영역객체를 위한 리버스 스카이라인 질의 처리
(Reverse Skyline Query Processing for Region Objects)

한 아 † 이 종 혁 † 박 영 배 ††
(Han Ah) (ZhongHe-Li) (Youngbae Park)

요 약 이전의 리버스 스카이라인 질의 처리 기법들은 고정된 조건 값을 가지는 점객체 환경만을 고려하기 때문에, "가격이 5만원~7만원이고, 해변까지의 거리가 1km~2km인 호텔"와 같이 조건이 범위로 주어지는 영역객체 환경에서의 질의처리에는 부적합하다. 이러한 한계를 극복하기 위하여, 본 논문에서는 점객체 뿐만 아니라 영역객체 또한 지원 가능한 리버스 스카이라인 질의 처리 기법을 제안한다. 이는 첫째, 점객체 환경에서의 리버스 스카이라인 질의처리 기법 중 효율이 좋은 ERSL기법을 확장한 기법으로써 높은 성능을 기대할 수 있다. 또한 둘째, 영역객체와 제안하는 가지치기 기법과의 겹침 관계에 따라 결과객체의 중요도를 다르게 하여 질의자가 결과객체를 차별적으로 사용할 수 있도록 선택권을 제공한다는 새로운 특징이 있다. 본 기법은 영역객체를 지원하는 최초의 기법으로써 성능을 비교할 다른 대상기법이 없다. 그러므로 제안하는 기법의 성능 및 질의결과에 영향을 주는 조건이 무엇이며, 그에 따라 소모되는 실행시간을 측정하여 본 기법의 효율성을 증명하였다.

키워드 : 스카이라인 질의, 리버스 스카이라인 질의, 영역객체, 영역질의, RSSA, BBRS

Abstract Existing methods to compute reverse skyline queries are not correct to process the queries in dataset with region objects which have conditions like a price is 5~7 dollars and a distance to beach is 1km~2km, since they consider datasets with only point objects. To solve the problem, we propose a novel method to process reverse skyline queries for region objects in this paper. It has advantages. First, it is expected to get a good performance, because it is extended from efficient reverse skyline (ERSL) algorithm which is a best algorithm to computing reverse skyline queries in datasets with point objects. Second, it can give a right of choice unlike the others to a person requesting the query. That is because results of reverse skyline have a difference preference according to proposed pruning methods and overlap relations. This algorithm is a first for supporting region objects. Therefore there are not any other algorithms to compare their performance. For that reason, our experiment to prove the efficiency of proposed algorithm is focused what conditions give an effect to its performance and result and how much time it needs to process the query.

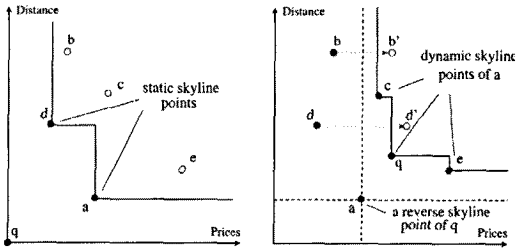
Key words : skyline query, reverse skyline query, region object, region query, RSSA, BBRS

1. 서 론

스카이라인 기법은 정보 요청자의 선호도를 바탕으로 그들이 흥미로워할 정보들을 제공해주는 방법으로써, 정보를 원하는 사람이 질의의 주체가 되어 자신의 사용목적에 맞는 정보를 검색한다. 예를 들어 여행을 가서 "바다에서 가깝고, 숙박료가 저렴한 호텔"을 찾는다고 가정하자. 이때 일반적으로 바다에서 가까울수록 숙박료가 비싸고, 질의자가 두 가지 조건 중 어느 조건을 더 중요하게 생각하는지 알지 못하기 때문에 질의처리기는 최적의 호텔을 하나만 찾아줄 수 없다. 그러나 이러한 경우 스카이라인 질의를 이용하면 여러 개의 호텔 중 질의자가 흥미로워할 것 같은 호텔들을 권해주어 그들이 자신에게 맞는 호텔을 선택할 수 있도록 도울 수 있

† 학생회원 : 명지대학교 컴퓨터공학과
pinkey83@nate.com
mslzh@mju.ac.kr
†† 종신회원 : 명지대학교 컴퓨터공학과 교수
parkyb@mju.ac.kr
논문접수 : 2010년 5월 24일
심사완료 : 2010년 6월 29일

정보과학회논문지: 데이타베이스 제37권 제4호(2010.8)



(a) Static Skyline (b) Dynamic/Reverse Skyline

그림 1 스카이라인과 리버스 스카이라인

다[1-5]. 그림 1(a)은 스카이라인 결과객체로써 가격, 해변까지의 거리의 조건을 가지는 5개의 호텔 중 질의자가 흥미로워할 것 같은 호텔 a, d를 보여준다.

