http://dx.doi.org/10.3745/KIPSTD.2011.18D.5.313

위치 제약 조건을 고려한 효율적인 스카이라인 계산

김 지 현[†] · 김 명[††]

요 약

다차원 데이터 집합에서 서로 지배되지 않는 데이터로 구성된 부분 집합을 스카이라인이라고 한다. 스카이라인 계산은 다차원 데이터를 대상으로 한 의사결정에 유용한 연산이다. 그러나 스카이라인이 지나치게 큰 경우 이를 의사결정에 활용하기 어려울 수 있다. 본 연구에서는 사용자가 제시하는 원점의 이동, 원점으로부터의 각도와 거리 정보를 반영하여 스카이라인의 일부를 효율적으로 구하는 방법을 모색하였다. 제안한 알고리즘은 스카이라인에 속하지 않는 데이터를 신속하게 제거해가며, 사용자의 요구를 점진적으로 반영할 수 있다는 특징을 갖는다. 알고리즘의 효율성은 실험을 통해 검증하였다.

키워드 : 다차원 데이터분석, 스카이라인 계산, 의사결정

Efficient Computation of a Skyline under Location Restrictions

Jihyun Kim[†] · Myung Kim[††]

ABSTRACT

The skyline of a multi-dimensional data set is a subset that consists of the data that are not dominated by other members of the set. Skyline computation can be very useful for decision making for multi-dimensional data set. However, in case that the skyline is very large, it may not be much useful for decision making. In this paper, we propose an algorithm for computing a part of the skyline considering location restrictions that the user provides, such as origin movement, degree ranges and/or distances from the origin. The algorithm eliminates noncandidate data rapidly, and returns in order the skyline points that satisfy the user's requests. We show that the algorithm is efficient by experiments.

Keywords : Multi-dimensional Data Analysis, Skyline Computation, Decision Making

1. 서 론

다차원 데이터 집합 A가 있다고 하자. A는 d차원 데이터로 구성되어 있으며, $p=(p_1, p_2, \cdots, p_d)$와 $q=(q_1, q_2, \cdots, q_d)$는 A에 속한 두 데이터라고 하자. 각 차원 i, $1 \le i \le d$, 에 대해 $p_i \le q_i$ 이고, 적어도 한 차원 j, $1 \le j \le d$, 에 대해 $p_j < q_j$ 를 만족하면 p는 q를 지배(dominate)한다고 한다. A에 속한 다차원 데이터 중에서 집합 내의 다른 어떤 데이터에게도 지배받지 않는 데이터 부분집합을 A의 스카이라인 (skyline) 이라고 한다[2]. 예를 들어, $A=\{(1,3),(2,4),(3,2),(4,3),(5,4)\}$ 라고 하자. 여기서 집합의 다른 어떤 점에도 지배되지 않는 점들은 $(1,3)$과 $(3,2)$뿐이므로, A의 스카이라인은 $\{(1,3),(3,2)\}$가 된다. 다른 예를 들어보자. 해수욕장 근처에 n개의 호텔, H_i, $1 \le i \le n$,이 있다고 하자. 각 H_i는 $(dist_i, view_i, price_i)$ 로 구성된 3차원 속성으로 표현될 수 있다. 여기서, $dist_i$는 해변으로부터의 거리, $view_i$는 전망이 좋은 정도, $price_i$는 가격이라고 하자. 이 호텔들의 3차원 속성 집합의 스카이라인은 해변에서 가까우면서 전망이 좋고 가격이 저렴한 것들이므로, 사람들이 가장 선호하는 호텔 집합이라고 볼 수 있다.

이와 같이 스카이라인은 다차원 데이터를 대상으로 의사결정에 유용하게 쓰일 수 있다. 방대한 다차원 데이터로부터 스카이라인을 구해 주면, 사용자는 이로부터 최종 의사결정을 할 수 있기 때문이다. 다차원 데이터 집합으로부터 스카이라인을 효율적으로 계산하는 연구는 최근 활발하게 진행되어 왔다[1~10]. 이 연구결과들은 스카이라인의 크기에 관계없이 스카이라인 전체를 계산한다. 그러나 스카이라인이 지나치게 큰 경우 이는 사용자의 의사결정에 도움이 되기 어려울 수 있다.

