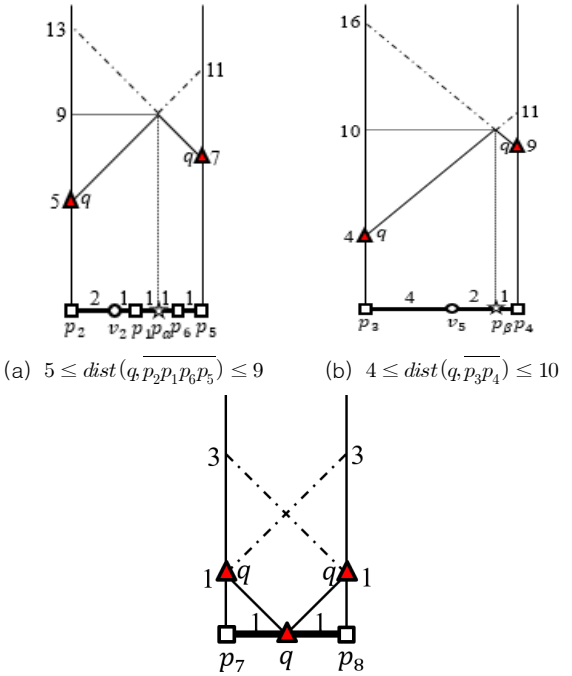
2. Computation of the distance from a query object to a data segment

질의 객체 q에서 데이터 세그먼트 $\overline{p_ip_j}$까지 최소 거리와 최대 거리를 계산하는 방법을 알아보자. 질의 객체 q에서 데이터 세그먼트 $\overline{p_ip_j}$까지 최소 거리와 최대 거리는 각각 $mindist(q,\overline{p_ip_j})$와 $maxdist(q,\overline{p_ip_j})$로 표기한다. $mindist(q,\overline{p_ip_j})$와 $maxdist(q,\overline{p_ip_j})$를 수학적으로 정의하면 다음과 같다. $mindist(q,\overline{p_ip_j}) = min\{dist(q,p)|p\in\overline{p_ip_j}\}$, $maxdist(q,\overline{p_ip_j}) = max\{dist(q,p)|p\in\overline{p_ip_j}\}$이다. $mindist(q,\overline{p_ip_j})$을 계산하는 방법은 다음과 같다. 만일 질의 객체 q가 데이터 세그먼트 $\overline{p_ip_j}$에 속한다면, 질의 객체 q에서 데이터 세그먼트 $\overline{p_ip_j}$까지 최소 거리는 $mindist(q,\overline{p_ip_j}) = 0$이다. 질의 객체 q가 데이터 세그먼트 $\overline{p_ip_j}$에 속하지 않는다면, 최소 거리는 $mindist(q,\overline{p_ip_j}) = min\{dist(q,p_i),dist(q,p_j)\}$이다. 요약하면, $mindist(q,\overline{p_ip_j})$는 다음과 같이 계산된다.

$$mindist(q,\overline{p_ip_j}) = \begin{cases} 0 & \text{if } q \in \overline{p_ip_j} \\ min\{dist(q,p_i),dist(q,p_j)\} & \text{otherwise} \end{cases}$$

질의 객체 q에서 데이터 세그먼트 $\overline{p_ip_j}$까지 최대 거리를 계산하는 방법은 $\overline{p_ip_j}$까지 최소 거리를 계산하는 방법보다 어렵다. $maxdist(q,\overline{p_ip_j})$을 계산하는 방법은 다음과 같다. 먼저, q에서 데이터 객체 p까지 거리 $dist(q,p)$을 계산한다. 만일 질의 객체 q에서 데이터 세그먼트에 있는 한 점 $p\in\overline{p_ip_j}$까지 경로가 $q\rightarrow p_i\rightarrow p$라면, q에서 p까지 거리는 $dist(q,p)=dist(q,p_i)+len(p_i,p)$이다. 비슷한 방법으로, q에서 p까지 경로가 $q\rightarrow p_j\rightarrow p$라면, q에서 p까지 거리는 $dist(q,p)=dist(q,p_j)+len(p_j,p)$이다. 만일 질의 객체 q가 데이터 세그먼트 $\overline{p_ip_j}$에 속한다면, q에서 p까지 거리는 $dist(q,p)=len(q,p)$이다. $dist(q,p)$가 q에서 p까지 최단 경로 거리를 의미하기 때문에, $dist(q,p)$는 다음과 같이 계산된다.

$$dist(q,p) = \begin{cases} min\{dist(q,p_i)+len(p_i,p),dist(q,p_j)+len(p_j,p)\} & \text{if } q \notin \overline{p_ip_j} \\ min\{dist(q,p_i)+len(p_i,p),dist(q,p_j)+len(p_j,p),len(q,p)\} & \text{otherwise} \end{cases}$$



(a) $5 \le dist(q,\overline{p_2p_1p_6p_5}) \le 9$ (b) $4 \le dist(q,\overline{p_3p_4}) \le 10$