리버스 스카이라인 기법은 스카이라인 기법의 특징을 역으로 이용한 기법으로써, 정보를 제공하는 대상이 질의의 주체가 되어 제공할 정보를 흥미롭게 생각할 것 같은 정보 사용자들을 검색한다[6,7]. 호텔 예의 경우, 호텔주인은 자신의 호텔을 홍보하기 위해 불특정 다수, 혹은 보유하고 있는 모든 고객에게 정보를 제공할 수 밖에 없다. 그러나 이러한 방법은 상관없는 고객에게도 정보가 제공되기 때문에 비용대비 높은 효율을 기대하기 어렵다. 이때 리버스 스카이라인 질의를 사용하면, 정보제공자(호텔주인)는 자신의 호텔을 스카이라인으로써 흥미로워 할 고객을 검색하여 그들에게만 정보를 제공할 수 있다[7]. 정보사용여부에 대한 최종적인 결정은 정보를 제공받은 정보사용자가 선택하겠지만, 정보제공자는 적은 비용으로 높은 효율을 기대할 수 있으며, 정보사용자 또한 특별한 노력 없이 비교적 필요한 정보만을 제공받을 수 있다.

리버스 스카이라인을 구하는 가장 원시적인 방법은 모든 객체들의 스카이라인을 검사하여 질의점이 해당 스카이라인에 포함되는지의 여부를 확인하는 것이다. 즉, 그림 1(b)의 경우 객체 a가 리버스 스카이라인인지 판별하기 위해서는 a의 스카이라인을 구해야 한다. 그 결과 객체 c, e와 함께 q도 a의 스카이라인에 포함되었으므로 a는 q에 대한 리버스 스카이라인이 된다. 동일한 방법으로 b~e 객체의 스카이라인을 모두 확인하면, c와 e객체 또한 질의점을 그들의 스카이라인에 포함하기 때문에 최종적으로 질의점 q에 대한 리버스 스카이라인 결과는 그림 2와 같이 a, c, e가 된다. 이러한 방법은 처리절차가 간단하지만 데이터셋(dataset)의 크기가 클수록, 고차원 일수록 실행시간이 오래 걸리기 때문에 비효율적이다.

실행시간의 낭비를 줄이고 보다 효율적으로 리버스 스카이라인 질의를 처리하기 위해 다양한 기법들이 제안되었다. [8]의 RSSA(Reverse Skyline using Skyline Approximations)기법은 전처리 과정을 통해 미리 계산된 정보를 기반으로 윈도우 질의를 이용하여 결과를 도출하고, [7]의 ERSL(Efficient Reverse Skyline)기법은 전처리 과정없이 가지치기 기법과 분기한정법(branch-and-bound)을 이용하여 결과를 도출한다. 특히 RSSA 기법은 다양한 분야에서 그들만의 고유한 환경에 맞는 새로운 리버스 스카이라인 처리 기법을 제안하는데 기초가 되어, [9]에서는 이를 토대로 센서데이터 환경에서의 리버스 스카이라인 질의 처리를 지원하는 MPRS (Monochromatic Probabilistic Reverse Skyline)기법을 제안하였다.

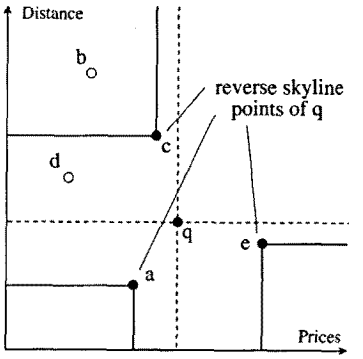


그림 2 리버스 스카이라인

이전까지의 리버스 스카이라인 기법들은 대상객체의 조건에 제한이 있어 성능에 한계가 있다. RSSA기법과 ERSL기법은 점객체 즉, 객체의 모든 조건이 고정된 하나의 값을 가지는 환경에서의 리버스 스카이라인 질의처리만을 지원한다. 또한 MPRS기법은 일정한 범위 안에서 수시로 변하는 즉, 어떤 센서에서 수집된 정보의 군집을 불확실한 영역객체(uncertain object)로 정의하고, 그들을 포함하는 환경에서의 질의처리를 지원한다. 이는 센서데이터 환경에 특화된 개념으로써 과거 변경된 정보를 토대로 영역 안의 일부 조건을 무시하여 객체를 임의대로 변경하기 때문에 이외의 분야에 적용하기에 적합하지 않다.