† 준 회 원 : 이화여자대학교 컴퓨터공학과 박사과정
†† 종신회원 : 이화여자대학교 컴퓨터공학과 교수
논문접수 : 2011년 5월 2일
수 정 일 : 1차 2011년 6월 28일
심사완료 : 2011년 6월 28일

본 연구에서는 다차원 데이터 집합으로부터 스카이라인을 구하기 전에 사용자의 위치 제약에 대한 요구 조건을 반영하여 스카이라인의 일부만을 신속하고 점진적으로 구할 수 있도록 하는 방법을 모색하였다. 사용자는 다차원 공간의 원점을 변경할 수도 있고, 원점으로부터 특정 각도 이내에 속한 스카이라인만을 구할 수 있도록 할 수도 있으며, 원점으로부터의 거리나 스카이라인에 속할 데이터의 개수도 지정할 수 있도록 하였다. 사용자의 요구조건을 스카이라인 계산 이전에 반영함으로써, 스카이라인 계산 시간을 크게 줄일 수 있고, 의사결정에 좀 더 최적화된 데이터만을 추출할 수 있게 된다. 본 논문의 구조는 다음과 같다. 2절에서 기존 연구를 소개하고, 3절에서는 스카이라인 계산 알고리즘을 제시한다. 4절에서 실험을 통해 알고리즘의 성능을 분석하고, 5절에서 결론을 맺는다.

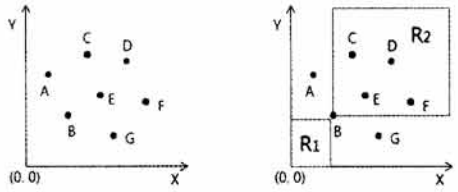
2. 기존 연구

다차원 데이터 집합에서 스카이라인을 구하는 기본적인 알고리즘이 [7]에 소개되어 있다. 이는 데이터 집합이 주기억장치에 모두 저장가능하다고 가정한다. 방대한 양의 데이터를 다루는 기법들은 최근 10여 년간 개발되어 왔다[1~10]. [2][3]에서는 데이터를 하드디스크에 저장해 놓고 분할정복 방법으로 스카이라인을 구하거나, 주기억장치에 윈도우를 두고 스카이라인 점들을 저장해가는 방법을 소개한다. [5]에서 소개한 $LESS$ 알고리즘은 외부 정렬과 필터링을 사용하여 스카이라인에 속하지 않을 데이터들을 신속하게 제거하는 방법을 소개한다. [1]에서 소개한 $SaLSa$ 알고리즘은 서버 클라이언트 환경을 가정하고, 서버에서 다차원 데이터를 정렬한 후 클라이언트 측으로 전송하며, 클라이언트 측으로 보내는 데이터의 양을 최소화하는 방법을 제시하였다. [9][10]에서는 점진적으로 스카이라인에 속한 데이터를 계산해서 반환하는 알고리즘을 제시하였고, [6]에서는 비트맵과 인덱스를 사용하여 스카이라인을 점진적으로 구하는 방법을 소개한다.

본 연구에서는 $LESS$[5]의 방법을 확장하여 샘플링을 통해 더욱 효율적인 필터를 제공하고, 사용자 요구조건에 최적인 스카이라인을 점진적으로 구해가는 환경에서 $SaLSa$ [1]의 정지점 기법을 변형하여 활용한 효율적인 스카이라인 계산 기법을 소개하고자 한다.

