KIPS Trans. Softw. and Data Eng.
Vol.11, No.5 pp.189~196
ISSN: 2287-5905 (Print), ISSN: 2734-0503 (Online)

https://doi.org/10.3745/KTSDE.2022.11.5.189

An Improved Skyline Query Scheme for Recommending Real-Time User Preference Data Based on Big Data Preprocessing

JiHyun Kim† · Jongwan Kim††

ABSTRACT

Skyline query is a scheme for exploring objects that are suitable for user preferences based on multiple attributes of objects. Existing skyline queries return search results as batch processing, but the need for real-time search results has increased with the advent of interactive apps or mobile environments. Online algorithm for Skyline improves the return speed of objects to explore preferred objects in real time. However, the object navigation process requires unnecessary navigation time due to repeated comparative operations. This paper proposes a Pre-processing Online Algorithm for Skyline Query (POA) to eliminate unnecessary search time in Online Algorithm exploration techniques and provide the results of skyline queries in real time. Proposed techniques use the concept of range-limiting to existing Online Algorithm to perform pretreatment and then eliminate repetitive rediscovering regions first. POAs showed improvement in standard distributions, bias distributions, positive correlations, and negative correlations of discrete data sets compared to Online Algorithm. The POAs used in this paper improve navigation performance by minimizing comparison targets for Online Algorithm, which will be a new criterion for rapid service to users in the face of increasing use of mobile devices.

Keywords : Skyline Query, Nearest Neighbor, Online Skyline Query, Block Nested Loop, Data Preprocessing

빅데이터 전처리 기반의 실시간 사용자 선호 데이터 추천을 위한 개선된 스카이라인 질의 기법

김 지 현† · 김 종 완††

요 약

스카이라인 질의(Skyline Query)는 객체의 다중 속성을 기준으로 사용자 선호에 적합한 대상을 탐색하는 기법이다. 기존 스카이라인 질의는 탐색 결과를 일괄처리(batch processing)로 반환하지만, 대화형 앱이나 모바일 환경의 등장으로 실시간 탐색 결과의 필요성이 증가하였다. 스카이라인을 위한 온라인 알고리즘(online algorithm)은 객체의 반환 속도를 향상해 실시간으로 선호 객체를 제공한다. 하지만 객체 탐색 과정에서 기존에 탐색한 영역을 재방문하여 반복 비교하는 불필요한 연산 시간이 소요된다. 본 논문은 온라인 알고리즘에서 불필요한 탐색 시간을 제거하여 스카이라인 질의 결과를 실시간으로 제공하기 위한 스카이라인 온라인 전처리 알고리즘을 제안한다. 제안 기법은 기존의 온라인 알고리즘에서 전처리를 수행함으로써 반복적으로 재탐색 되는 영역을 미리 제거하여 탐색 성능을 향상하였다. 실험 결과, 기존 온라인 알고리즘과 비교 시 이산 데이터 집합의 표준 분포, 편향 분포, 양의 상관 및 음의 상관분포에서 향상된 성능을 보였다. 제안 기법은 비교 대상을 최소화하여 탐색 성능을 향상하므로 모바일 장치의 사용이 증가하는 현실에서 사용자들에게 신속한 서비스를 제공할 수 있는 새로운 기준이 될 것이다.

키워드 : 스카이라인 질의, 최근접 이웃, 온라인 스카이라인 질의, 블록 중첩 루프, 데이터 전처리

1. 서 론

스마트 기기가 발전함에 따라 대규모(volume)의 멀티미디어 콘텐츠가 빠르게(velocity) 생산되며, 각종 센서와 SNS로 인해 다양한(variety) 자료수집이 가능하다. 빅데이터(big data)가 생성되고 활용도가 높아지면서 데이터의 의미를 파악하기 위해 데이터 마이닝(data mining)이 데이터 분석 기법으로 사용된다. 특히, 데이터의 속성이 다양한 경우에는 사용자 선호도에 부합하는 데이터 분석 및 탐색 기법이 필요하다.

스카이라인 질의(Skyline Query)[1] 는 다차원 속성을 비교하여 사용자 선호도에 맞는 데이터(또는 객체)를 추천하는

※ 이 논문은 2018년 대한민국 교육부와 한국연구재단의 지원을 받아 수행된 연구임(NRF-2018S1A5A8027993).
† 비 회 원 : 한양대학교 인공지능학과 석사과정
†† 종신회원 : 삼육대학교 인공지능융합학부 교수
Manuscript Received : December 7, 2021
Accepted : December 31, 2021
* Corresponding Author : Jongwan Kim(kimj@syu.ac.kr)

대표적인 기법으로 다른 속성에 의해 지배되지 않는 데이터를 검색한다. 여기서 지배된다는 것은 하나의 객체가 다른 객체에 대해 모든 속성에서 좋지 않은 값을 가지는 상태를 의미한다.

스카이라인 질의는 사용자의 선호도를 만족시키는 데이터를 반환하기 위해 불완전 데이터 검색[2,3], 이동객체 탐색[4], 하둡을 이용한 빅데이터 처리[5] 등에 사용된다. 명확한 속성이 제공되는 경우에는 선호도를 중심으로 순위를 구하는 문제[6,7]에 사용되어 다양한 연구 분야를 포함하고 있다.