본 논문에서는 점객체 뿐만 아니라 영역객체(region object)를 가지는 데이터환경도 지원 가능한 새로운 리버스 스카이라인 기법을 제안한다. 여기서 영역개체란 일반적인 데이터베이스 환경에서 하나의 정보가 범위로 주어진 것을 말한다. 이는 전체 영역이 하나의 객체이기 때문에 부분적으로 조건을 무시하거나 제외할 수 없어 센서데이터 환경에서의 불확실한 영역객체와 다르다.

제안하는 기법은 ERSL기법을 영역객체도 지원 가능하도록 확장한 기법으로써 ERSL기법의 높은 성능을 유

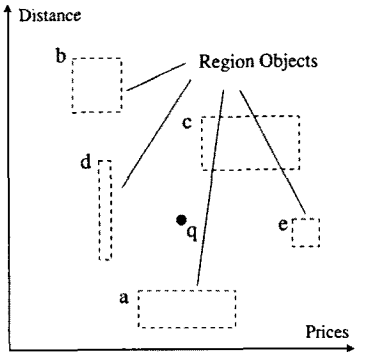


그림 3 영역객체 데이터베이스

지할 것을 기대할 수 있다. 또한 영역객체와 제안하는 가지치기 기법의 겹침 관계에 따라 각 객체의 선호도를 재 계산하여 결과객체를 차별화함으로써, 서로 다른 선호도 값을 가지는 결과객체들을 질의자가 차별적으로 사용할 수 있도록 선택권을 제공한다는 점에서 의미가 있다. 이러한 제안된 기법을 RSRD(Reverse Skyline in Region Database) 라고 명명한다.

RSRD기법은 영역객체의 리버스 스카이라인 질의 처리를 위한 최초의 기법이기 때문에 성능을 비교할 다른 대상기법이 존재하지 않는다. 그러므로 RSRD기법의 성능 및 질의결과에 영향을 주는 조건이 무엇이며, 그에 따라 소모되는 실행시간을 측정하여 본 기법의 효율성을 증명하였다. 또한 기존의 점객체 환경에서의 리버스 스카이라인 질의처리 기법인 RSSA, ERSL기법과의 실행시간을 비교하여 본 기법의 성능을 보였다.

이후, 2장에서는 리버스 스카이라인 질의의 개념과 질의를 처리하기 위한 RSSA, ERSL, MPRS기법을 소개한다. 3장에서는 RSRD기법을 위한 가지치기 기법들과 선호도 재 정의기법을 제안하고, 전체적인 RSRD의 처리과정을 설명한다. 4장에서는 제안하는 기법의 성능을 증명하기 위하여 실험한 결과를 비교 분석하여, 마지막으로 5장에서 결론을 도출한다.

2. 리버스 스카이라인

리버스 스카이라인(Reverse Skyline)은 질의점 q를 자신의 스카이라인 안에 포함하는 객체들의 집합이다[6-9]. 즉, 객체 p_1, $p_2 \in P$이고 질의점 $q \in P$일 때, 모든 차원에서 $|p_2-p_1| \leq |q-p_1|$이거나, 적어도 한 차원에서 $|p_2-p_1| < |q-p_1|$을 만족하는 객체 p_1은 리버스 스카이라인 객체이다[8,10]. 이는 질의점의 정보에 관심을 가질만한 객체들을 검색함으로써 정보 스스로 자신을 사용할 가능성이 있는 고객을 찾는 것과 같다.

예를 들어, "가격이 5만원이고, 해변까지의 거리가 500m"인 호텔을 소유하고 있다고 가정하자. 이 호텔을 홍보하려면 이전에는 많은 비용을 들여 불특정 다수, 혹은 알고 있는 모든 고객에게 정보를 제공하는 방법밖에 없었다. 그러나 리버스 스카이라인 질의를 이용하면 고객의 선호도정보를 바탕으로 자신의 호텔에 관심이 있을 것 같은 고객들을 검색할 수 있다. 그 결과, 호텔은 방문 가능성이 있는 고객에게만 홍보하여 저비용으로 높은 효율을 얻을 수 있으며, 고객은 자신의 선호에 근접한 정보를 얻어 선택의 폭을 넓힐 수 있다.

리버스 스카이라인 질의를 효율적으로 계산하기 위하여 VLDB(07)에서 발표된 RSSA기법을 시작으로 ERSL기법 등이 제안되었다. 또한 여러 분야에서 자신의 환경에 맞는 다양한 리버스 스카이라인 질의 처리 기법들이 연구되고 있다.