(c) $0 \le dist(q,\overline{p_7p_8}) \le 1$

Fig. 5. Computation of the minimum and maximum distances from q to $\overline{p_2p_1p_6p_5}$, $\overline{p_3p_4}$, and $\overline{p_7p_8}$

그림 5는 질의 객체 q에서 3개 데이터 세그먼트 $\overline{p_2p_1p_6p_5}$, $\overline{p_3p_4}$, $\overline{p_7p_8}$까지 최소 거리와 최대 거리를 계산하는 방법을 보여준다. 그림에서 1점 쇄선은 q에서 데이터 세그먼트에 속하는 지점까지 최단 경로가 아닌 경로의 길이를 나타낸다. 그림 5(a)에서 보이는 것처럼, $dist(q,p_2)=5$, $dist(q,p_5)=7$이기 때문에, q에서 $\overline{p_2p_1p_6p_5}$까지 최소 거리와 최대 거리는 $mindist(q,\overline{p_2p_1p_6p_5})=5$, $maxdist(q,\overline{p_2p_1p_6p_5})=9$가 된다. 비슷한 방법으로, 그림 5(b)에서 보이는 것처럼, q에서 $\overline{p_3p_4}$까지 최소 거리와 최대 거리는 $mindist(q,\overline{p_3p_4})=4$, $maxdist(q,\overline{p_3p_4})=10$이다. 마지막으로, 그림 5(c)에서 보이는 것처럼, q에서 $\overline{p_7p_8}$까지 최소 거리와 최대 거리는 $mindist(q,\overline{p_7p_8})=0$, $maxdist(q,\overline{p_7p_8})=1$이다. q에서 $\overline{p_7p_8}$까지 최소 거리가 $mindist(q,\overline{p_7p_8})=0$인 이유는 q가 데이터 세그먼트 $\overline{p_7p_8}$에 포함되기 때문이다. 표 2는 질의 객체 q에서 데이터 세그먼트 $\overline{p_2p_1p_6p_5}$, $\overline{p_3p_4}$, $\overline{p_7p_8}$까지 최소 거리와 최대 거리 계산 결과를 요약한 것이다.

Table 2. $mindist(q,\overline{p_ip_j})$ and $maxdist(q,\overline{p_ip_j})$ for $\overline{p_ip_j}\in\overline{P}$

$\overline{p_ip_j}$	$mindist(q,\overline{p_ip_j})$	$maxdist(q,\overline{p_ip_j})$
$\overline{p_2p_1p_6p_5}$	$mindist(q,\overline{p_2p_1p_6p_5})=5$	$maxdist(q,\overline{p_2p_1p_6p_5})=9$
$\overline{p_3p_4}$	$mindist(q,\overline{p_3p_4})=4$	$maxdist(q,\overline{p_3p_4})=10$
$\overline{p_7p_8}$	$mindist(q,\overline{p_7p_8})=0$	$maxdist(q,\overline{p_7p_8})=1$

3. Sorting data segments based on their maximum distance to query object

그림 6은 $\overline{P}=\{\overline{p_2p_1p_6p_5},\overline{p_3p_4},\overline{p_7p_8}\}$에 속하는 데이터 세그먼트들을 정렬하는 방법을 설명한다. 그림 6에서 $\overline{P}$에 속하는 데이터 세그먼트를 q까지 최대 거리의 내림차순으로 $\overline{p_3p_4}$, $\overline{p_2p_1p_6p_5}$, $\overline{p_7p_8}$ 순서로 정렬하였다. 구체적으로, $\overline{p_3p_4}$, $\overline{p_2p_1p_6p_5}$, $\overline{p_7p_8}$까지 최대 거리는 각각 $maxdist(q,\overline{p_3p_4})=10$, $maxdist(q,\overline{p_2p_1p_6p_5})=9$, $maxdist(q,\overline{p_7p_8})=1$이다. 만일 최대 거리가 같은 데이터 세그먼트가 2개 이상 있는 경우에, q까지 최소 거리를 기준으로 내림차순으로 다시 정렬한다.

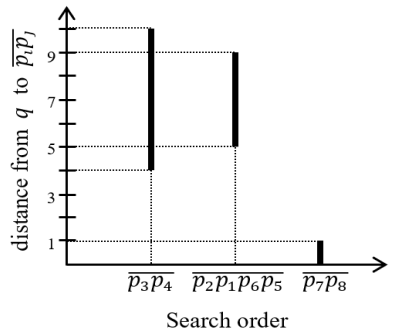


Fig. 6. Search order of data segments in $\overline{P}$ based on their maximum distance to q

V. Farthest neighbors search algorithm for road networks

1. FANS algorithm

FANS 알고리즘은 4단계로 구성된다. 1단계에서는 정점 시퀀스에 있는 인접한 데이터 객체들을 데이터 세그먼트로 그룹화한다(1−2줄). 2단계에서는 q에서 각 데이터 세그먼트까지의 최소 거리와 최대 거리를 계산한다(3−5줄). 3단계에서는 모든 데이터 세그먼트를 q까지의 최대 거리를 기준으로 내림차순 정렬한다(6−7줄). 4단계에서는 정렬된 데이터 세그먼트를 차례로 조사하면서, q에서 가장 멀리 떨어진 k개의 객체를 찾는다(8−18줄). 4장에서 1단계부터 3단계까지 설명했기 때문에, 5장에서는 4단계를 자세하게 설명한다.

