최근접 이웃 탐색 기반의 향상된 스카이라인 질의를 위한 전처리 기법‡

김지현*, 이상민*, 전형준**, 진창균**, 김지윤+, 권진영++, 김종완†, 오덕신§
*삼육대학교 인공지능·빅데이터 소프트웨어코드 연구소
삼육대학교 {**컴퓨터·메카트로닉스공학부, +식품영양학과, †스미스학부대학, §경영정보학과}
++건국대학교 정보통신경영학과
yeaheggg@gmail.com, sangmin010203@gmail.com, chariot0720@gmail.com, jcg6074@naver.com,
wldbs3592@naver.com, oac0801@naver.com, kimj@syu.ac.kr, ohds@syu.ac.kr

Nearest Neighbor-based Pre-processing Scheme for Advanced Skyline Query

Ji-Hyun Kim*, SangMin Lee*, Hyeongjun Jeon**, ChangGyun Jin**, JiYun Kim+, Jin young Kwon++,
Jongwan Kim†, Dukshin Oh§
*AI·Big Data and Software Code Lab., Sahmyook University

요 약

스카이라인 질의는 객체의 속성을 기준으로 사용자의 선호에 적합한 대상을 탐색하는 기법이다. 기존 스카이라인 질의는 일괄처리 방식으로 탐색 결과를 반환하지만 대화형 앱이나 모바일 환경과 같이 잦은 위치이동 발생 시 일괄처리 방식으로 스카이라인 질의 결과를 신속하게 받기 어렵다. 최근접 이웃(Nearest Neighbor) 알고리즘은 사용자와 상호 작용이 필요한 대화형 앱에서 실시간으로 선호 객체를 탐색하여 사용자에게 전달함으로써 객체의 반환 속도를 향상시켰다. 그러나 최근접 이웃 알고리즘은 객체 탐색 과정에서 반복적인 비교 연산을 수행하여 불필요한 탐색 시간이 소요된다. 본 논문은 대화형 앱에서 신속한 스카이라인 결과를 산출하고자 연산 대상 객체의 범위를 축소함으로써 최근접 이웃 스카이라인 질의 알고리즘의 성능을 향상시킨 전처리 기법을 제안한다. 데이터 객체는 최대 40,000 개의 실험에서 제안 기법은 최근접 이웃 알고리즘보다 50% 빠른 성능을 나타내어 본 연구의 가용성이 증명되었다.

1. 서론

스마트 기기의 발전에 따라 대규모(Volume)의 멀티미디어 콘텐츠가 빠르게(Velocity) 생산되며, 각종 센서와 SNS 로 인해 다양한(Variety) 데이터의 수집이 용이해졌다. 빅데이터가 발전하면서 데이터의 의미를 찾는 것이 중요해졌고 이로 인해 데이터 마이닝이 중요한 데이터 분석 기법으로 사용된다. 특히 데이터 객체의 속성이 다양한 경우에는 사용자의 선호도에 부합하는 데이터를 탐색할 수 있는 분석 기법이 필요하다.

Skyline Query [1]는 다차원 속성을 비교하여 사용자의 선호도에 맞는 객체를 추천하는 대표적인 기법으로 다른 속성에 의해 지배되지 않는 객체들을 탐색한다. 여기서 '지배된다'의 의미는 하나의 객체가 다른 객체에 대해 모든 속성에서 좋지 않은 값을 가지는 상태를 말한다. 예를 들어 중고차를 구매할 때 자동차 A, B 가 속성으로 가격과 주행거리를 가지고 A 는 (3, 5), B 는 (6, 7)의 경우라고 하자. 사용자의 선호가 낮은 값일 때 B 는 모든 속성에서 높은 값을 가지고 있으므로 A 에 의해 지배된다. 즉, 점 A 가 Skyline 후보군이 된다.