3. 제안하는 스카이라인 계산 알고리즘

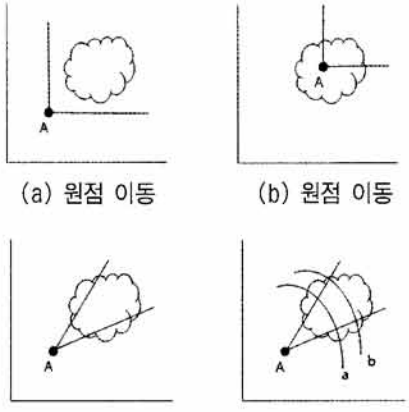
본 연구에서 제안하는 알고리즘을 소개하기 위해, 우선 (그림 1) 2차원 데이터를 예로 하여, 스카이라인 관련 기초지식을 살펴보자. (그림 1)의 (a)에 7개의 2차원 데이터가 도식화되어 있다. 여기에서 스카이라인에 해당되는 데이터는 A, B, G 이다. (그림 1)의 (b)에는 데이터 B에 지배받는 데이터들(영역 R_2)이 나타나 있다. B가 스카이라인에 포함되기 위해서는 영역 R_1이 비어 있어야 한다.



(a) 데이터 (b) B에 지배받는 데이터

(그림 1) 다차원 데이터와 스카이라인

본 연구에서는 사용자로부터 위치 관련 요구 조건을 받아 그 범위에 속한 스카이라인의 일부를 계산하며, 사용자 요구조건에 최적인 것들부터 점진적으로 반환하는 알고리즘을 개발한다. 사용자의 요구조건은 (그림 2)에 나타나 있다. (a)와 (b)처럼 원점을 이동할 수 있고, (c)와 같이 각도 제한을 둘 수 있으며, (d)와 같이 원점으로부터 거리 a까지 계산하거나 거리 b까지 추가로 계산할 수 있도록 한다. 제안한 알고리즘은 (그림 3)에 있다.



(a) 원점 이동 (b) 원점 이동

(c) 각도 제한 (d) 점진적 계산 범위 조정

(그림 2) 사용자 요구조건

스카이라인 계산 알고리즘

[입력] A: 다차원 데이터 집합, O: 이동한 원점 좌표, F : 거리 함수, d_1, d_2: 각도, s: 샘플 개수

[출력] 스카이라인 집합 K

[단계 1] (1) 집합 A로부터 사용자의 위치 요구조건(새로운 원점 O를 기준으로 하여 d_1과 d_2의 사이)을 만족하는 샘플 데이터를 s개 읽어 들이고, 그 집합을 S라 하자. (2) S의 원소들 중에서 O에 가장 가까우면서 d_1과 d_2의 중간에 위치한 원소 p를 구한다.

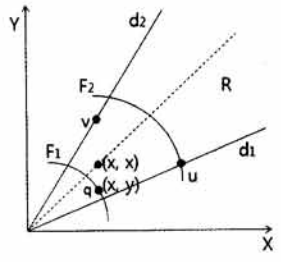
[단계 2] (1) 집합 A를 스캔하면서 사용자의 위치 요구조건을 만족하면서 p 에 지배받지 않는 데이터들을 추출하여 집합 B를 만든다. 이 때, B의 각 원소에 대해 O 기준 새 좌표와 유클리드 거리 F값이 함께 구해진다. (2) B의 원소들을 F값의 오름차순으로 정렬한다.

[단계 3] 집합 B를 스캔하면서 스카이라인을 구하여 집합 K에 저장해 간다. 스카이라인 점을 구할 때마다 정지선(stop line)을 갱신하고, 스캔 중인 원소 $b \in B$의 F값이 정지선을 넘게 되면 알고리즘을 종료한다.

(그림 3) 스카이라인 계산 알고리즘

다음은 제안한 알고리즘에 대한 설명이다. [단계 1]은 집합 A에서 샘플링을 통해 한 원소 p를 구한다. p는 최종 스카이라인 집합 K에 포함되지 않을 원소들을 가능한 많이 지배하는 원소이다. 참고로, 데이터 a, $a \in A$, 의 F값은 O를 기준으로 하여 유클리드 거리로 계산하고, a의 각도는 $\tan(d_1) \leq \tan(a) \leq \tan(d_2)$ 로 점검한다. [단계 2]에서는 집합 A에 p를 기준으로 필터링을 적용하여 집합 B를 구성한다. B의 데이터들은 F값의 오름차순으로 정렬된다.