질의 객체 q까지 최대 거리가 가장 큰 데이터 세그먼트부터 탐색을 시작한다. 따라서, q까지 최대 거리가 가장 큰 데이터 세그먼트를 가장 먼저 조사하고, q까지의 최대 거리가 가장 작은 데이터 세그먼트를 가장 나중에 조사한다. 변수 $prundist$는 q에서 k번째 최원접 후보 객체까지 거리를 저장하며, 0으로 초기화한다. $prundist$는 FANS 알고리즘의 종료 여부를 결정하기 위한 임계값으로 사용한다(12줄). 만일 $prundist$의 값이 현재 처리 중인 질의 객체 q에서 데이터 세그먼트 $\overline{p_ip_j}$까지 최대 거리보다 크다면, 알고리즘을 종료하고 결과 집합 P_k 를 반환한다. 그렇지 않다면, $\overline{p_ip_j}$에 있는 데이터 객체 p가 k번째 최원접 후보 객체 p_{kth}보다 q에서 먼 곳에 있는지 검사한다. 만일

Algorithm 1: FANS(q,k,P)
Input: q : a query point, k : number of requested FNs, P : set of data points
Output: : set of data points farthest from q
1 //step 1: it groups adjacent data points in a vertex sequence
2 $\overline{P}\leftarrow$group_data_points(P) // adjacent data points in a vertex sequence are grouped into a data segment
3 //step 2: it computes $mindist(q,\overline{p_ip_j})$ and $maxdist(q,\overline{p_ip_j})$ for each data segment $\overline{p_ip_j}$ in $\overline{P}$
4 **for** each data segment $\overline{p_ip_j}\in\overline{P}$ **do**
5 compute $mindist(q,\overline{p_ip_j})$ and $maxdist(q,\overline{p_ip_j})$
6 // step 3: it sorts data segments by a decreasing order of their maximum distance to q
7 $\overline{P}\leftarrow sort\_by\_dec\_order(\overline{P})$
8 // step 4: it explores sorted data segments sequentially to find data points farthest from q
9 $P_k\leftarrow\varnothing$ // P_k keeps a set of k data points farthest from q so far
10 $prundist\leftarrow 0$ // $prundist$ keeps the distance from q to the kth farthest data point so far
11 **for** each data segment $\overline{p_ip_j}\in\overline{P}$ **do**
12 **if** $maxdist(q,\overline{p_ip_j}) < prundist$ **then**
13 **go to** line 19 // the remaining data segments can be safely ignored according to Corollary 1
14 **else** // it indicates that $maxdist(q,p_ip_j)\geq prundist$
15 **for** each data point $p\in\overline{p_ip_j}$ **do**
16 **if** $dist(q,p) > dist(q,p_{kth})$ **then** // it checks whether a data point p in $\overline{p_ip_j}$ is farther from q than p_{kth}
17 $P_k\leftarrow P_k\cup\{p\}-\{p_{kth}\}$
18 $prundist\leftarrow dist(q,p_{kth})$ // p_{kth} is the kth farthest data point to q among candidate data points in P_k
19 **return** P_k // P_k is returned if $maxdist(q,\overline{p_ip_j})$ is less than $prundist$ or all data segments in $\overline{P}$ are explored

$dist(q,p_{kth}) < dist(q,p)$가 성립하면, 결과 집합 P_k에서 p_{kth}를 제거하고, p를 새롭게 추가한다. 즉, $P_k \leftarrow P_k \cup \{p\} - \{p_{kth}\}$이다. 결과 집합 P_k의 원소의 개수가 k보다 작다면 ($|P_k| < k$), q에서 p_{kth}까지 거리는 $prundist = 0$로 정해진다. $maxdist(q,\overline{p_ip_j})$가 $prundist$보다 작거나(12−13줄), $\overline{P}$에 속하는 모든 데이터 세그먼트를 조사했다면(11줄), FANS 알고리즘은 결과 집합 P_k를 반환하고 종료한다. 따름정리 1은 종료 조건($maxdist(q,\overline{p_ip_j}) < prundist$)을 만족하면, 남은 데이터 세그먼트들은 조사할 필요가 없다는 것을 의미한다.

따름정리 1. 종료 조건($maxdist(q,\overline{p_ip_j}) < prundist$)을 만족하면, $\overline{p_ip_j}$에 포함된 데이터 객체 p는 q의 결과 집합 P_k에 속하지 못한다. $prundist$는 q에서 현재 k번째 최원접 후보 객체 p_{kth}까지 거리를 의미한다. $\overline{P}$에 속하는 데이터 세그먼트는 q까지 최대 거리를 기준으로 내림차순으로 정렬되어 있고, 현재 처리중인 데이터 세그먼트에서 q까지 최대 거리가 $prundist$보다 작다면, 나머지 데이터 세그먼트에 속하는 데이터 객체들은 결과 집합에 포함되지 않는다.