스카이라인 질의는 선호 대상을 모두 검색한 후에 일괄적으로 결과를 제공하므로 사용자가 선호하는 대상을 즉시 확인하기에는 시간이 소요된다. 그러나 모바일 환경이 발전함에 따라 대화형 앱(interactive application)에서 선호 데이터를 제공할 필요성이 증가하고 있다. 이때, 사용자는 신속한 질의 결과를 요구하므로 스카이라인 질의를 통한 선호 데이터는 실시간으로 사용자에게 제공되어야 한다. 대화형 앱에서 실시간으로 스카이라인 질의 결과를 제공하기 위한 연구는 온라인 알고리즘(online algorithm)[8,9]이 대표적이다.

온라인 알고리즘은 최근접 이웃(nearest neighbor) 탐색 기법을 사용하여 사용자와 상호 작용이 필요한 대화형 앱에서 실시간으로 선호 객체를 반환한다. 알고리즘은 탐색영역에 포함된 데이터의 개수에 따라 연산 속도가 달라지는데 만약, 탐색영역에 데이터 분포가 밀집되면 한 번 방문한 영역을 여러 번 재탐색 및 비교하기 때문에 탐색 성능이 저하된다. 예를 들어 전체 집합 S = {A, B}에서, A = {a, b, c}이고 B = {b, c, d}일 때, S를 탐색한다면, A, B의 원소 중 b, c 원소가 중복되어 두 번 접근이 필요하다. 반면에 S = {A, B}, A = {a, b}이고 B = {c, d}일 경우, S 부분을 탐색할 때, 중복되는 원소들이 없으므로 각 원소는 한 번만 접근됨을 알 수 있다.

본 논문에서는 객체 중첩에 따른 탐색영역 재방문 및 비교 횟수 증가 문제를 해결하기 위해 스카이라인 온라인 전처리 알고리즘인 POA (Pre-processing Online Algorithm for Skyline Query)를 제안한다. POA는 합성 함수(composite function) 기반의 범위 제한기법을 사용하여 온라인 알고리즘에서 방문 영역을 축소함으로써 객체 중첩 문제를 해결함과 동시에 연산 속도를 향상한다.

논문의 구성은 다음과 같다. 2장에서는 기존 스카이라인 질의 알고리즘들을 설명한다. 3장에서는 논문에서 제안하는 POA 알고리즘을 소개하고 이를 기존 온라인 알고리즘에 적용할 때 나타나는 효과를 기술한다. 4장에서 POA와 선행 온라인 알고리즘 기법 간의 연산 속도를 비교하여 제안 기법의 성능을 증명한다. 마지막으로 5장은 결론을 서술한다.

2. 관련 연구

2.1 스카이라인 질의

스카이라인 질의는 Fig. 1과 같이 2차원 공간에 데이터의 속성을 나타내면서 원점 (0, 0)을 중심으로 각 축에서 가장 작은 속성을 만족하는 데이터를 검색한다. 일반적으로 해변에서 가깝고(x축, distance) 숙박비(y축, price)가 저렴한 두 가지 속성의 호텔을 탐색하는 예가 사용되는 것과 같이 본 논문은 전처리를 이용한 스카이라인 탐색 성능 향상을 제안하므로 스카이라인 질의 정의에 따라 원점에 가까운 작은 값을 탐색하며 속성은 2차원으로 가정한다.

Fig. 1은 두 속성을 단순한 수치로만 표현하여 스카이라인 질의를 수행한다.

스카이라인 질의의 특성은 다음과 같다. 전체 데이터 집합 Ds에 대하여 $P \in D_s$ 일 때, $P = \{p.d_i, ..., p.d_j\}$이며, j가 $1 \le j \le d$ 일 때 $p.d_j$는 객체 p의 j번째 차원(Dimension)을 의미한다.

데이터가 스카이라인 질의 결과에 포함되기 위해서는 다음과 같이 두 가지 조건이 필요하다.

[조건 1] 지배(dominate) 관계

데이터 $p, r \in D_s$에 대하여, 다음의 조건 (1)과 (2)를 만족하면 p는 r을 지배한다.

(1) $\exists j \in [1,d], p.d_j < r.d_j$

(2) $\forall i \in [1,d], p.d_i \le r.d_i$

[조건 2] 스카이라인 객체(skyline object)

데이터 $P \in D_s$에 대하여, $\nexists r \in D_s$인 $r < p$를 만족시키는 모든 p는 스카이라인 객체가 된다.

2.2 블록 중첩 루프(BNL) 알고리즘

블록 중첩 루프(BNL, Block Nested Loop)[10]는 초기 스카이라인 질의에서 사용한 직관적인 방법으로 스카이라인 객체 후보군 리스트를 만들고 리스트의 객체와 새로 탐색 되는 데이터 객체를 비교한다.

BNL은 리스트 내부 객체와 데이터를 비교할 때 두 가지 경우가 발생한다. 입력된 데이터 객체가 (1) 리스트 내부 객체에 의해 지배되는 경우와 (2) 지배되지 않는 경우이다. 이때 (1)은 후보군 리스트에 포함되지 못하고 (2)는 리스트에 포함된다.

예를 들어, Fig. 1에서 A(2, 2)의 객체가 들어올 경우, 리스트 내부 객체가 존재하지 않기 때문에 스카이라인 객체 후보 리스트에 추가된다. 다음으로 B(2, 4)가 객체로 들어온다면, 현재 스카이라인 객체 후보군 리스트 {(2, 2)} 와 비교한다. B와 D는 (2, 2)에 의해 지배되므로 후보군 리스트에 포함되지 않는다. 마지막으로 C(1, 4)의 경우 후보군 리스트 {(2, 2)} 와 비교할 때 지배되지 않기 때문에 최종 리스트는 {(2, 2), (1, 4)}가 된다.

