# 사용자 요구 조건을 반영한 효율적인 스카이라인 계산 알고리즘

김지현, 김명
이화여자대학교 컴퓨터공학과
e-mail: jhrosa@ewhain.net, mkim@ewha.ac.kr

# An Efficient Algorithm for Computing Skylines Considering Users' Preferences

Jihyun Kim, Myung Kim
Dept of Computer Science and Engineering, Ewha Womans University

요 약

다차원 점들로 구성된 점집합이 주어졌을 때, 그 집합에 속한 다른 점들에 지배되지 않는 점들로 구성된 부분집합을 스카이라인이라고 한다. 방대한 양의 다차원 정보를 다차원 점집합으로 보았을 때 그 집합의 스카이라인은 사용자가 의사 결정을 할 때 유용한 정보일 수 있으므로, 스카이라인을 신속하게 계산하려는 데 많은 관심이 모아지고 있다. 최근에는 방대한 크기의 점집합에 대해 스카이라인을 신속하게 계산하는 알고리즘들이 많이 개발되었다. 그러나 점집합이나 그 점집합의 스카이라인이 매우 큰 경우에 스카이라인 전체를 계산하는 것은 실제 사용자에게 큰 도움이 되지 않을 수가 있다. 이 논문에서는 스카이라인을 계산하기 전에 사용자가 자신의 선호도를 거리나 개수로 제시하는 경우, 이를 반영하여 사용자의 선호도를 가장 잘 만족하는 스카이라인 일부분을 구하는 방법을 제안한다.

## 1. 서론

$d$ 차원의 점들로 구성된 점집합 $S$가 있다고 하자. 그리고 $p=(p_1,p_2,\ldots,p_d)$와 $q=(q_1,q_2,\ldots,q_d)$는 $S$에 속한 두 점이라고 하자. $p_i \le q_i$, $1 \le i \le d$를 만족하고 적어도 하나의 $i$에 대해 $p_i < q_i$를 만족한다면 $p$는 $q$를 지배(dominate)한다고 한다[3]. $S$에 속한 점들 중에서 집합의 어떤 점에도 지배되지 않는 점들로 구성된 부분집합을 점집합 $S$의 스카이라인(skyline)이라고 한다[3].

스카이라인의 예를 실생활에서 들어보자. 예를 들어, 경치 좋은 바닷가에 여러 집들이 있다고 하자. 이 들 중에서 바닷가에 가까우면서 가격이 저렴한 집들의 집합은 스카이라인에 해당하며, 이러한 스카이라인 정보는 사용자가 집구매를 위해 최종적으로 원하는 집을 선택할 때 많은 도움을 줄 수 있다. 따라서 최근에는 스카이라인 계산에 많은 관심이 모이고 있다.

스카이라인 계산에 대한 최근 연구들은 점집합과 그 점집합의 스카이라인이 방대하여 계산 도중에 주기억장치가 모자라는 경우의 해결책을 강구하는 경우가 많다. 데이터 정렬시에 외부 정렬을 사용하거나, 데이터 처리 과정에서 스카이라인에 해당하지 않는 점들을 가능하면 빨리 제거하는데 많은 노력을 기울여 왔으며 효율적인 알고리즘들이 많이 개발되어 있다 [2, 3, 4].

스카이라인을 실생활에 적용하여 사용자가 최종 결정을 하는데 도움을 주기 위해서는 스카이라인 전체를 구하는 것보다 사용자가 가장 선호하는 스카이라인의 작은 부분집합을 구해 주는 것도 매우 유용하다고 볼 수 있다. 예를 들어, 바닷가에 가까운 집 수백 채에 대한 정보를 주는 것보다는 바닷가에 가까우면서 특정 지점으로부터 일정 거리 안에 드는 집들에 대한 정보를 받거나, 특정 지점으로부터 가장 가까운 집들에 대한 정보 십여 개만 받는 것이 사용자에게 더 유익할 수 있다.