2. Processing an example kFN query using the FANS algorithm

그림 3의 k-최원접 이웃 질의 예제를 사용하여 FANS 알고리즘으로 2개의 최원접 데이터 객체를 찾는 과정을 설명한다. FANS 알고리즘의 1단계부터 3단계 과정을 진행하여, k = 2, $P=\{p_1,p_2,...,p_8\}$, $\overline{P} = \{\overline{p_2p_1p_6p_5}, \overline{p_3p_4}, \overline{p_7p_8}\}$가 주어져 있다. 그리고, 표 2에서 보이는 것처럼, q에서 $\overline{P}$의 데이터 세그먼트까지 최대 거리는 $maxdist(q,\overline{p_3p_4}) = 10$, $maxdist(q,\overline{p_2p_1p_6p_5}) = 9$, $maxdist(q,\overline{p_7p_8}) = 1$이기 때문에 $\overline{p_3p_4}$, $\overline{p_2p_1p_6p_5}$, $\overline{p_7p_8}$순서로 데이터 세그먼트를 조사한다.

FANS 알고리즘은 먼저 q까지의 최대 거리가 가장 큰 데이터 세그먼트 $\overline{p_3p_4}$을 탐색한다. 결과 집합 P_k와 임계값인 $prundist$는 각각 $P_k \leftarrow \varnothing$, $prundist \leftarrow 0$으로 초기화한다. 현재 결과 집합과 임계값은 각각 $P_k = \varnothing$, $prundist = 0$이다. 데이터 세그먼트 $\overline{p_3p_4}$의 최대 거리와 임계값이 $maxdist(q,\overline{p_3p_4}) = 10$, $prundist = 0$으로 알고리즘의 종료 조건($maxdist(q,\overline{p_3p_4})$ < prundist)을 만족하지 않는다. 결과 집합 $\overline{P}$에 추가할 데이터 객체를 찾기 위해 데이터 세그먼트 $\overline{p_3p_4}$에 포함된 데이터 객체 p_3, p_4에서 q까지 거리를 각각 계산한다. $dist(q,p_3) = 4$, $dist(q,p_4) = 9$이고, 데이터 객체 p_3, p_4는 결과 집합 $\overline{P}$에 추가된다. 결과 집합 P_k와 임계값 $prundist$은 다음과 같이 갱신된다. $\overline{P} = \{\langle p_4,9\rangle, \langle p_3,4\rangle\}$, $prundist = dist(q,p_3) = 4$.

같은 방식으로, 데이터 세그먼트 $\overline{p_2p_1p_6p_5}$을 탐색한다. 이때, 결과 집합과 임계값은 각각 $\overline{P} = \{\langle p_4,9\rangle, \langle p_3,4\rangle\}$, $prundist = 4$이다. 데이터 세그먼트 $\overline{p_2p_1p_6p_5}$의 최대 거리와 임계값이 $maxdist(q,\overline{p_2p_1p_6p_5}) = 9$, $prundist = 4$이므로, 알고리즘 종료 조건($maxdist(q,\overline{p_2p_1p_6p_5})$ < $prundist$)을 만족하지 않는다. 결과 집합 P_k에 추가할 데이터 객체를 찾기 위해 데이터 세그먼트 $\overline{p_2p_1p_6p_5}$에 포함된 데이터 객체 p_1, p_2, p_5, p_6에서 q까지 거리를 각각 계산한다. 여기서 $dist(q,p_1) = 8$, $dist(q,p_2) = 5$, $dist(q,p_5) = 7$, $dist(q,p_6) = 8$가 되어서, p_1이 결과 집합 P_k에 추가되고, p_3은 P_k에서 제거된다. 결과 집합 P_k와 임계값 $prundist$은 다음과 같이 갱신된다. $\overline{P} = \{\langle p_4,9\rangle, \langle p_1,8\rangle\}$, $prundist = dist(q,p_1) = 8$. 편의상 p_1과 p_6에서 q까지의 거리가 8로 같지만 무작위로 p_1을 선택한다.

마지막으로, 데이터 세그먼트 $\overline{p_7p_8}$을 탐색한다. 이때, 결과 집합과 임계값은 각각 $\overline{P} = \{\langle p_4,9\rangle, \langle p_1,8\rangle\}$, $prundist = 8$이다. 데이터 세그먼트 $\overline{p_7p_8}$의 최대 거리와 임계값이 $maxdist(q,\overline{p_7p_8}) = 1$, $prundist = 8$이므로, 알고리즘 종료 조건($maxdist(q,\overline{p_7p_8})$ < $prundist$)을 만족한다. FANS 알고리즘은 k-최원접 이웃 질의 결과 $\overline{P}$= $\{\langle p_4,9\rangle, \langle p_1,8\rangle\}$를 질의 객체 q에게 알려주고 종료한다.