BNL은 스카이라인 질의 중 직관적인 방법이지만, 후보군 리스트의 객체와 데이터 객체 전체를 비교하기 때문에 데이터가 많을수록 계산 복잡도가 증가한다.

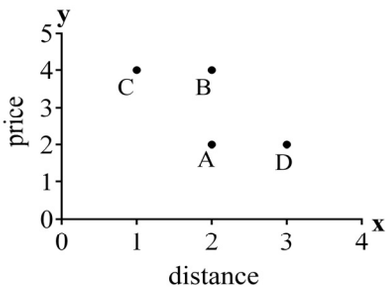


Fig. 1. Block Nested Loop Query

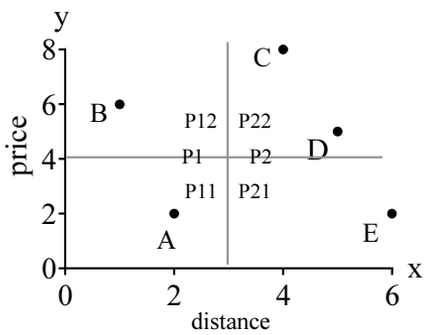


Fig. 2. Divide and Conquer Query

2.3 분할 정복 알고리즘

분할 정복(D&C, Divide and Conquer)[10]은 스카이라인 객체를 탐색하기 위해 탐색 범위 전체를 비교하는 BNL과 달리 중간값을 기준으로 탐색영역을 n개의 부분집합으로 나눈다. 나눠진 부분들에 대해 각각 연산을 수행하고 결괏값을 구한 뒤 해당 값을 가지고 다시 분할 정복을 진행한다. 몇 차례 분할 정복을 시행한 뒤 나눠진 객체들을 합치면 스카이라인 질의 결과가 완성된다.

예를 들어 Fig. 2에서 A부터 E까지 객체들을 입력받고, 전체 객체들에 대해 먼저 x축을 기준으로 중간값을 얻는다. Fig. 2에서는 전체 영역에 대한 중간값이 x = 3이므로 해당 선을 기준으로 전체 데이터를 P1, P2 두 개 영역으로 나눈다. 나눠진 각 영역에서 스카이라인 객체를 구하고 P1에 의해 지배되는 P2 영역의 점들은 모두 제거한다. 현재, 나눠진 부분에서는 모든 점이 스카이라인 객체에 해당하므로 제거할 대상이 없다. 다음으로 y축을 기준으로 중간값을 얻는다. y축에 대해서는 중간값이 y = 4이므로 해당 선을 기준으로 전체 영역을 P11, P12, P21, P22로 나눈다. P11, P12, P21의 경우 지배되는 영역이 없지만, P22의 경우 P11에 의해 지배되므로 제거된다. 분할 정복 과정은 분할 영역이 비어있거나 하나의 객체가 남을 때까지 반복한다. 그러나 D&C는 영역을 반복적으로 나누기 때문에 영역이 크면 탐색 성능이 저하된다.

2.4 최근접 이웃 스카이라인 질의

최근접 이웃(NN, Nearest Neighbor) 스카이라인[8]은

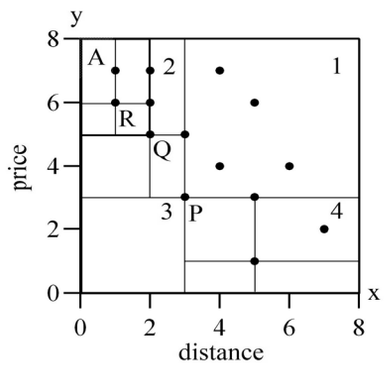


Fig. 3. Nearest Neighbor !Query

일괄처리의 단점을 해결하기 위해 분할 정복 알고리즘과 최근접 이웃 기법을 [11,12]를 결합하여 실시간 스카이라인을 구현하였다.

NN은 D&C처럼 영역을 나누면서 해당 영역에서 스카이라인을 구하는 즉시 반환하기 때문에 대화형 애플리케이션에 적합하다. NN은 전체 데이터 중에서 영점 (0, 0)과 가까운 객체를 찾고 해당 객체를 기준으로 2, 4 사분면을 재연산한다. 이때, NN에 의해 지배되는 1사분면의 객체들은 제거된다.

예를 들어, Fig. 3에서 첫 번째 NN 객체는 전체 영역을 대상으로 탐색하므로 유클리드 거리[13,14]가 $3\sqrt{2}$인 P(3, 3)가 된다. P를 중심으로 사분면을 나누고 2 사분면에서 NN을 찾는다. 즉, (0, 3)부터 (3, 8)의 영역에서 NN 객체를 탐색하여 유클리드 거리가 $\sqrt{29}$인 Q(2, 5)를 NN으로 선택한다. 마지막으로 Q에서 사분면으로 영역을 나누고 2 사분면을 탐색하면 유클리드 거리가 $\sqrt{37}$ 인 R(1, 6)이 NN으로 선택된다. 이때, 영역 A의 객체들은 P, Q, R을 찾는 과정에서 반복적으로 탐색 되어 세 번의 연산 과정에 참여하게 된다. A 영역의 객체가 많을수록 불필요한 연산에 노출되는 객체가 존재하며 탐색 과정이 반복되면서 성능 저하가 발생한다.