최근접 이웃 질의(Nearest Neighbor Query, NN) [2]는 Skyline Query 의 정의에 부합하기 위해 빠른 결과를 제공한다. 먼저, D&C(Divide and Conquer) 알고리즘을 사용하여 데이터를 여러 부분의 파티션으로 나누고 각 부분에 대한 최근접 이웃 [3]을 구한다. NN의 성능은 탐색 영역에 포함된 데이터의 개수에 따라 달라지는데, 만약 탐색 영역에 데이터의 분포가 밀집되어 있다면 방문한 영역을 재 탐색하게 되므로 탐색 시간이 중복되어 전체적으로 성능 저하가 발생한다.

‡ 이 논문은 2020 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (NRF-2018R1D1A1B07045642, NRF-2017R1D1A1B03035884).
§ 교신저자

본 논문에서는 NN 알고리즘을 사용할 때 전처리를 통해 성능 개선에 도움을 주는 Pre-NN 알고리즘을 제안한다. Pre-NN 은 NN 질의에서 방문해야 할 데이터의 개수를 줄여 줌으로써 탐색 성능을 향상 시킨다.

탐색할 데이터 객체가 많다면 NN 은 최악의 성능을 가지게 되므로 객체들 사이의 지배관계를 평가하여 불필요한 데이터들은 제거하는 방법으로 NN 의 성능 향상에 도움을 줄 수 있다.

논문의 구성은 다음과 같다. 2 장은 기존의 스카이라인 질의에서 사용한 BNL, D&C 및 NN 알고리즘에 대해 설명한다. 3 장에서는 논문에서 제안하는 Pre-NN 알고리즘을 소개하고 이를 기존 NN 에 적용 시 갖게 되는 효과에 대해 기술한다. 4 장에서 Pre-NN 과 선행하는 NN 기법 간의 연산 속도를 비교함으로써 제안 기법의 성능을 증명한다. 마지막으로 5 장은 결론을 서술한다.

2. 관련연구

본 절에서는 Skyline Query [1]에서 적용한 BNL(Block Nested Loop), D&C(Divide and Conquer)를 살펴보고 제안 기법이 개선하고자하는 NN 알고리즘 [2]에 대해 설명한다.

2.1 블록 중첩 루프 알고리즘

BNL 은 Skyline 을 구하기 위한 가장 직관적인 방법으로 Skyline 후보군 리스트를 만들고 리스트 내부의 객체와 새로 탐색되는 데이터 객체를 비교한다.

스카이라인 탐색을 시작하면 첫 번째 객체가 리스트에 들어가며 다음 탐색 객체와 리스트의 객체를 비교하여 만약 탐색 객체가 리스트의 객체를 지배한다면 리스트의 객체와 탐색 객체의 위치를 바꾼다.

리스트 내부 객체와 데이터의 객체를 비교할 때 세 가지 경우가 발생한다. 데이터 객체가 (1) 리스트 내부 객체에 의해 지배되는 경우 (2) 지배되지 않는 경우 (3) 위의 (1), (2)번이 모두 아닌 경우이다. 위 경우들 중 (1)을 제외하고는 모두 Skyline 후보군 리스트에 포함된다. 후보군 리스트를 사용하여 리스트 내부 객체들과 데이터 객체 전체를 비교하므로 BNL 의 Skyline 후보군 리스트는 계산 과정이 길어질 수록 길어진다 [4].

2.2 분할 정복 알고리즘

D&C 는 데이터를 여러 부분으로 나눈 뒤, 나눠진 부분들에 대해서 Skyline 객체를 구하고 부분 객체들을 합치면서 전체 Skyline 질의 결과를 구한다. 예를 들어 입력 받은 데이터의 중간 값을 계산한 뒤, 전체 데이터를 n 개의 파티션으로 나눈다. 파티션의 각 부분에서 Skyline 을 구하고 해당 영역에서 다시 중간 값을 계산하여 파티션으로 나누는 일을 반복한다. D&C 는 해당 과정을 파티션이 비어있거나 하나의 객체가 남을 때까지 반복한다. D&C 방식은 결국 전체 영역을 살펴보면서 반복적으로 파티션을 나누는 작업을 하기 때문에 전체 영역의 개수가 크면 클수록 성능이 나쁘다 [4].