VI. Performance study

1. Experimental settings

Table 3. Real-World Roadmaps

Name	Vertices	Edges	Vertex sequences
SJ	18,263	23,874	20,040
NA	175,813	179,179	12,416
SF	174,956	223,001	192,276

본 연구에서는 인터넷에서 얻을 수 있는 3개의 실제 도로 지도를 사용하여 비교 실험을 진행하였다[18]. 표 3은 실험에 사용된 도로 지도에 대한 간단한 설명을 제공한다. SJ 지도 데이터는 캘리포니아 샌 호아킨 지역(San Joaquin)의 도시 거리를 저장한다. NA 지도 데이터는 북미지역(North America)의 고속 도로를 저장한다. 마지막으로, SF 지도 데이터는 캘리포니아 샌프란시스코 지역

(San Francisco)의 도시 거리를 저장한다. 실험에서 사용하는 인자들의 조건은 표 4에 정리하였다. 편의상 데이터 공간의 각 차원은 단위 길이 [0,1]에 독립적으로 정규화하였다. 각 실험에서 표 4에 표시된 범위 내에서 한 개의 인자의 값을 변경하면서, 질의 처리 시간을 측정하였다.

Table 4. Experimental Parameter Settings

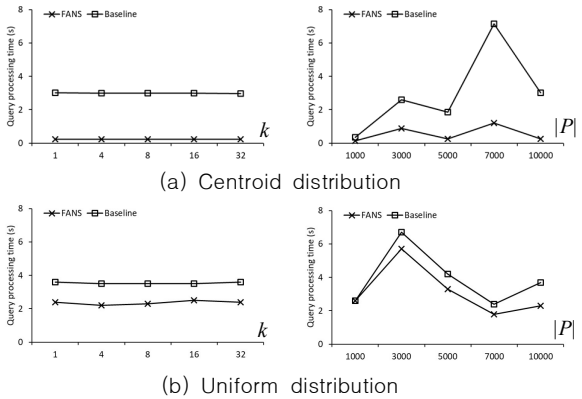
Parameter	Range
Number of FNs (k)	1, 4, 8, **16**, 32
Number of data points ($\|P\|$)	1, 3, 5, 7, **10** ($\times 10^3$)
Distribution of data points	Centroid, Uniform
Roadmap	SJ, NA, SF

데이터 객체들의 위치는 균등 분포와 비균등 분포를 따른다. 데이터 객체들의 비균등 분포는 정규분포를 활용하여 생성한다. 구체적으로, 5개의 중심점의 위치를 무작위로 선택한다. 데이터 객체들은 5개의 중심점을 기준으로 각각 정규분포를 따른다. 중심점이 정규분포의 평균이 되고, 데이터 공간의 측면 길이의 1%가 표준편차가 된다.

FANS 알고리즘의 성능을 비교 분석하기 위하여, Baseline 알고리즘을 사용한다. Baseline 알고리즘은 질의 객체에서 모든 데이터 객체까지 거리를 각각 계산하고, 가장 멀리 있는 k개의 데이터 객체를 선택한다. 두 방법 모두 Microsoft Visual Studio 2019에서 C++ 컴파일러를 사용하여 구현했고, 유사한 작업에서는 공통 모듈을 사용했다. 비교 실험은 Windows 10 운영체제, 4.2 GHz 프로세서와 32 GB 메모리가 설치된 컴퓨터에서 수행되었다. 질의 사용자들에게 빠른 질의 처리를 보장하기 위해서, 색인 구조는 주기억 장치에 존재해야 한다. 이러한 가정은 최근 연구(예: [15, 19])에서 많이 채택되었고, 온라인 지도 서비스와 상용 위치기반서비스에 필수적이다. 우리는 무작위로 선택된 50개의 질의 객체에 대하여 k-최원접 이웃 탐색 질의를 수행하고, 질의 처리 시간의 평균을 측정하였다. 마지막으로, 도로 네트워크에서 두 지점 사이의 거리를 빠르게 계산하기 위하여, TNR(Transit Node Routing)[20] 방법을 사용하였다. 본 연구에서는 질의 객체와 데이터 객체들이 자유롭게 도로를 따라서 이동할 수 있다고 가정하기 때문에, 데이터 객체들에 대한 미리 계산된 색인 기법을 사용하지 않는다.