BNL, D&C는 일괄처리 방식으로 알고리즘의 시작과 반환에만 I/O가 존재하므로 결과 반환까지 걸리는 시간이 빠르다[2]. 하지만 일괄처리 방식은 사용자가 연산 중간에 결과를 받을 수 없으며 시간이 경과 한 후 최종 스카이라인 객체들을 받기 때문에 질의에 대한 응답을 실시간으로 제공할 수 없다. 예를 들어, 데이터 10,000개에 대한 스카이라인 결과가 100개라면 결과를 모두 구한 뒤 한 번에 반환하므로 사용자의 대기시간이 증가한다.

사용자의 선호도에 실시간 대응하기 위해서는 스카이라인 처리 과정 중 일부를 신속하게 반환해야 한다. BNL, D&C는 일괄처리에 기반하므로 실시간 반환이 안 되며 이를 해결하기 위해 NN이 연구되었다. 그러나 NN은 대상 객체를 재탐색하여 불필요한 연산을 수행하는 단점을 여전히 해결하지 못하였다.

본 논문에서는 NN의 문제를 해결하여 실시간 처리가 가능하면서 불필요한 연산 횟수를 줄인 새로운 스카이라인 질의 기법을 제안한다.

3. 스카이라인 질의 전처리 기법

본 논문에서 제안하는 스카이라인 질의 전처리 기법인 POA는 빅데이터에서 속성을 중심으로 사용자 선호 데이터를 실시간으로 추천하기 위한 기법이다. 제안 기법은 실시간 추천을 지원하는 NN 알고리즘을 수행하기 전에 탐색 범위를 축소하여 연산할 객체 수를 줄인다.

탐색 범위 제한은 다음의 범위 축소 기법에 따른다.

[정의 1] 탐색 범위 축소

한 데이터 객체 A는 x축과 y축 좌표인 (α,β)를 가지며 $(\alpha,\beta)\in\mathbb{Z}$라 할 때, x축과 y축의 범위 제한은 각각 Equation (1), (2)와 같다.

Equation (1)의 함수 g는 x축에 대한 최솟값을 나타내며 α 값이 $y=\beta$ 일 때 β 값을 입력받은 minX 함수보다 작거나 같으면, x축 범위가 최솟값으로 제한되어 탐색 범위가 축소된다.

$(\alpha,\beta)\in\mathbb{Z}$ 일 때,

$$g:(\alpha,\beta)\ \text{if}\ y=\beta\wedge\alpha\le\min X(\beta) \tag{1}$$

Equation (2)의 함수 f는 y축에 대한 최솟값을 나타내며 β값이 $x=\alpha$일 때 α값을 입력받은 minY 함수보다 작거나 같으면 y축 범위가 최솟값으로 제한되어 탐색 범위가 축소된다.

$(\alpha,\beta)\in\mathbb{Z}$ 일 때,

$$f:\beta\ \text{if}\ x=\alpha\wedge\beta\le\min Y(\alpha) \tag{2}$$

위의 탐색 범위 축소에서 minX와 minY 함수는 입력값과 연결된 값 중 최솟값을 찾는 함수이다. 범위 축소는 입력값에 대한 최솟값 정보를 저장 및 참조하여 탐색 범위를 제한함으로써 가능하다.

[Lemma 1] $(\alpha,\beta)\in\mathbb{Z}$ 에 대해서 만약,

$g\circ f:(\alpha,\beta)\mapsto g(f(\alpha,\beta))$ 라면, $(\alpha,\beta)\in skyline\,list$.

Lemma 1은 하나의 객체 (α,β)에 대하여 정의 1에서 제시한 함수 f와 g를 합성 함수 $g\circ f$로 적용할 때 $(\alpha,\beta)\in skyline\,list$가 성립한다고 말한다. 스카이라인 질의는 정의 1에 의해 각 벡터(vector)에서 최솟값을 찾는 방식과 유사하다. 따라서 각 객체를 탐색하여 해당 객체에 따른 최솟값을 찾고 이를 기존 최솟값과 비교하면 스카이라인 객체를 식별할 수 있다.

함수 f의 조건을 만족한 경우 β를 반환하고 함수 g는 반환된 β를 입력값으로 받은 뒤 함수 g의 조건을 충족시킨 결과로 (α,β)를 반환한다. 이렇게 된다면 (α,β)는 스카이라인 객체 후보 자격을 가지게 된다. 즉, $g\circ f$ 결과는 데이터의 탐색 범위를 축소하기 위한 최소 범위를 찾고 스카이라인 객체 후보군에 추가하는 함수이다.