BNL 과 D&C 는 Skyline 객체를 모두 구한 후에 한 번에 결과를 반환하는 일괄처리 방식이다. 일괄처리 방식은 즉각적인 반응(interactive reaction)을 요구하는 대화형 애플리케이션에서 사용하기에는 부적절하다. 따라서 NN 알고리즘[2]에서는 이러한 점을 해결하기 위해 Divide and Conquer 에 Nearest Neighbor 방식 [3]을 결합하여 스카이라인의 일부를 구함과 동시에 반환하는 알고리즘을 제안하였다.

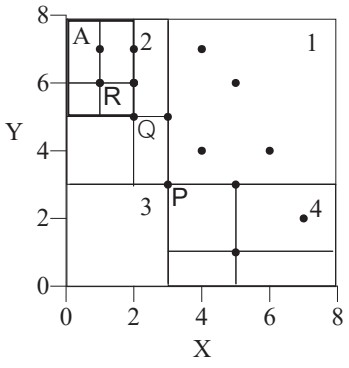
2.3 최근접 이웃 스카이라인 질의

NN 알고리즘은 D&C 방식을 토대로 최근접 객체를 찾는다. NN 은 기존의 BNL 과 D&C 와 같은 일괄처리 방식이 즉각적인 반환을 할 수 없다는 단점을 해결하였다.

NN 은 D&C 방식과 같이 영역을 분할하면서 해당 영역에서 스카이라인을 구하는 즉시 반환하기 때문에 대화형 애플리케이션에 적합하다. NN 알고리즘을 사용하기 위해서는 전체 데이터 중에서 영점 (0, 0)과 가까운 NN 객체를 찾고 해당 객체를 기준으로 2, 4 사분면을 대상으로 재연산한다. 이때, NN 에 의해 지배되는 1 사분면의 객체들은 제거된다.

예를 들어, (그림 1)에서 첫 번째 NN 점은 전체 영역을 대상으로 구하게 되므로 유클리드 거리 [5]가 $3\sqrt{2}$ 인 P (3, 3) 객체가 된다. P 를 대상으로 2, 4 사분면으로 영역을 나누고 2 사분면을 대상으로 NN 객체를 찾는다. 다음은 (0, 3)에서 (3, 8)까지의 영역에서 NN 객체를 찾게 되므로 유클리드 거리가 $\sqrt{29}$인 Q (2, 5)가 NN 이 된다. 마지막으로 Q에 의해 2, 4 사분면으로 영역을 나누고 2 사분면을 살펴보게 된다면. 유클리드 거리가 $\sqrt{37}$ 인 R(1, 6)이 NN 에 해당한다.

여기서 영역 A 의 객체들은 P, Q, R 을 찾는 과정에서 반복적으로 재 탐색되어 3 번의 연산 과정에 참여하게 된다. A 영역의 객체 개수가 많을 수록 불필요한 연산에 노출되는 객체가 존재하며 이는 성능 저하를 발생 시킨다. 위 와 같은 NN 의 단점은 객체들에 대한 지배관계 비교를 통한 탐색 대상 축소 방법으로 극복할 수 있다.



(그림 1) Nearest Neighbor Query

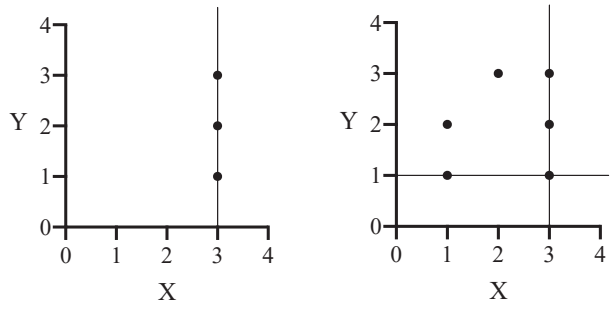
3. 스카이라인 질의 전처리 기법

스카이라인은 다량의 데이터 객체에서 사용자의 요구를 만족하는 속성을 중심으로 객체를 추천하기 위

해 사용된다. 기존의 스카이라인 질의에서 NN 알고리즘은 대화형 앱을 위해 실시간으로 스카이라인 객체를 출력하였으나 반복적인 객체 비교에 의해 탐색 성능이 저하되는 단점이 있었다.