[단계 3]에서는 집합 B의 데이터를 순서대로 하나씩 (예, b) 스캔하면서, b가 현재 K에 속한 스카이라인 점들에 지배되지 않는다면 b를 K에 포함시킨다. K에 원소가 하나씩 포함될 때마다 정지선 (stop line) 이 다음과 같이 갱신된다. 원점 O로부터의 거리가 F_1인 q가 스카이라인 점으로 판정되었고, 그 점이 (그림 4)에서 (x, y)라고 하자. 이 경우 그림에서 R로 표시된 회색 영역에 있는 데이터는 스카이라인이 될 수 없다. 따라서 q가 스카이라인 점으로 판정될 때, 각도 d_1과 d_2에 닿는 점 u와 v 중에서 원점 O로부터 더 멀리 떨어진 v의 F값(그림에서 F_2)으로 정지선을 갱신해 놓는다. B를 스캔하다가 현재 처리 중인 b의 F값이 정지선을 넘으면 b와 그 이후의 데이터는 스카이라인이 될 수 없으므로 알고리즘을 종료한다.



(그림 4) 정지선의 계산

제안한 알고리즘은 다음과 같은 특징을 갖는다. 첫째, 입력 데이터 집합인 A는 분량이 방대하여 하드디스크에 저장되어 있다고 해도, 사용자의 요구조건이 반영되고 샘플 데이터 p로 선별된 집합 B는 주기억장치에 저장될 수 있을 정도로 작다고 가정해도 좋다. 참고로, 샘플로 필터링하는 아이디어는 $LESS$[5]의 알고리즘과 유사하나 본 연구에서는 사용자 요구조건을 활용하여 대상 데이터를 더욱 많이 감소시켜 알고리즘의 속도를 높였다.

둘째, 원점을 O로 변경하는 것과 각도를 사용하는 것은 집합 A를 첫 스캔할 때 모두 처리되며, 계산이 간단하므로 오버헤드가 거의 발생하지 않는다. [단계 1]에서 많은 데이터가 제거되어 [단계 2]에서의 정렬 시간이 크게 감소된다.

셋째, [단계 3]에서 정지선을 사용함으로써 [단계 2]에서 미처 제거되지 않은 데이터들이 제거된다. 최적의 정지선은 원점 O에서 가까우면서 d_1과 d_2사이의 중앙에 위치한 스카이라인 점이 확인될 때 정해지므로, 최종 스카이라인 K가 구성되는 초기 단계에 확정된다. 본 논문의 정지선은 $SaLSa$[1]의 정지점과는 다르다. $SaLSa$에서 정지를 위한 F값은 최적으로 정해져 있으나, 이는 방대한 양의 원본 데이터 A 전체를 필터링 없이 정렬한 후에야 적용되는 것이다. 또한 $SaLSa$에서의 데이터 정렬 순서로는 사용자 요구조건에 최적인 순서대로 스카이라인 원소들이 구해지지 않는다.

넷째, 제안한 알고리즘은 사용자가 제시한 원점에서 가장 가까운 스카이라인 원소들부터 계산해 내므로, 사용자는 구하고자 하는 스카이라인 원소의 개수나 원점으로부터의 거리도 제시할 수 있고, 이는 알고리즘에 점진적으로 반영될 수도 있다.

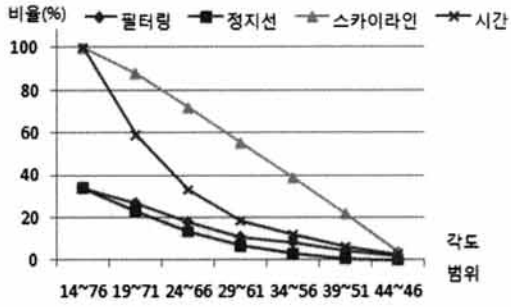
종합하면, 제안하는 알고리즘은 기존의 알고리즘들인 $LESS$[5] 와 $SaLSa$[1]의 장점을 포함하고, 사용자의 요구조건에 최적인 스카이라인 데이터부터 순서대로 스카이라인을 구해주어, 스카이라인 계산 속도를 크게 향상 시키고, 사용자가 다차원 데이터의 의사 결정을 용이하게 한다.