Table 1. Searching with Composite Function

functions not using Lemma 1		functions using Lemma 1	
f	g	f	g
○	○	○	○
○	×	○	×
×	○	×	-
×	×	-	-

Table 2. Operation Steps in POA

(x, y)	x	min y	y	min x	minimum values	Skyline object candidates
(3, 5)	3	5	5	3	x=3, y〈=5 y=5, x〈=3	{(3, 5)}
(2, 5)	2	5	5	2	x=2, y〈=5 x=3, y〈=5 y=5, x〈=2	{(2, 5)}
(2, 7)	2	7	-	-	-	{(2, 5)}
(3, 2)	3	2	2	3	x=2, y〈=5 y=5, x〈=2 x=3, y〈=2 y=2, x〈=3	{(3, 2), (2, 5)}
(4, 2)	4	2	2	4	-	{(3, 2), (2, 5)}

Lemma 1에서 함수 g가 f에 의해 영향을 받도록 설계한 이유는 Table 1의 오른쪽과 같이 경우의 수를 감소함으로써 시간 복잡도에서 이득을 취하기 위해서이다. Table 1에서 합성 함수를 사용하지 않을 때 하나의 객체가 겪을 수 있는 경우의 수는 여덟 가지이다. 하지만 Lemma 1에 따라 불필요한 경우는 연산을 중단하도록 구현하여 경우의 수는 다섯 가지로 감소하며, 결과적으로 연산 성능은 약 38%(3/8)가 향상된다.

Table 2는 POA의 스카이라인 객체 후보 연산 과정을 나타낸다. Lemma 1에 기반하여 각 축에 대한 최솟값을 찾는 과정은 다음과 같다.

먼저, 데이터 (3, 5)가 들어오면 정의 1에 의해 {x = 3일 때 y 〈= 5}, {y = 5일 때 x 〈= 3}의 정보가 들어온다. 첫 번째 입력이므로 스카이라인 객체 후보군에 (3, 5)가 추가된다. 다음으로 데이터 (2, 5)가 들어오면 {x = 2일 때 y 〈= 5}의 정보가 추가되고, 후보군에 들어 있는 y = 5일 때의 정보는 새로 들어올 정보보다 큰 범위를 가지므로 {y = 5일 때 x 〈= 2}의 정보로 갱신된다. 즉, 기존 후보군에 들어있던 데이터 (3, 5)는 {y = 5일 때 x 〈= 2}의 조건을 만족시키지 못하므로 제거되고 조건을 만족시키는 데이터 (2, 5)가 후보군에 추가된다. 이어서, 데이터 (2, 7)이 들어오면 기존의 조건 {x = 2일 때 y 〈= 5}에 어긋나므로 해당 객체에 대한 탐색이 종료된다.

계속해서 데이터 (3, 2)의 경우 {x = 3일 때, y 〈= 5} 조건에 해당하므로 {x = 3일 때, y 〈= 2}로 변경되고 {y = 2일 때, x 〈= 3}의 조건이 추가된다. (4, 2)의 경우 {x = 4일 때, x 〈= 2}의 경우는 만족하지만, {x = 2일 때, y 〈= 3}의 조건

```
Algorithm: POA
Input: Dataset ds
M: max value of integer
Initialize yMin list and xMin list as M
candidate: sets of skyline object candidates
1:  for (x, y) in data:
2:      if y is smaller than yMin[x] :
3:          yMin[x] = y
4:          if x is smaller than xMin[y] :
5:              xMin[y] = x
6:              if x != M and x in candidate:
7:                  delete candidate[x]
9:              insert candidate[x] = y
10: return candidate
```

Fig. 4. POA Algorithm

을 만족시키지 못하므로 함수는 종료되고, 최종적으로 스카이라인 객체 후보로 {(3, 2), (2, 5)}가 반환된다.

POA에 대한 알고리즘은 Fig. 4와 같다. POA 알고리즘은 두 가지 단계로 진행된다. 첫 번째, 모든 데이터에 대하여 현재 들어온 객체의 y 값이 x 값을 색인(index)으로 가지는 저장소의 최소 y 값보다 작다면 yMin[x]의 값을 y로 대체한다 (1~3행).

두 번째, 현재 들어온 객체의 x 값이 y 값을 색인으로 가지는 저장소의 최소 x 값보다 작다면 xMin[y]의 값을 x로 대체한다 (4~5행). 이때 x가 초깃값이 아니면서 후보군에 이미 들어있다면 스카이라인 후보군에서 x를 색인으로 가지는 부분을 제거해야 한다 (6~7행). 8행에서 새로운 객체 (x, y)를 스카이라인 후보군에 저장하고 마지막으로 저장된 스카이라인 후보군, candidate를 반환한다 (10행).

기존 기법인 NN은 최근접 이웃 객체를 결정하기 위해 데이터 집합의 객체를 모두 비교하여 O(n)의 시간 복잡도를 갖는다[15]. 알고리즘 Fig. 4는 객체를 하나씩 선택하여 스카이라인 후보를 구성하므로 NN과 같이 O(n)의 성능을 보인다. 두 기법이 동일한 시간 복잡도를 갖지만, NN은 탐색영역이 변경되어도 기존에 비교했던 객체를 다시 비교하기 때문에 중복비교로 탐색 시간이 오래 걸리며 POA는 중복데이터를 제거하기 때문에 탐색 성능이 빠르다.

POA는 데이터들의 지배 관계를 비교하여 연산에 불필요한 객체들을 미리 제거한다. 따라서 본 알고리즘의 장점은 데이터 개수가 전체 영역의 최대 포함 가능 개수에 가까워질수록 밀집된 데이터가 많아지므로 불필요한 객체가 많이 제거되어 탐색 성능이 향상된다.