2.1 RSSA 기법

RSSA기법은 전처리 과정을 통해 모든 객체들에 대한 다이나믹 스카이라인(Dynamic Skyline)[3,4]을 디스크에 유지하고, 질의 요청 시 후보객체의 스카이라인을 확인하여 질의점이 그것에 지배되는지의 여부로 리버스 스카이라인 객체를 판별할 수 있다고 설명한다. 그러나 만일 질의점과 스카이라인간의 지배관계가 비교 불가능한 경우, 실행시간 낭비의 원인이 되는 윈도우 질의(Window Query)를 통해서만 최종적으로 결과객체를 도출할 수 있다[8].

RSSA기법은 리버스 스카이라인 질의를 처리하는 최초의 기법으로써 다양한 분야에서 응용·확장되고 있다. 그러나, 메모리의 낭비가 심하고, 객체의 추가·삭제로 인해 영향을 받는 저장된 스카이라인들을 수정·관리해야 하므로 객체의 변화에 유연하지 못한 단점이 있다.

2.2 ERSL 기법

ERSL기법은 스카이라인 질의처리 기법[1-3] 중 효율이 좋은 BBS(Branch and Bound Skyline) 기법[3]을 바탕으로 리버스 스카이라인 질의 처리에 맞게 확장한 기법으로써 RSSA기법보다 메모리의 낭비를 절감할 뿐만 아니라, 객체의 추가 삭제로 인한 변화에 유연하다. 또한 윈도우 질의를 수행할 필요가 없어 실행시간 또한 절감되었다[6,7]. ERSL기법의 처리과정을 간단히 나열하면 아래와 같다.

가) 첫 번째 후보객체로써 질의점 기준 가장 가까운 객체인 1-NN을 구한다.

나) 선택된 후보객체를 이용하여 다음 후보객체가 위치할 대상영역을 계산하고, 그 영역에서 다음 후보객체를 선택한다.

다) 후보객체와 질의점과의 거리의 2배 영역 안에 다른 객체가 존재하는지 판별하여 비어있으면 최종 결과객체가 된다.

2.3 MPRS 기법

MPRS기법은 여러 곳에 분포되어 있는 센서들에서 실시간으로 수집된 데이터를 대상으로 리버스 스카이라인 질의를 처리하기 위한 기법이다[9]. 이때 수집된 데이터들은 일정한 범위 내에서 수시로 변하기 때문에 하나의 센서에서 수집된 정보들의 군집을 불확실한 영역(Uncertain Region: UR)으로 정의하고, 영역 안에서의 객체변화를 확률분포로 계산하여 가지치기(Probabilistic Pruning Method)에 이용하였다.

MPRS기법은 영역객체의 특징에 부합하는 리버스 스카이라인 질의처리를 지원한다는 새로운 접근으로써 의미가 있다. 그러나, 유동적으로 변화하는 데이터에 대한 확률분포 값을 계산하고 유지·관리해야 하기 때문에 처리과정이 복잡하고, 질의 시점의 확률분포에 따라 리버스 스카이라인 결과값이 달라지는 불확실성의 문제가 있다. 이러한 센서데이터 환경에 특화된 고유한 특성 때문에 다양한 분야에 적용하기에 적합하지 않다.

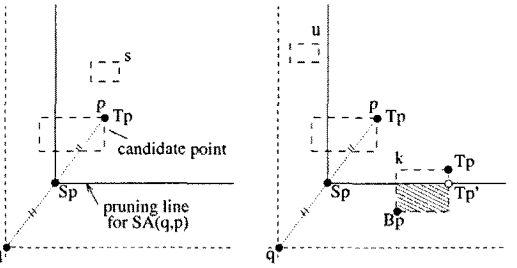
3. 영역객체를 위한 리버스 스카이라인

본 논문에서는 영역데이터 환경에서의 리버스 스카이라인 질의처리를 위한 RSRD(Reverse Skyline in Region Database) 기법을 제안한다. RSRD 기법은 점객체 환경에서 좋은 성능으로 리버스 스카이라인 질의 처리를 지원하는 ERSL기법을 영역객체 환경에서의 질의처리도 가능하도록 확장한 기법이다. 그렇기 때문에 ERSL기법과 동일한 방법으로 R*-tree로 색인된 데이터셋에 대하여 질의점과의 거리를 기준으로 정렬되는 우선순위 큐(Heap)를 이용하여 결과를 도출한다. 이는 리버스 스카이라인 결과객체가 포함되어 있는 디스크 페이지만 접근하기 때문에 낭비가 적어 좋은 성능을 기대할 수 있다.