4. 성능 평가

알고리즘의 성능평가는 2.40GHz 인텔 i5 CPU와 4.0GB 메모리를 장착한 PC에서 C언어와 .NET 환경에서 하였다. 평가 목적은 다양한 분포의 데이터와 사용자의 위치 요구조건에 대해 [단계 1]의 샘플링 효과, [단계 3]의 정지선 사용 효과를 점검하고, 전체 스카이라인과 계산된 스카이라인의 원소 개수간 비율 대비 실행시간을 비교하는 것이다.

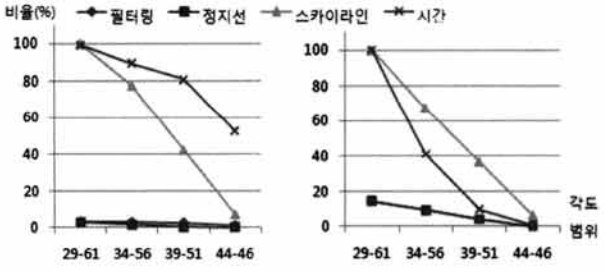
실험 데이터로는 [2]에서와 같이 데이터를 합성하여 다차원 데이터를 생성하였다. 데이터 분포는 스카이라인 연구에서 주로 사용되는 (T_1) 독립(independent) 분포, (T_2) 상관 (correlated) 분포, (T_3) 반상관 (anti-correlated) 분포를 이용하였다. 실험에 사용한 다차원 데이터 집합 A는 28M개의 2차원 데이터로 주기억장치에 생성하였고, 원점의 이동은 실행시간에 영향이 거의 없으므로 실험에는 포함하지 않았다. 사용자의 요구조건인 각도 조건은 $14^\circ \leq d_1 \leq 44^\circ$, $46^\circ \leq d_2 \leq 76^\circ$ 로 하였으며, 실험은 각 5° 간격으로 실시하였다.

(그림 5)는 독립 분포 데이터의 실험 결과이다. 다양한 각도 범위에 대해, 필터링과 정지선 사용시 원본 데이터와 비교했을 때 처리되는 데이터의 분량 비교를 백분율로 표시하였다. 또한 사용자의 요구조건을 반영하지 않고 필터링만을 했을 때의 실행시간과 사용자의 요구조건과 정지선을 사용했을 때의 실행시간 비율을 나타내었다. 또한 각도 제한을 주었을 때 구해지는 스카이라인 원소 개수를 전체 스카이라인의 원소 개수와도 비율로 표시하였다. 예를 들어, $34^\circ \sim 56^\circ$ 의 경우, 필터링 후 데이터는 8.46%로 감소하였고, 정지선 사용 후 2.76%로 감소하였으며, 실행 시간은 12%로 감소하였으나 스카이라인 원소는 39%를 구할 수 있었다. 각도 제한이 커도 스카이라인 원소비율은 전체적으로 크게 감소하지 않는다는 것을 알 수 있다.



(그림 5) 독립분포 데이터 (T_1)의 스카이라인 계산

(그림 6)은 상관 분포 데이터 집합 (T_2)과 반 상관 분포 데이터 집합 (T_3)에 대한 실험 결과이다. (그림 6)(a)는 T_2 의 경우로, 특히 필터링과 정지선 사용의 효과가 크다는 것을 알 수 있다. 차원 상관 관계가 높아 모든 데이터가 $29^\circ \sim 61^\circ$ 에 있었고, 이 경우를 100% (실제 1.93초)로 하여 실행시간 비율을 계산하였다. (그림 6)(b)는 T_3 의 경우의 실험 결과이다. T_2 에 비해 필터링 효과가 상대적으로 작고, 정지선의 활용은 거의 도움이 되지 않는다. $29^\circ \sim 61^\circ$ 의 경우 실행시간은 71초, $34^\circ \sim 56^\circ$ 의 경우 29초가 걸렸다. T_3 의 경우는 각도를 크게 줄여야 실행시간을 감소시킬 수 있다. 40% 정도의 스카이라인을 구하려면 $39^\circ \sim 51^\circ$ 정도 되어야 하며, 필터링과 정지선으로 4% 정도의 데이터를 처리하면 되었다. 본 연구의 실험은 2차원 데이터로 하였으나, 알고리즘은 다차원 데이터에도 적용 가능하다.