4. 실 험

데이터 집합은 NN 알고리즘과 실험 환경을 동일하게 구성

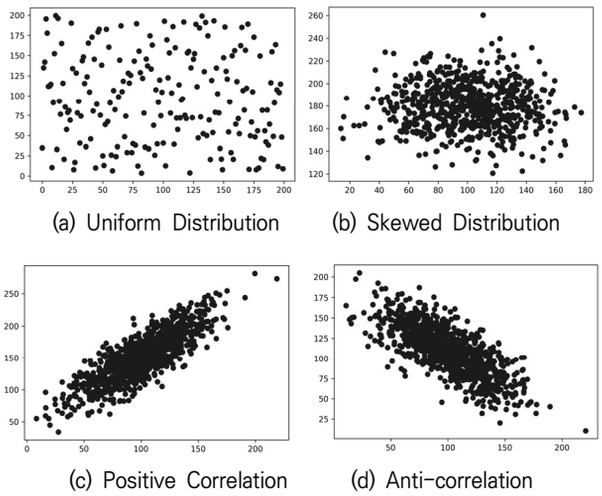


Fig. 5. Discrete Data Distributions

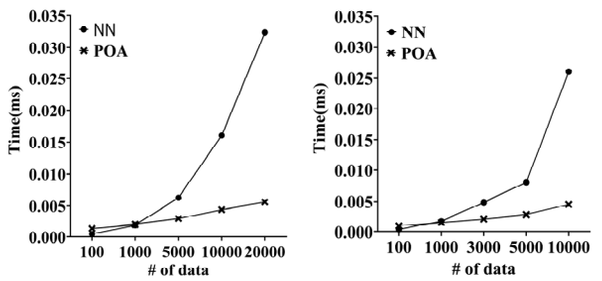
Table 3. Simulation Environments

Category	Contents
System	Intel Xeon Silver 4114 CPU 2.20GHz * 2, RAM 128GB
Pgm. lang.	Python
# of Data	60,000 (Max)
Data set & distributions	Uniform, Skewed, Positive correlation, Anti-correlation
Data region	200 × 200, 300 × 300 Object has discrete distribution uniquely.
# of attributes	2
# of runs for simulation	5,000

하기 위해 데이터 중복을 허용하지 않으며 각 데이터의 좌표는 유일하다고 가정한다. 데이터 집합의 분포는 Fig. 5와 같이 균등 분포, 편향 분포, 양 및 음의 상관관계를 포함하며 각 축은 데이터의 분포 범위를 나타낸다.

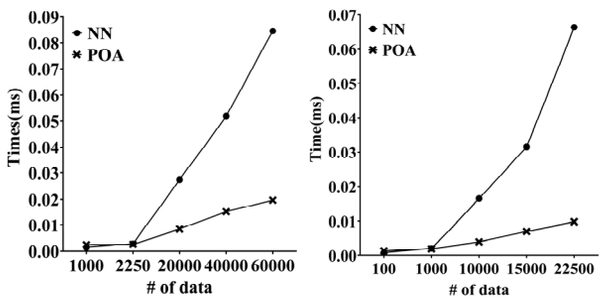
실험 환경은 Table 3과 같다. 실험 데이터는 균등 분포, 편향 분포, 양 및 음의 상관관계로 구성된다. 데이터의 x, y 좌표는 데이터 영역에서 중복되지 않는 유일한 값을 갖는 이산분포(discrete distribution)를 사용한다.

균등 분포와 편향 분포의 경우 데이터 범위와 개수에 따라 다음과 같이 실험의 유형을 달리한다. 첫 번째 실험에서는 x축과 y축의 범위는 200으로 지정하였고 중복이 없다는 가정에 따라 전체 영역으로 들어올 수 있는 데이터 수는 40,000개가 된다. 균등 분포의 경우 데이터 집합은 전체 영역에 대해 값들을 랜덤하게 받아들이므로 데이터들은 Fig. 5A와 같이 영역의 전반에 고르게 분포하게 된다. 편향 분포의 경우 Fig. 5B와 같이 데이터 집합이 형성되기 때문에 전체 영역으로 들어올 수 있는 데이터 개수의 1/4에 해당하는 사각형의 네 모서리 중 한 부분만 실험한다.



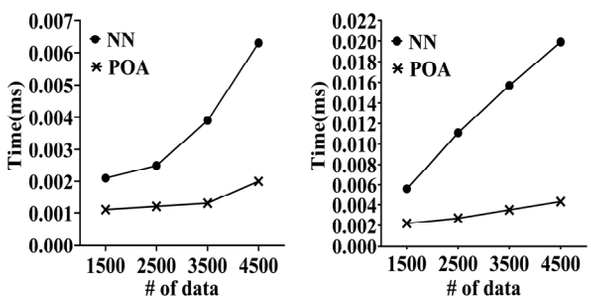
(a) Uniform (b) Skewed

Fig. 6. Performance in a Region 200 × 200



(a) Uniform (b) Skewed

Fig. 7. Performance in a Region 300 × 300



(a) Positive Correlation (b) Anti-correlation

Fig. 8. Performance in Correlations

양의 상관관계를 나타내는 Fig. 5C는 다른 데이터 분포보다 적은 수의 스카이라인 객체를 반환할 수 있으며 음의 상관관계인 Fig. 5D는 기울기 내의 모든 데이터가 스카이라인 객체로 나타날 수 있다. 이처럼 다른 데이터 집합을 포함한 이유는 제안 기법의 성능이 다양한 데이터 분포에서도 긍정적으로 평가되는 것을 보이기 위함이다.