3.1 리버스 스카이라인 후보객체 선정

효율적인 질의처리를 위해서는 리버스 스카이라인으로 가능성 있는 최소한의 후보객체를 선정하는 것이 중요하다. 이를 위해 SA(Search Area)영역과 SA를 이용한 가지치기 기법을 제안한다. 이것은 기존에 선택된 후보객체의 영향으로 리버스 스카이라인이 되지 못하는 객체를 미리 고려대상에서 제외하기 위함이다. 질의점에서 가장 가까운 거리에 있는 1-NN(First Nearest Neighbor)객체를 첫 번째 후보객체로 선정한 후 이전까지 선택된 후보객체들의 SA을 이용하여 다음 후보객체를 선정한다면 이전의 리버스 스카이라인 기법들보다 절반의 후보객체를 설정할 수 있다. 후보객체 p의 SA영역은 아래와 같이 정의된다.

정의 1. Search Area (SA): 질의점이 q이고 모든 차원 $i = \{1,\cdots,d\}$에서 후보객체 $p = (p_1^-,p_1^+;\cdots;p_d^-,p_d^+)$일 때 SA(q,p)는 다음과 같다. (1) 만일 $q_i < p_i^-$ 이면, {SA(q,p) | $[q_i, (p_i^+ + q_i)/2]$}이고, (2) $q_i > p_i^+$이면, {SA(q,p) | $[(p_i^- + q_i)/2, q_i]$}이다.



(a) 객체 p의 SA (b) SA(q,p)와 객체 k

그림 4 후보객체 p의 SA와 다음 후보객체와의 관계

그림 4(a)는 2차원 데이터환경에서 후보객체 p를 기준으로 정의된 SA(q,p)을 보여준다. 영역으로 주어진 객체 p는 오른쪽 상단의 위치(Tp: Top point)와 왼쪽 하단의 위치(Bp: Bottom point)로 저장된다. 이때 SA는 질의점과 p의 Tp과의 거리 이등분점인 Sp(Search point)을 기준으로, Sp에 의해 지배되지 않는 영역(그림에서 음영된 영역)을 말하며, p의 SA영역 안에 존재하는 객체 만이 다음 후보객체로써 가능성이 있다. 즉, 그림 4(a)의 객체 s와 같이 SA영역 밖에 온전히 존재하는 객체는 후보객체에서 제외된다. 왜냐하면 그들의 스카이라인에는 질의점이 아닌, 질의점을 지배하는 객체 p가 스카이라인으로 포함되기 때문이다.

정리 1. SA을 이용한 가지치기 기법

후보객체 p와 질의점 q가 주어질 때, 객체 p에 대한 SA(q,p)영역에 포함되지도, 겹치지도 않는 객체들은 리버스 스카이라인에서 제외된다. □

정리 1과 같이, 객체의 전체영역 즉 Tp와 Bp 모두 다른 후보객체의 SA영역 밖에 존재하면 그 객체는 확실히 리버스 스카이라인이 될 수 없다. 그러나 그림 4(b)의 객체 k와 같이 Tp가 SA영역 밖에 있지만 Bp가 포함되어 SA영역에 겹쳐(overlap)있는 경우, Tp의 조건에서는 p의 영향을 받아 질의점을 흥미로워하지 않을 수도 있으나, Bp의 조건에서는 흥미로워할 가능성이 있기 때문에 이를 리버스 스카이라인에서 제외할 수 없다. 즉, 이러한 객체들은 영역 안의 어떠한 조건 값도 무시할 수 없으므로 후보객체로 선정한다. 단, 전체 영역이 온전히 SA(q,p)에 포함되는 객체(u)보다 정보에 대한 만족도가 떨어지므로 그에 맞게 선호도를 재 계산하여 객체의 중요도에 차별을 준다. 겹침 관계에 따른 선호도 재 측정 방법 및 변경된 k의 영역 결정방법은 3.3절에서 자세히 설명하도록 한다.

3.2 결과객체 판별

이전까지 최소한의 리버스 스카이라인 후보객체를 선

정하기 위한 SA영역을 이용한 가지치기 방법을 설명하였다. 이를 통해 선정된 후보객체는 CA(Check Area) 영역과 CA영역을 이용한 가지치기 기법에 의해 최종적으로 리버스 스카이라인인지 아닌지 결정된다. 후보객체 p의 CA영역은 아래와 같이 정의된다.