2. Experimental results

그림 7은 SJ 지도 데이터를 사용하여, FANS 알고리즘과 Baseline 알고리즘의 질의 처리 시간을 측정한 결과를 보여준다. 그림 7(a)는 데이터 객체들이 비균등 분포할 때, 최원접 객체들의 개수(k)를 1부터 32까지 변경한 경우와 데이터 객체들의 개수($|P|$)를 1,000부터 10,000까지 변경한 경우에 대하여 각각 두 알고리즘의 질의 처리 시간을 비교하였다. Baseline 알고리즘은 k개의 최원접 객체를 찾기 위하여, 질의 객체 q에서 모든 데이터 객체까지 거리를 계산하고, 그 중에서 가장 멀리 떨어진 k개의 데이터 객체를 찾기 때문에, k값의 변화가 Baseline 알고리즘의 질의 처리 시간에 크게 영향을 주지 않는다. FANS 알고리즘은 데이터 객체들을 그룹화하고, 질의 객체로부터 데이터 세그먼트까지 최소 거리와 최대 거리를 계산한다. 그 이후에 질의 객체로부터 멀리 떨어져 있는 데이터 세그먼트부터 순차적으로 조사하여. 불필요한 거리 계산이 많이 줄었기 때문에, FANS 알고리즘의 질의 처리 시간은 Baseline 알고리즘보다 12.5배 짧았다. 그림 7(a)의 오른쪽 도표는 데이터 객체들의 개수($|P|$)를 1,000부터 10,000까지 변경하면서 질의 처리 시간을 비교하였다. 도로 네트워크에서 질의 처리 시간은 거리 측정 횟수에 비례하고, 두 지점 사이의 거리 측정 시간은 두 지점의 위치에 따라 달라질 수 있기 때문에 데이터 객체들의 분포가 질의 처리 시간에 많은 영향을 준다. 데이터 객체들의 개수가 작은 경우에도 질의 객체의 위치에 따라서 질의 처리 시간이 오래 걸리는 경우가 발생한다. 구체적으로, $|P|$=7,000에서의 질의 처리 시간이 $|P|$=10,000에서의 질의 처리 시간보다 오래 걸린다. FANS 알고리즘은 모든 경우에서 Baseline 알고리즘보다 우수한 성능을 보여주었다. 평균적으로, FANS 알고리즘의 질의 처리 시간은 Baseline 알고리즘보다 6.1배 짧았다.



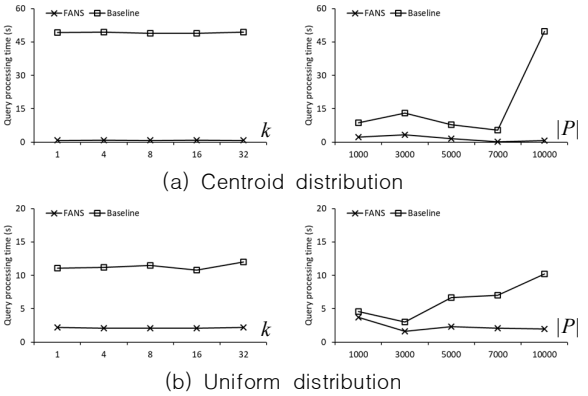
(a) Centroid distribution

(b) Uniform distribution

Fig. 7. Comparison of query processing time for roadmap SJ

그림 7(b)는 데이터 객체들이 균등 분포할 때, 최원접 객체들의 개수(k)와 데이터 객체들의 개수($|P|$)를 변경하면서 질의 처리 시간을 비교하였다. 데이터 객체들이 균등 분포하는 경우에, FANS 알고리즘과 Baseline 알고리즘은 차이가 비균등 분포하는 경우보다 질의 처리 시간의 차이가 크지 않았다. 이유는 데이터 객체들이 균등 분포하는 경우에는 데

이터 객체들의 그룹화가 잘되지 않아서 FANS 알고리즘이 성능이 좋지 않았다. 그럼에도 불구하고, FANS 알고리즘의 질의 처리 시간은 Baseline 알고리즘보다 1.4배 짧았다.

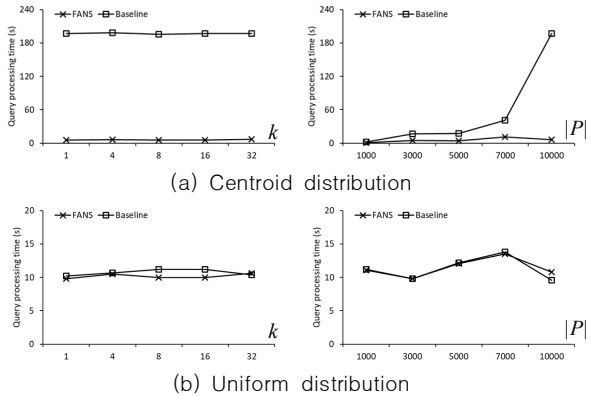


(a) Centroid distribution

(b) Uniform distribution

Fig. 8. Comparison of query processing time for roadmap NA

그림 8은 NA 지도 데이터를 사용하여, FANS 알고리즘과 Baseline 알고리즘의 질의 처리 시간을 측정한 결과를 보여준다. 그림 8(a)는 데이터 객체들이 비균등 분포할 때, 최원접 객체들의 개수(k)와 데이터 객체들의 개수($|P|$)를 변경하면서 질의 처리 시간을 비교하였다. 최원접 객체들의 개수 k를 변경하면서 실험한 경우에, FANS 알고리즘의 질의 처리 시간은 Baseline 알고리즘보다 평균적으로 60배 짧았다. 이것은 데이터 객체들을 그룹화해서 처리하는 방법이 매우 효율적임을 분명하게 보여준다. 데이터 객체들의 개수 $|P|$를 변경하면서 실험한 경우에, FANS 알고리즘의 질의 처리 시간은 Baseline 알고리즘보다 평균적으로 26.3배 짧았다. 그림 8(b)는 데이터 객체들이 균등 분포할 때, 최원접 객체들의 개수(k)와 데이터 객체들의 개수($|P|$)를 변경하면서 질의 처리 시간을 비교하였다. 모든 경우에서 FANS 알고리즘은 Baseline 알고리즘보다 우수한 성능을 보여주었다.