본 논문에서 제안하는 스카이라인 질의 전처리기법인 Pre-NN 은 NN 알고리즘을 수행하기 전에 각 속성을 나타내는 축에서 탐색 범위를 제한 함으로써 처리 성능을 향상시킨다.

본 기법에서 속성 범위를 제한하는 방법은 다음과 같다. (그림 2(a))에서 x 축을 기준으로 x=3 이라는 가상의 수직선을 긋는다면 (3, 1), (3, 2), (3, 3)의 객체가 존재한다. 해당 객체들은 모두 x 축은 3 으로 고정되어 있고, y 축만 다르다. 즉 3 으로 고정되어 있는 x 축 외에 y 축에 대해서만 여러 값을 갖는다. 선분에서 최솟값을 갖는 객체는 (3, 1)이 되므로 해당 정보만이 Skyline 후보군으로 저장한다. (그림 2(b))는 (그림 2(a)) 이후의 상황으로 y=1 이라는 가상의 선에서 최솟값은 (1, 1)이 된다. 기존에 Skyline 후보군에 들어있었던 (3, 1)보다 (1, 1)이 x 속성에 대해 더 작은 값을 가지게 되므로 (3, 1)을 후보군에서 제거하고 (1, 1)을 후보군에 포함시킨다.



(a) x=3 (b) y=1

(그림 2) Pre-NN 예시

제안기법에서 각 축의 수직선으로 구별되는 최소 영역은 다음과 같이 정의된다.

Definition: 수직선의 최소범위(Minimum Range of Vertical Line).
ds 를 탐색영역의 데이터 세트라 할 때 x, y 는 데이터 객체의 속성이라 하자. 두 속성은 2 차원 공간에서 각각 x 축과 y 축으로 표현된다. 이때, 각 축에서 데이터의 탐색 범위를 축소하기 위한 객체의 최소범위 (x, y)는 다음과 같다.

$$\{\forall x, \exists y, x \in ds \ \wedge \ y \in \min(y)\} \quad (1)$$
$$\{\exists x, \forall y, x \in \min(x) \ \wedge y \in ds\} \quad (2)$$

위의 정의는 x, y 축에 대한 범위 제한을 표현하고 있으며 수식 (1)은 x 축의 탐색 범위를 제한하는 것으로 x 를 전체 데이터 세트(ds)로 놓고, y 는 각 x 의 수직선 상에 있는 좌표 중 최솟값을 의미한다. 이는 (그림 2(a))에 적용된다. 수식 (2)는 y 축의 수직선을 기준으로 y 를 전체 데이터 세트(ds)로 놓고, x 는 최솟값으로 탐색 범위를 제한한다.

```
Algorithm: pre_nn (ds)
Input: Dataset ds
M ← max value of integer
yMin ← dict-type which default value is m
xMin ← dict-type which default value is m
candidate ← empty dictionary
1:  for x, y in data
2:      if y < yMin[x]
3:          yMin[x] = y
4:          if xMin[yMin[x]] > x
5:              if yMin[xMin[x]] isnot m
                and if yMin[xMin[x]] in candidate
6:                  delete candidate[xMin[yMin[x]]
7:              xMin[yMin[x]] = x
8:              candidate[xMin[yMin[x]] = yMin[x]
9:          elseif xMin[yMin[x]] < x
10:             if x in candidate
11:                 delete candidate[x]
12: return candidate
```

(그림 3) Pre-NN 알고리즘

Pre-NN 알고리즘은 크게 두 가지 단계로 진행된다. 첫 번째 단계로, 모든 데이터에 대하여 만약 x 값에 대해 최소값으로 저장된 y 값이(yMin[x]) 실제 y 속성보다 크다면 yMin[x]의 값을 y 로 대체한다 (1~3). 두 번째 단계로, 만약 y 값에 대해 최솟값으로 저장된 x 값이 실제 x 속성보다 클 때(4 line) 만약 y 의 최솟값으로 저장된 배열의 값이 초기값이 아니면서 후보군에 이미 들어 있다면 스카이라인 후보군에서 해당 객체를 지워줘야 한다 (5~6 line).