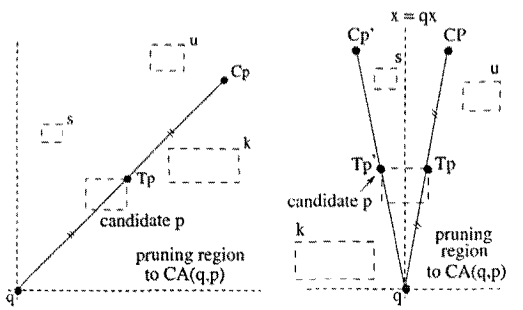
정의 2. Check Area (CA): 질의점이 q이고 모든 차원 $i = \{1,\cdots,d\}$에서 후보객체 $p = (p_1^-,p_1^+;\cdots;p_d^-,p_d^+)$일 때 CA(q,p)는 다음과 같다. (1) 만일 $q_i < p_i^-$이면, $\{CA(q,p) \mid [q_i, 2p_i^+ - q_i)]\}$이고, (2) $q_i > p_i^+$이면, $\{CA(q,p) \mid [2p_i^- - q_i, q_i]\}$이다. 그러나 d차원에 대해서 (3) $p_d^- < q_d < p_d^+$이면, $\{CA(q,p) \mid [-2\ p_d^+ - q_i, 2\ p_d^- - q_i]\}$이다.

그림 5는 후보객체 p를 기준으로 정의된 CA(q,p)을 보여준다. 이는 두 가지 경우에 따라 다르게 생성된다. 첫 번째는 그림 5(a)와 같이 객체 p의 영역 안에서 질의점과 모든 차원에서 일치하는 부분이 없는 경우이고, 두 번째는 그림 5(b)와 같이 질의점과 적어도 하나의 차원에서 일치하는 부분이 있는 경우이다. 전자의 경우 CA(q,p)는 p의 Tp를 기준으로 질의점과의 거리에 두 배를 한 점인 Cp와 질의점이 이루는 영역으로 정의하며, 후자의 경우 질의점의 기준 차원 값(예제에서는 x차원 값)에 따라 CA_1(q,Tp), CA_2(q,Tp')영역으로 나누어 전자와 동일한 방법으로 각각 생성한다. 결과적으로 두 번째 경우의 CA(q,p)는 CA_1, CA_2두 영역의 합(union)으로 정의할 수 있다(그림 5(b)의 음영부분).

후보객체 p는 이렇게 생성된 자신의 CA영역에 다른 객체가 포함되는지에 따라 리버스 스카이라인인지 아닌지 결정된다. 즉, CA영역 안에 어떤 객체가 온전히 포함되면 객체 p는 리버스 스카이라인에서 제외되고, CA이 비어있거나 또는 겹쳐있는 경우 객체 p는 리버스 스카이라인으로 확정된다. 그 이유는 p의 CA에 임의의 객체 s가 포함된다면 질의점이 아닌 질의점을 지배하는 s가 p의 스카이라인에 포함되기 때문이다.

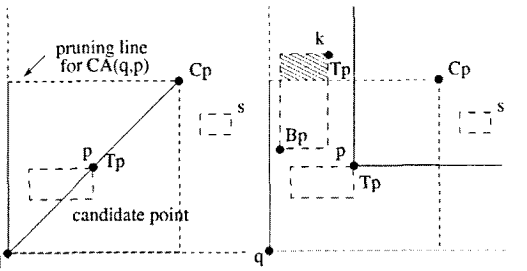
정리 2. CA을 이용한 가지치기 기법

후보객체 p와 질의점 q가 주어질 때, 객체 p에 대한



(a) 첫 번째 경우 (b) 두 번째 경우

그림 5 CA을 정의하는 두 가지 방법



(a) 객체 p의 CA(q,p) (b) CA(q,p)와 객체 k

그림 6 후보객체 p의 CA(q,p)

CA(q,p)영역에 다른 객체의 전체 조건이 모두 포함되면 p는 리버스 스카이라인에서 제외된다. □

SA영역에서와 마찬가지로 CA영역의 포함관계에 있어 그림 6(b)의 객체 k와 같이 CA(q,p)에 겹치는 경우도 고려해야 한다. k의 Bp의 조건에서는 CA(q,p)에 포함되므로 p가 결과객체로 선정되는 것을 방해하지만, Tp의 조건에서는 p에게 아무런 영향도 미치지 않기 때문이다. 이렇게 다른 객체가 자신의 CA영역에 겹치는 경우 포함되는 만큼의 선호도를 절감하여, 중요도를 낮춘 후 리버스 스카이라인에 포함한다. 이때 만일 여러 개의 객체가 CA영역에 겹친다면, 가장 많은 부분이 겹치는 객체 하나만을 기준으로 p의 선호도를 재 계산한다.