균등 분포의 경우 Fig. 6A와 같이 100개, 1,000개, 5,000개, 10,000개, 20,000개 데이터로 실험을 진행하였다. 실험 결과 NN은 100개 데이터 집합에서 POA보다 빠른 탐색 성능을 나타내며 1,000개까지 유지된다. 데이터 수가 적으면 시작점으로부터 최근접 이웃 객체를 찾는 과정은 1:1 비교이다. 그러나 제안 기법은 객체 수가 적어도 탐색 범위 축소 연산을 수행하기 때문에 상대적으로 NN보다 탐색 시간이 소요된다. 데이터 객체가 1,000개 이상부터 POA가 빠른 성능을 나타내며 데이터 집합이 증가해도 성능의 편차는 크지 않다. 이에 반해, NN은 데이터 집합이 증가할 때 성능이 급격하게 나빠지며 이는, NN이 탐색영역 데이터를 반복적으로 비교하기 때문이다.

편향 분포의 경우 Fig. 6B와 같이 100개, 1,000개, 3,000개, 5,000개, 10,000개 일 때의 POA와 NN의 실행 시간을 비교하였다. 편향 데이터 집합에서도 100~1,000개까지 데이터에서 NN의 탐색 성능이 빠르며 그 이후에는 POA의 성능이 완만하게 증가하지만, 여전히 NN보다 좋은 성능을 나타낸다. 편향 분포는 데이터 집합의 편향 정도에 영향을 받지만, 실험 결과는 논문에서 해결하고자 하는 NN의 단점이 동일하게 나타내고 있다. 실험 Fig. 6B에서는 1,000~5,000개 구간의 변화를 보기 위해 3,000개 데이터를 포함하였으나 데이터 증가에 따른 성능 변화는 크지 않았다. 또한, 표준 분포와 달리 20,000개 데이터 집합은 측정하지 않았지만, 데이터가 증가할수록 성능이 더 나빠질 것을 예측하기에는 큰 어려움이 없다.

두 번째 실험에서는 Fig. 7과 같이 데이터 공간을 x, y축으로 확장하여도 같은 성능 그래프를 나타내는지 확인하기 위해 x축과 y축의 범위를 각각 300개로 지정하였다. 전체 영역의 이산 데이터 개수는 x × y로 90,000개가 된다. 균등 분포 데이터는 1,000개, 2,250개, 20,000개, 40,000개, 60,000개의 데이터로 진행하였다. 편향 분포의 경우 최대 데이터 개수의 1/4개까지인 100개, 1,000개, 10,000개, 15,000개, 22,500개 일 때의 POA와 기존 NN의 실행 시간을 비교하였다.

이산 균등 분포인 Fig. 7A에서 POA는 NN보다 데이터 개수가 2,250개 일 때부터 성능이 향상되며 데이터의 밀집도가 가장 높을 때 4.4배(POA: 0.019, NN: 0.084) 빨라졌다. 또한 Fig. 7B의 편향 분포에서는 1,000개를 기점으로 성능 향상이 나타났으며 기존 NN과 비교하면 처리 속도가 최대 7.3배(POA: 0.009, NN: 0.066) 향상되었다. 실험에서 NN의 성능이 일정한 데이터 집합까지 POA보다 빠르다. 이는 적은 데이터 개수에서 NN의 중복비교 연산이 POA의 중복비교 방지 연산 횟수보다 적게 수행되는 것을 의미한다.

세 번째 실험은 Fig. 8과 같이 양과 음의 상관관계에 대한 실험이다. 양의 상관관계는 하나의 속성이 증가함에 따라 다른 속성도 증가하는 비례 관계이며, 음의 상관관계는 하나의 속성이 감소하면 다른 속성은 증가하는 반비례 관계이다. 실험은 Fig. 8과 같이 두 상관관계에 대해 데이터 집합을 1,500개, 2,500개, 3,500개, 4,500개일 때 진행하였고 POA와 기존 NN의 실행 시간을 비교하였다.

상관관계에 POA를 적용한 경우에는 Fig. 8과 같이 처리 시간에 있어서 기존 NN과 확연한 차이를 보인다. 특히, NN은 음의 상관관계에서 성능이 급격하게 나빠지고 POA는 완

Table 4. Performance Comparison in Normalized Data Set

Category	Uniform	Skewed	Correlation	Anti-correlation
Performance of [POA + NN] Compared to the NN	0.33 times	1.4 times	3 times	4.6 times

만히 증가하여 양의 상관관계와 대조를 이룬다.

제안 기법의 실험 결과, 균등 분포의 경우 전체 영역의 2.5% 이상 데이터가 분포되어 있으면 POA의 성능이 좋아진다. 즉, Fig. 6에서는 데이터의 개수 1,000, Fig. 7에서는 데이터의 개수가 2,250개 일 때 성능이 향상된다. 편향 분포의 경우 1,000개를 기점으로 POA가 좋은 성능을 보였으며 양 상관과 음 상관에서는 Fig. 8과 같이 처음부터 NN보다 빠른 수행시간을 나타낸다.

본 실험에서는 이산분포 데이터의 특성상 탐색영역의 크기에 따라 특정 데이터 분포가 최대로 가질 수 있는 데이터의 개수가 다르므로 실험의 다양성을 보이기 위해 각 분포에서 데이터 집합의 수를 달리하여 실험하였다. 따라서 상이한 데이터 개수에 대한 성능지표를 통일하기 위해 각 실험의 높은 결과에 대해 데이터 개수를 기준으로 Table 4와 같이 결과를 정규화하였다.