4 번 줄에 이어서 xMin[yMin[x]]의 값을 x 로 대체한다. 그 뒤 해당 객체를 스카이라인 후보군에 저장한다 (7~8 line). 4 번 줄과 반대로 xMin[yMin[x]]의 값이 x 보다 작고 또한 x 가 후보군 안에 들어 있다면 candidate 에서 x 에 대한 정보를 삭제한다 (9~11 line). 마지막으로 저장된 후보군 딕셔너리(Dict)를 반환한다 (12 line).

Pre-NN 알고리즘은 같은 선상에 위치한 데이터들의 지배 관계를 비교하여 연산에 불필요한 객체들은 미리 제거해주는 알고리즘이다. 본 알고리즘의 장점은 데이터의 개수가 전체 영역의 최대 포함 가능 개수에 가까워질수록 밀집된 데이터가 많아지므로 불필요한 객체가 제거되어 탐색 성능이 향상된다는 것이다.

4. 실험

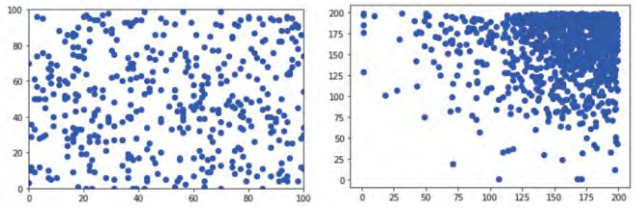
실험을 진행할 대상은 (그림 4(a))의 이산 균등 분포 데이터 세트와 (그림 4(b))의 이산 편향 분포 데이터 세트이다. NN 알고리즘과의 실험 환경을 같게 하기 위해 데이터의 중복은 없다고 가정한다.

실험 환경은 다음의 <표 1>과 같다.

<표 1> 실험 환경

구분	내용
시스템	Intel Xeon Silver 4114 CPU 2.20GHz * 2 개, RAM 128GB

언어	Python
데이터 범위	200 개, 300 개
데이터 수	1,000 ~ 20,000 개
속성 수	2 개
실험 횟수	5,000 회

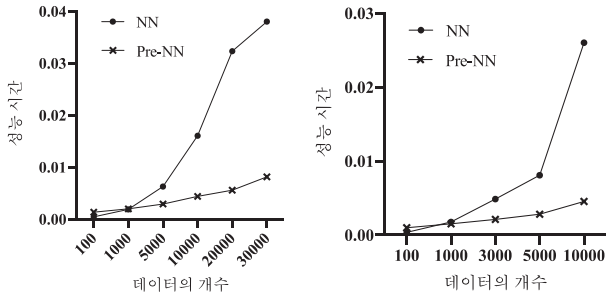


(a) 균등 분포 (b) 편향 분포

(그림 4) 이산 데이터 분포

x 축과 y 축의 범위는 편의를 위해 200 개로 지정하였고 중복이 없다는 가정에 따라 전체 영역으로 들어올 수 있는 데이터 세트의 개수는 x × y 로 40,000 개가 된다. 균등 분포의 경우 데이터 세트는 범위 사이의 값들을 랜덤하게 받아들이므로 데이터들은 (그림 4(a))와 같이 영역의 전반에 고르게 분포하게 된다.

실험은 극단적인 상황을 제외하기 위해 데이터의 개수를 최소 100 개, 최대 30,000 개로 지정한다. 편향 분포의 경우 (그림 4(b))와 같이 사각형의 특정 부분으로 치우친 데이터 세트가 형성되기 때문에 전체 영역으로 들어올 수 있는 데이터 개수의 1/4 크기에 해당하는 개수(사각형의 네 모서리 중 한 부분)만 실험한다. 즉 균등 분포의 경우 100 개, 1,000 개, 5,000 개, 10,000 개, 20,000 개, 30,000 개의 데이터로 실험을 진행하였고, 편향 분포의 경우 최대 데이터 개수의 1/4 개까지인 100 개, 1,000 개, 3,000 개, 5,000 개, 10,000 개 일 때의 pre-NN 과 NN 의 실행 시간을 비교하였다.