3.3 겹침 관계에 따른 선호도 재정의

영역객체는 일정한 범위 안에서 동일한 선호도를 가지는 객체로써, 전체 영역의 어떠한 조건에서도 동일한 중요도를 가진다. 이러한 객체가 어떤 후보객체의 SA영역에 겹쳐지거나 혹은, 자신의 CA영역에 다른 객체가 겹쳐져 부분적으로 리버스 스카이라인의 조건에 만족되면, 만족되는 만큼의 선호도를 유지할 것이라 기대할 수 있다. 그러므로 이들을 리버스 스카이라인 결과객체에 포함시키되 중요도의 차별을 주어 그림 7과 같이 결과객체를 정렬하는데 사용하도록 한다. 이는 기존의 리버스 스카이라인과 달리 질의자가 질의 결과를 선택적으로 사용할 기회를 준다. 예를 들어, 질의자는 자신의 정보를 100%만족하는 고객, 50% 만족하는 고객의 중요성을 다르게 인지하여 서비스를 차별 제공하는 기준으로 사용할 수 있다.

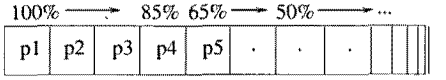
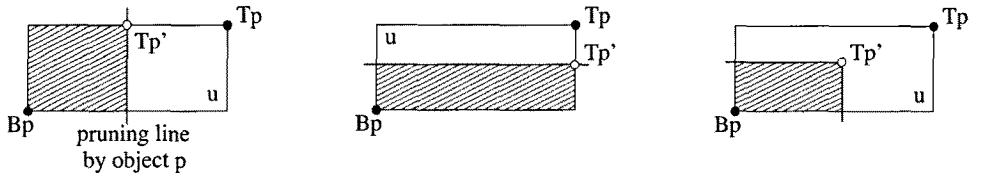


그림 7 선호도에 따라 정렬된 결과 리스트

3.3.1 해당객체가 후보객체의 SA에 겹치는 경우

그림 9와 같이 영역객체 u가 SA(q,p)에 겹칠 때, u는



(a) 첫 번째 경우 (b) 두 번째 경우 (c) 세 번째 경우

그림 8 가지치기 영역에 겹쳐져 영역이 나뉘는 모양

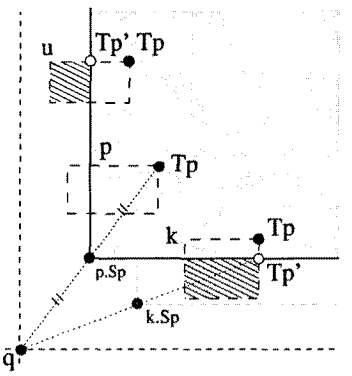


그림 9 객체 u와 SA(q,p)의 겹침 관계

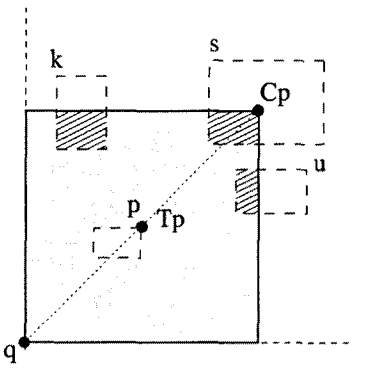


그림 10 객체 u와 CA(q,p)의 겹침 관계

리버스 스카이라인 후보객체로 선정됨과 동시에 전체 면적에서 SA영역에 겹쳐진 면적(빗줄 영역)만큼의 퍼센트로 선호도가 수정된다. 즉, 최초에 u의 선호도가 100%였고 SA에 겹치는 영역이 50%라면, 이후 u의 선호도는 50%(=100 -50)이 된다. 동일한 방법으로 k또한 선호도가 80%로 수정되어 리버스 스카이라인 후보객체는 [p,k,u] 순서로 저장된다. (SA(q,p)에 완전히 포함되는 객체의 선호도는 100%이다.) 이때, SA영역과 객체 간의 겹침 관계는 그림 8의 (a), (b)의 경우만 발생하며, 이후 이들(u,k)의 SA은 SA(q,p) 경계선과 만나는 지점인 Tp'을 기준으로 생성된다.

정리 3. 선호도 재정의 방법1: SA 기준

객체 u가 후보객체 p의 SA영역에 겹치면, u는 후보객체로 선정되며 그 선호도는 전체 영역 중 SA영역에 겹치는 영역만큼으로 재정의된다. □

3.3.2 해당객체의 CA에 다른 객체가 겹치는 경우

그림 10과 같이 영역객체 u가 CA(q,p)에 겹칠 때, 객체 p는 u가 자신의 CA영역에 겹치는 면적만큼의 퍼센트 만큼 선호도가 감소된다. 단, CA(q,p)에 겹치는 객체가 여러 개일 경우 그 중 가장 많은 영역이 겹치는 객체만을 고려한다. 그러므로 그림 10의 경우 p가 최초에 100%의 선호도를 가졌다고 가정 할 때, p는 여전히 리버스 스카이라인 객체로 남지만 가장 많은 영역(50%)이 겹치는 객체 k를 기준으로 선호도가 50%(100-100×0.5)로 재정의된다. 그러나 만일 객체 p의 선호도가 이미 감소하여 60%라면, 60-60×0.5 로 30%가 된다.