(a) T_2 경우 (b) T_3 경우

(그림 6) 스카이라인 계산

5. 결 론

본 연구에서는 사용자가 원하는 각도와 위치 이동을 고려하여 스카이라인의 일부를 신속하게 구하는 방법을 제시하였다. 필터링과 정지선의 사용이 독립 분포 데이터와 상관 분포 데이터에 대해 매우 효과적이라는 것을 실험을 통해 보였고, 반상관 분포 데이터의 경우는 각도를 줄임으로써, 실행시간을 줄일 수 있다는 것을 보였다. 또한 각도를 줄여도 구하고자 하는 스카이라인의 원소 개수는 크게 감소되지 않는다는 것도 알 수 있었다.

참 고 문 헌

[1] I. Bartolini, P. Ciaccia, M. Patella, "SaLSa: Computing the Skyline without Scanning the Whole Sky," *CIKM 2006*, pp.405~414, USA, Nov., 2006.

[2] S. Borzsonyi, D. Kossmann, and K. Stocker. "The Skyline operator," *ICDE 2001*, pp.421~430, Germany, Apr., 2001.

[3] J. Chomicki, P. Godfrey, J. Gryz, D. Liang, "Skyline with Presorting," *ICDE 2003*, pp.717~719, India, 2003.

[4] E. Dellis, B. Seeger, "Efficient Computation of Reverse Skyline Queries," *VLDB 2007*, pp.291~302, Austria, Sep., 2007.

[5] P. Godfrey, R. Shipley, and J. Gryz. Maximal vector computation in large data sets. *VLDB 2005*, pp.229~240, Norway, Aug., 2005.

[6] D. Papadias, Y. Tao, G. Fu, B. Seeger, "Progressive Skyline Computation in Database Systems," *ACM Transactions on Database Systems 30(1)*, pp.41~82, 2005.

[7] F. P. Preparata, M. I. Shamos, "*Computational Geometry - An Introduction*", Springer, 1985.

[8] A. Siddique, Y. Morimoto, "K-Dominant Skyline Computation by Using Sort-Filtering Method," *PAKDD 2009*, pp.839~848, 2009.

[9] K.-L. Tan, P.-K. Eng, B. C. Ooi, "Effient Progressive Skyline Computation," *VLDB 2001*, pp.301~310, Italy, 2001.

[10] T. Xia, D. Zhang, Y. Tao, "On Skylining with Flexible Dominance Relation," *ICDE 2008*, pp.1397~1399, Mexico, 2008.

김 지 현

e-mail : jhrosa@ewhain.net

1995년 상명대학교 정보과학과(학사)

2000년~2004년 (주)한진정보통신주식회사 근무

2007년 이화여자대학교 컴퓨터학과 (공학석사)

2011년 이화여자대학교 컴퓨터공학과 박사과정

관심분야 : 이동객체 데이터베이스, 스카이라인, 데이터분석, 스트림 데이터, 온톨로지, 정보검색

김 명

e-mail : mkim@ewha.ac.kr

1981년 이화여자대학교 수학과(학사)

1983년 서울대학교 계산통계학과(이학석사)

1993년 University of California, Santa Barbara, USA(공학박사)

1993년~1995년 University of California, Santa Barbara, 박사후 과정, 강사

1995년~현 재 이화여자대학교 컴퓨터공학과 교수

관심분야 : 대용량 데이터분석, 실시간 추천시스템, 정보검색, 고성능컴퓨팅 등