본 연구에서는 사용자가 제시하는 거리와 개수 정보를 고려하여 이를 가장 잘 만족하는 스카이라인의 부분집합을 구하는 효율적인 알고리즘들을 제안한다. 2절에서는 기존의 연구에 대해 살펴보고, 3절에서 알고리즘들을 제시하며, 4절에서 분석과 실험결과를 제시하고, 5절에서 결론을 맺는다.

## 2. 기존 연구 분석

주어진 다차원 점집합 (또는 데이터) 에서 스카이라인을 효율적으로 찾는 알고리즘이 [1]에 소개되어 있으나 이 알고리즘은 점집합과 스카이라인이 모두 주기억장치에 저장 가능하고, 스카이라인 계산 역시 주기억장치에서 실행 가능하다는 가정을 한다. 그러나 실생활에서 스카이라인을 계산할 데이터가 주기억장치에 저장될 수 없을 정도로 클 수 있기 때문에 데이터를 효율적으로 외부 정렬하거나, 정렬과정에서 스카이라인이 될 수 없는 데이터가 발견될 때마다 제거해 가는 기법들이 개발되어 있다[3, 4]. 서버/클라이언트 환경을 가정하고 스카이라인을 클라이언트 측에

서 계산할 때 서버로부터의 데이터 전송을 최소화하려는 기법도 개발되어 있다[2]. 비트맵과 인덱스를 사용하여 스카이라인에 속한 데이터가 하나씩 구해질 때마다 출력해 주는 기법[5]과 가장 많은 수의 데이터를 지배하는 스카이라인 점들을 출력해 주는 알고리즘[6]도 개발되어 있다. 그러나 이 기법들은 스카이라인에 포함되는 데이터의 개수에 무관하게 스카이라인을 구하고 그 분량이 방대할 수도 있기 때문에 필요 이상으로 시간 낭비가 될 수도 있다.

## 3. 사용자 선호도를 고려한 스카이라인 계산 알고리즘

본 연구는 스카이라인을 계산하기 전에 사용자가 원하는 요구조건을 반영하여 스카이라인의 부분집합을 신속하게 구하는 것에 초점을 맞춘다. 사용자는 다음과 같이 자신의 요구조건을 제시할 수 있다. (조건 1) 원점에서 가장 가까운 $n$개의 스카이라인을 구한다. (조건 2) 원점 기준으로 거리가 $x$ 이내인 스카이라인을 구한다. (조건 3) 원점에서 가장 가까운 스카이라인을 구하고, 그 점으로부터 거리가 $x$이내인 스카이라인을 구한다. (조건 1)을 만족하는 스카이라인은 [알고리즘 1]로 구할 수 있다.

**[알고리즘 1]**

[단계 1] 입력 점집합 $A$로부터 샘플 점들을 골라내고, 샘플 중에서 스카이라인을 구한 후, 그 중에서 원점에 가장 가까운 스카이라인 점 $p$를 찾는다.

[단계 2] 입력 점집합 $A$를 스캔하면서 $p$에 지배되는 모든 점들을 제거하고 남은 점들로 구성된 점집합 $B$를 생성한다.

[단계 3] 집합 $B$의 점들에 대해 원점으로부터의 거리를 계산하고, 거리의 오름차순으로 점들을 정렬한다.

[단계 4] 집합 $B$의 점들을 순차적으로 스캔해 가면서 스카이라인을 구하여 윈도우 $SKY$에 넣어간다. $SKY$에 속한 점들의 개수가 $n$이 되면 계산을 멈춘다.

[단계 1]과 [단계 2]는 LESS 알고리즘의 아이디어[4]를 확장한 것이고, [단계 4]의 경우 집합 $B$의 점 $p$가 스카이라인 점인지 알기 위해서는 이미 구하여 $SKY$에 저장해 놓은 스카이라인 점들과만 비교하면 된다. 이 알고리즘은 [단계 2]에서 많은 점들을 제거하고, [단계 4]에서 필요한 만큼의 스카이라인을 구하는 효율적인 알고리즘이다.