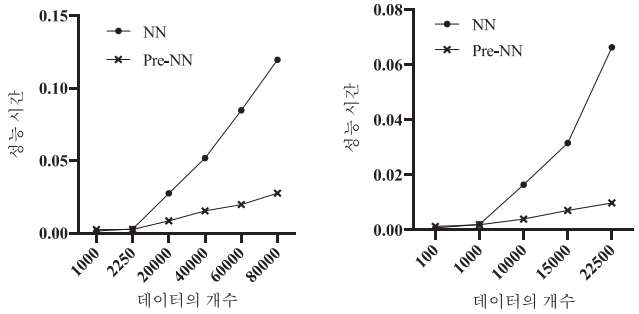
(a) 이산 균등 분포 (b) 이산 편향 분포

(그림 5) 200×200 평면에서의 성능 실험

데이터 공간이 x, y 축을 기준으로 확장하여도 동일한 성능 그래프를 나타내는지 확인하기 위해 두 번째 실험에서는 x축과 y축의 범위를 300 개로 지정하였다. 전체 영역으로 들어올 수 있는 데이터 세트 개수는 x × y 로 90,000 개가 된다. 균등 분포 데이터는 1,000 개, 2,250 개, 20,000 개, 40,000 개, 60,000 개, 80,000 개의 데이터로 진행한다. 편향 분포의 경우 최대 데이터 개수의 1/4 개까지인 100 개, 1,000 개, 10,000 개, 15,000 개, 22,500 개 일 때의 pre-NN 과 기존 NN 의 실행 시간을 비교하였다.

이산 균등 분포에서 Pre-NN 을 적용한 NN 은 기존 NN 에 비해 데이터의 밀집도가 커질 때마다 각각 약 1 배, 2 배, 4 배, 6 배의 순서로 빨라진다. 편향 분포에서 Pre-NN 을 적용한 NN 은 기존 NN 에 비해 약 1 배, 1.5 배, 3 배, 4.5 배의 순서로 빨라진다.

실험 결과 균등 분포의 경우 전체 영역으로 들어올 수 있는 최대 데이터 개수의 2.5%에 해당하는 개수 이상의 데이터가 분포되어 있으면 ((그림 5)에서는 데이터의 개수 1,000, (그림 6)에서는 데이터의 개수 2,250 에 해당) Pre-NN 의 성능이 좋아진다. 편향 분포의 경우 1,000 개를 기점으로 Pre-NN 을 사용했을 때 성능 향상을 보였다.



(a) 이산 균등 분포 (b) 이산 편향 분포

(그림 6) 300×300 평면에서의 성능 실험

5. 결과

본 논문에서 제안한 Pre-NN 알고리즘은 스카이라인 질의를 수행할 때 지배 관계를 미리 비교함으로써 NN 에서 처리할 데이터 객체의 수를 축소하였다. 이는 NN 에서 이웃하는 객체들에 대하여 비교해야 할 연산 횟수를 줄여줌으로써 전체 성능을 향상시키는 효과를 가져온다.

실험을 통해 제안 기법이 기존 NN 알고리즘 방식보다 빠른 속도로 스카이라인을 반환할 수 있음을 보였다.

참고문헌

[1] S. Bo¨rzso¨nyi, D. Kossmann, and K. Stocker. “The skyline operator,” In Proc. IEEE Conf. on Data Engineering, Heidelberg, Germany, pp. 421-430, 2001

[2] D. Kossmann, F. Ramsak, and S. Rost, “Shooting Stars in the Sky: an Online Algorithm for Skyline Queries,” VLDB, pp. 275-286, 2002

[3] N. Roussopoulos, S. Kelley, and F. Vincent. “Nearest neighbor queries,” In Proc. of the ACM SIGMOD Conference, San Jose, CA, May 1995

[4] D. Papadias, Y. Tao, G. Fu, and B. Seeger: “An optimal and progressive algorithm for skyline queries,” In: ACM SIGMOD International Conference on Management of Data, pp. 467-478, 2003

[5] P. E. Danielsson, “Euclidean distance mapping,” Computer Graphics and image processing, Vol. 14, No. 3, pp. 227-248, 1980.