(a) Centroid distribution

(b) Uniform distribution

Fig. 9. Comparison of query processing time for roadmap SF

그림 9는 SF 지도 데이터를 사용하여, FANS 알고리즘과 Baseline 알고리즘의 질의 처리 시간을 측정한 결과를 보여준다. 그림 9(a)는 데이터 객체들이 비균등 분포할 때, 최원접 객체들의 개수(k)와 데이터 객체들의 개수($|P|$)를 변경하면서 질의 처리 시간을 비교하였다. 데이터 객체들이 비균등 분포할 때, 데이터 객체들의 그룹화가 잘되기 때문에, 공유 실행 방법의 효과가 매우 크다. 결과적으로, FANS 알고리즘이 Baseline 알고리즘보다 우수한 성능을 보여주었다. 그림 9(a)의 왼쪽 도표에서 보는 것처럼, 최원접 객체들의 개수 k값을 변경하면서 실험한 경우에, FANS 알고리즘의 질의 처리 시간은 Baseline 알고리즘보다 평균적으로 32배 짧았다. 그림 9(a)의 오른쪽 도표에서 보는 것처럼, 데이터 객체들의 개수 $|P|$를 변경하면서 실험한 경우에, FANS 알고리즘의 질의 처리 시간은 Baseline 알고리즘보다 평균적으로 9.2배 짧았다. 또한, 데이터 객체들의 개수가 증가할수록, FANS 알고리즘과 Baseline 알고리즘의 성능 차이는 빠르게 증가하였다. 그림 9(b)는 데이터 객체들이 균등 분포할 때, 최원접 객체들의 개수(k)와 데이터 객체들의 개수($|P|$)를 변경하면서 질의 처리 시간을 비교하였다. 데이터 객체들이 균등 분포할 때, FANS 알고리즘과 Baseline 알고리즘이 매우 비슷한 성능을 보여주었다. SF 지도 데이터가 복잡하고, 데이터 객체들이 균등 분포하기 때문에, 데이터 객체들의 그룹화가 잘되지 않는다. 결과적으로, FANS 알고리즘은 Baseline 알고리즘과 비슷한 성능을 보여주었다.

VII. Conclusions

일반적으로 사람이나 자동차와 같은 운송수단은 주어진 경로를 따라서 이동한다[3, 7, 21]. 그러므로, 유클리드 거리를 사용한 기존 공간 질의 처리 방법들은 도로 네트워크에서의 공간 검색 문제를 해결하는 데 어려움이 있다. 도로 네트워크에서 k-최원접 이웃 검색 문제는 네트워크 거리 계산 횟수를 효율적으로 줄이는 것이 중요하다. 본 논문에서는 도로 네트워크에서 k-최원접 이웃 검색을 위한 FANS 알고리즘을 제안했다. FANS 알고리즘은 질의 객체와 데이터 객체 간의 불필요한 거리 계산을 줄이기 위하여, 인접한 데이터 객체들을 그룹화하고 질의 객체에서 데이터 세그먼트까지 최대 거리를 기반으로 효율적인 가지치기 기법을 제시한다. 잘 알려진 최단 경로 거리 계산 알고리즘을 사용하여 FANS 알고리즘을 쉽게 구현할 수 있다. 마지막으로, 실제 지도 데이터를 사용한 다양한 조건

에서 실험을 수행하여 FANS 알고리즘의 우수성을 입증했다. 실험 결과를 요약하면, 데이터 객체들이 비균등 분포하는 경우에 FANS 알고리즘은 기존 방법보다 평균적으로 24.5배 이상 우수한 성능을 보여주었다.

REFERENCES

[1] R.R. Curtin, J. Echauz, and A.B. Gardner, “Exploiting the Structure of Furthest Neighbor Search for Fast Approximate Results,” Information Systems, Vol. 80, pp. 124-135, February 2019.

[2] B. Yao, F. Li, and P. Kumar, “Reverse Furthest Neighbors in Spatial Databases,” Proceedings of International Conference on Data Engineering, pp. 664-675, April 2009.

[3] X.-J. Xu, J.-S. Bao, B. Yao, J. Zhou, F. Tang, M. Guo, and J. Xu, “Reverse Furthest Neighbors Query in Road Networks,” Journal of Computer Science and Technology, Vol. 32, No. 1, pp. 155-167, January 2017.