실험 결과는 POA가 기존 NN보다 표준 분포에서 0.33배만큼 빠르고 편향 분포 대비 1.4배의 성능을 보였다. 사향 데이터 집합인 양의 상관관계에서는 3배만큼 빨랐으며 음의 상관관계는 4.6배만큼 빨랐다. POA는 상관관계를 포함하여 데이터의 편향성이 클수록 높은 성능을 나타냈으며 표준 분포에서도 성능이 차별화되고 있다.

5. 결 론

스카이라인 질의는 사용자의 선호 데이터를 탐색하지만, 스카이라인 결과를 일괄적으로 제공함으로써 모바일 환경에서 선호 데이터를 실시간으로 제공하기 어렵다.

NN을 사용한 온라인 스카이라인 알고리즘은 데이터의 반환 속도를 향상하여 실시간 선호 데이터 제공이 가능하다. 그러나 반복적인 비교 연산은 성능 향상의 병목현상으로 작용하였다.

본 논문에서 제안한 POA 알고리즘은 스카이라인 질의를 수행할 때 지배 관계를 미리 비교하여 NN에서 처리할 탐색 범위를 제한하였다. 최종적으로는 객체 수가 축소되며 NN에서 이웃 객체들에 대한 비교 연산 횟수를 줄여줌으로써 성능이 향상되었다.

실험 결과는 다양한 데이터 세트에서 제안 기법인 POA 알고리즘이 기존보다 확연한 성능 차이가 있음을 나타냈다. 결과적으로 제안 기법은 스카이라인 질의를 사용하여 사용자 선호 데이터를 검색할 때 실시간 결과를 제공함으로써 처리 성능뿐만 아니라 사용자의 만족도 향상에 기여할 것이다.

References

[1] S. Borzsonyi, D. Kossmann, and K. Stocker, “The skyline operator,” In *Proceedings of IEEE Conference on Data Engineering*, Heidelberg, Germany, pp.421-430, 2001.

[2] Y. Gulzar, A. A. Alwan, and S. Turaev, “Optimizing skyline query processing in incomplete data,” in *IEEE Access*, Vol.7, pp.178121-178138, 2019.

[3] Z. Cai, X. Cui, X. Su, L. Guo, Z. Liu, and Z. Ding, “Continuous road network-based skyline query for moving objects,” in *IEEE Transactions on Intelligent Transportation Systems*, Vol.22, No.12, pp.7383-7394, 2021.

[4] X. Yingyuan, X. Jiao, H. Wang, C. H. Hsu, Li Liu, and W. Zheng, “Efficient continuous skyline query processing in wireless sensor networks,” *Sensors*, Vol.19, No.13, pp.2902, 2019.

[5] C. Kalyvas and M. Maragoudakis, “Skyline and reverse skyline query processing in SpatialHadoop,” *Data Knowledge and Engineering*, Vol.122, pp.55-80, 2019.

[6] G. Stoupas, A. Sidiropoulos, D. Katsaros, and Y. Manolopoulos, “Skyline-based university rankings,” *ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium*, pp.347-352, 2020.

[7] Z. Zheng, K. Ruan, and M. Yu, “k-dominant Skyline query algorithm for dynamic datasets,” *Frontiers of Computer Science*, Vol.15, No.151602, pp.1-6, 2021.

[8] D. Kossmann, F. Ramsak, and S. Rost, “Shooting stars in the sky: An online algorithm for skyline queries,” *Proceedings of the 28th Very Large Data Bases Conference*, pp.275-286, 2002.

[9] T. Erlebach, F. H. Liu, H. H. Liu, M. Shalom, W. H. Wong, and S. Zaks, “Complexity and online algorithms for minimum skyline coloring of intervals,” *Theoretical Computer Science*, Vol.788, pp.66-78, 2019.

[10] 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.

[11] N. Roussopoulos, S. Kelley, and F. Vincent, “Nearest neighbor queries,” In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pp.71-79, 1995.

[12] F. V. Alejandro and M. David, "Guarantees on nearest-neighbor condensation heuristics," *Computational Geometry*, Vol.95, No.1, pp.101732, 2021.
[13] P. E. Danielsson, "Euclidean distance mapping," *Computer Graphics and Image Processing*, Vol.14, No.3, pp.227-248, 1980.
[14] T. Bellitto, A. Pêcher, and A. Sédillot, "On the density of sets of the Euclidean plane avoiding distance 1," *Discrete Mathematics & Theoretical Computer Science*, Vol.23, No.1, 2021.
[15] R. Weber, H. J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," *Proceedings of the 24th VLDB Conference New York*, USA, pp.194-205, 1998.

김 지 현
https://orcid.org/0000-0003-3313-4382
e-mail : jhkhy@hanyang.ac.kr
2020년 성신여자대학교 컴퓨터소프트웨어학과(학사)
2021년 ~ 현 재 한양대학교 인공지능학과 석사과정
관심분야 : Skyline Query, Mobilenet, Lightweight System

김 종 완
https://orcid.org/0000-0003-4716-8380
e-mail : kimj@syu.ac.kr
2016년 ~ 현 재 삼육대학교 인공지능융합학부 교수
관심분야 : Big Data & Smart Data, Data Mining, Machine & Deep Learning