(조건 2)를 고려한 알고리즘 **[알고리즘 2]**는 [알고리즘 1]과 유사하나 [단계 2]에서 원점으로부터의 거리가 $x$를 넘는 점들을 제거한다는 것과 [단계 4]에서 개수 제한 없이 스카이라인 점들을 모두 계산한다는 점이 다르다.

(조건 3)을 고려한 알고리즘 **[알고리즘 3]**은 [알고리즘 2]와 유사하나 [단계 2]에서 점 $p$를 기준으로 거리가 $x$가 넘는 점들을 추가적으로 제거한다는 것과 [단계 4]에서 가장 처음으로 구한 스카이라인 점으로부터 거리가 $x$를 넘는 스카이라인 점들을 제거한다는 점이 다르다.

## 4. 알고리즘 분석과 성능평가 결과

[알고리즘 1]은 [단계 2]의 $B$와 [단계 4]의 $n$의 크기를 줄여 시간을 감소시켰다. [알고리즘 2]는 [단계 2]에서 더욱 많은 점들을 제거한다는 점에서 효율적이나, 사용자가 원점으로부터 거리 $x$안에 들어올 스카이라인 점들이 몇 개 정도인지 잘 알 수 없다는 단점이 있다. [알고리즘 3]은 사용자가 가장 선호할 스카이라인 점으로부터 거리가 $x$이내인 점들을 구하는 것이므로 효율적인 방법이라고 볼 수 있다.

제안하는 알고리즘들을 인텔 코어 i5 CPU, 2.4 GHz 컴퓨터에서 C언어로 작성하여 성능평가를 하였다. 50만개의 서로 연관성 높은 점들로 $A$를 임의 작성하였다. $B$를 구하지 않는 경우는 스카이라인 점 237개를 모두 구하는 경우 0.32초가 걸렸고, $n$을 20으로 제한하는 경우 0.29초가 걸렸다. [알고리즘 1]에서 $B$는 $A$의 약 10% 정도인 55,140개가 되었고, $n$을 20으로 하는 경우 0.052초 걸려 실행 시간이 1/6 정도로 감소하였다. [알고리즘 2]와 [알고리즘 3]에서 $n$이 대략 20개가 되도록 거리를 제한하였더니 [알고리즘 2]는0.025초 걸렸고, [알고리즘 3]은 0.028초 걸렸다. 알고리즘 모두 [단계 3]에서 외부 정렬이 가능하여 $A$가 커도 실행 가능하다.

## 5. 결론

본 연구에서는 사용자가 특정 위치로부터의 거리나 개수로 표현한 선호도를 고려하여 스카이라인의 일부를 계산하는 알고리즘을 제안하였다. 이는 스카이라인 점들 중에서도 사용자가 가장 필요로 하는 소수의 점들만을 구함으로써 시간 효율성을 높인 연구이다.

## 참고문헌

[1] H. T. Kung, F. Luccio, and F. P. Preparata. “On finding the maxima of a set of vectors,” *Journal of the ACM*, Vol. 22, No. 4, pp. 469~476, 1975.

[2] Ilaria Bartolini, Paolo Ciaccia, Marco Patella, “SaLSa: Computing the Skyline without Scanning the Whole Sky,” CIKM’06, November 5~11, 2006, Arlington, Virginia, USA.

[3] Stephan Borzsonyi, Donald Kossmann, and Konrad Stocker, ″The Skyline Operator,″ in ICDE 2001, pp. 421~430, Heidelberg, Germany, Aprl. 2001.

[4] P. Godfrey, R. Shipley, and J. Gryz, “Maximal vector computation in large data sets,” In VLDB 2005, pp. 229~240, Trondheim, Norway, Aug. 2005.

[5] Kian-Lee Tan, Pin-Kwang Eng, and Beng Chin Ooi, “EÆcient Progressive Skyline Computation,” Proceedings of the 27th VLDB Conference, Roma, Italy, 2001.

[6] MD. Anisuzzaman Siddique, Yasuhiko Morimoto, ″Continous k=Dominant Skyline Computation by Using Divide and Conquer Strategy,″ IPSJ Online Transaction, Vol 3, 97~109, June 2010.