[4] J. Liu, H. Chen, K. Furuse, and H. Kitagawa, “An Efficient Algorithm for Arbitrary Reverse Furthest Neighbor Queries,” Proceedings of Asia-Pacific Web Conference, pp. 60-72, 2012.

[5] Q.T. Tran, D. Taniar, and M. Safar, “Reverse k Nearest Neighbor and Reverse Farthest Neighbor Search on Spatial Networks,” Transactions on Large-Scale Data- and Knowledge-Centered Systems I, Vol. 5740, pp. 353-372, 2009.

[6] H. Wang, K. Zheng, H. Su, J. Wang, S.W. Sadiq, and X. Zhou, “Efficient Aggregate Farthest Neighbour Query Processing on Road Networks,” Proceedings of the Australasian Database Conference, pp. 13-25, 2014.

[7] D. Papadias, J. Zhang, N. Mamoulis, Y. Tao, “Query Processing in Spatial Network Databases,” Proceedings of International Conference on Very Large Data Bases, pp. 802-813, September 2003.

[8] C. Shahabi, M.R. Kolahdouzan, and M. Sharifzadeh, “A Road Network Embedding Technique for K-Nearest Neighbor Search in Moving Object Databases,” GeoInformatica, Vol. 7, No. 3, pp. 255-273, September 2003.

[9] M.R. Kolahdouzan and C. Shahabi, “Voronoi-Based K Nearest Neighbor Search for Spatial Network Databases,” Proceedings of International Conference on Very Large Data Bases, pp. 840-851, August 2004.

[10] X. Huang, C.S. Jensen, and S. Saltenis, “The Islands Approach to Nearest Neighbor Querying in Spatial Networks,” Proceedings of International Symposium on Spatial and Temporal Databases, pp. 73-90, 2005.

[11] H. Samet, J. Sankaranarayanan, and H. Alborzi, “Scalable Network Distance Browsing in Spatial Databases,” Proceedings of International Conference on Management of Data, pp. 43-54, June 2008.

[12] K.C.K. Lee, W.-C. Lee, B. Zheng, and Y. Tian, “ROAD: A New Spatial Object Search Framework for Road Networks,” IEEE Transactions on Knowledge and Data Engineering, Vol. 24, No. 3, pp. 547-560, March 2012.

[13] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, “The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles,” Proceedings of International Conference on Management of Data, pp. 322-331, May 1990.

[14] R. Zhong, G. Li, K.-L. Tan, L. Zhou, and Z. Gong, “G-Tree: An Efficient and Scalable Index for Spatial Search on Road Networks,” IEEE Transactions on Knowledge and Data Engineering, Vol. 27, No. 8, pp. 2175-2189, February 2015.

[15] T. Abeywickrama, M.A. Cheema, and D. Taniar, “k-Nearest Neighbors on Road Networks: A Journey in Experimentation and In-Memory Implementation,” Proceedings of the VLDB Endowment, Vol. 9, No. 6, pp. 492-503, January 2016.

[16] Y. Gao, L. Shou, K. Chen, G. Chen, “Aggregate Farthest-Neighbor Queries over Spatial Data,” Proceedings of International Conference on Database Systems for Advanced Applications, pp. 149-163, 2011.

[17] S. Wang, M.A. Cheema, X. Lin, Y. Zhang, and D. Liu, “Efficiently Computing Reverse k Furthest Neighbors,” Proceedings of International Conference on Data Engineering, pp. 1110-1121, May 2016.

[18] Real datasets for spatial Databases, *https://www.cs.utah.edu/~lifeifei/SpatialDataset.htm.*

[19] L. Wu, X. Xiao, D. Deng, G. Cong, A.D. Zhu, and S. Zhou, “Shortest Path and Distance Queries on Road Networks: An Experimental Evaluation,” Proceedings of the VLDB Endowment, Vol. 5, No. 5, pp. 406-417, January 2012.

[20] H. Bast, S. Funke, and D. Matijevic, “TRANSIT: Ultrafast Shortest-Path Queries with Linear-Time Preprocessing,” Proceedings of 9th DIMACS Implementation Challenge, pp. 175-192, 2006.

[21] J. Yoo, “CRM using Short Range Location Based Technology,” Journal of the Korea Society of Computer and Information, Vol. 21, No. 12, pp. 91-96, December 2016.

Authors

Taelee Kim is currently a M.S. student at the department of Software, Kyungpook National University. She is interested in spatial network database, big data processing and machine learning.

Hyung-Ju Cho is an associate professor at the department of software, Kyungpook National University. His current research interests include moving object databases and query processing in mobile peer-to-peer networks.

Hee Ju Hong is currently a student of Daegu Science High School.

Hyogeun Nam is currently a student of Daegu Science High School.

Hyejun Cho is currently a student of Daegu Science High School.

Gyung Yoon Do is currently a student of Daegu Science High School.

Pilkyu Jeon is currently a student of Daegu Science High School.