정리 4. 선호도 재정의 방법 2: CA 기준

객체 u가 후보객체 p의 CA영역에 겹치면, p의 선호도는 u가 겹쳐진 영역만큼 감소한다. □

3.4 RSRD기법의 처리과정

이전까지 우리는 영역객체 환경에서의 리버스 스카이라인 질의를 처리하기 위한 2가지 가지치기 기법과 겹침 관계에 따른 선호도 재 정의 기법에 대하여 설명하였다. 이러한 기법들을 적절한 절차와 방법으로 적용하면 효율적으로 리버스 스카이라인 질의를 처리할 수 있다. 이번 장에서는 RSRD기법의 전체적인 처리과정을 간단히 수도코드(sudo-code)로 기술하고 설명하도록 한다.

3.4.1 RSRD기법

RSRD기법은 모든 데이터셋을 R*-tree로 색인한 후, 질의점과의 거리를 기준으로 정렬된 큐인 힙(Heap)에 엔트리들을 in-queue, de-queue하는 과정을 통해 처리한다. 우선 질의 q가 요청되면, R*-tree의 루트노드의 모든 엔트리들을 질의점까지의 거리가 가까운 순서대로 힙에 삽입한다(알고리즘 1 : line 3). 이후, 힙의 리스트에 더 이상 엔트리가 존재하지 않을 때까지 힙의 첫 번째 엔트리(e)부터 리버스 스카이라인 판별과정을 수행한다(line 4-5).

```
알고리즘 1.RSRD Algorithm
1:  procedure RSRD(R*-tree R, Query point q)
2:  RSL←∅, C←∅, F←∅,
3:  Insert all entries of a root R in a heap H sorted by distance
      from q
4:  while H is not empty do
5:    Remove top entry e from H
6:    if e is globally dominated by some point in C then
7:      PruningMethodUsingCA(RSL,e);
8:      discard e
9:    else [e is not globally dominated]
10:     if e is an intermediate entry then
11:       for all child ei of e do
12:         if ei is globally dominated by C then
13:           discard ei
14:         else
15:           insert ei into H
16:         end if
17:       end for
18:     else [e is a leaf entry]
19:       PruningMethodUsingSA(RSL,C,e);
20:       PruningMethodUsingCA(RSL,e);
21:     end if
22:   end if
23: end while
24: Refinement(RSL,F);
25: RecomputePreference_2(RSL);
26: RETURN RSL
27: end procedure
```

힙의 첫 번째 엔트리로써 선택된 e가 이전까지 찾아진 후보객체들에 의해 지배된다면, e는 리버스 스카이라인이 아니므로 제거된다. 단, 정리 2에 따라 e에 의해 리버스 스카이라인에서 제외되거나 선호도가 감소되는 후보객체가 존재할 수 있으므로 PruningMethodUsing-CA함수를 통해 확인한다(line 7-8). 그러나 e가 후보객체들에게 지배되지 않는다면, e가 R*-tree의 중간노드(intermediate entry)인지 리프노드 (leaf entry)에 따라 다음의 처리과정을 거친다.

e가 중간노드이면, e의 자식노드(child entries: ei)들의 지배관계를 확인하여 이전과 동일한 방법으로 후보객체들에 의해 지배된다면, ei를 삭제하고, 그렇지 않으면 힙에 삽입한다(line 11-15). 만일 e가 리프노드 이면 PruningMethodUsingSA함수를 이용하여 정리 1에 따라 현재 유효화된 SA영역에 엔트리 e가 포함되는지의 여부를 확인하고, PruningMethodUsingCA함수를 이용하여 정리 2에 따라 e에 의해 영향을 받는 이전의 후보객체들을 처리한다(line 19-20).

힙에 더 이상 엔트리가 없어 모든 후보객체를 선정하였다면, Refinement 단계를 수행해야 한다. Refinement 함수는 현재까지 선정한 후보객체 중 리버스 스카이라인이 아닌 객체가 포함되었는지 다시 한번 확인함으로써 최종적으로 리버스 스카이라인 결과객체를 결정하게 된다(line 24). 이렇게 선정된 리버스 스카이라인 객체는 자신의 CA영역의 겹침 관계를 확인하여 정리 4에 따라 선호도를 재 계산함으로써 RSRD기법을 마치게 된다

3.4.2 PruningMethodUsingSA

알고리즘 2의 PruningMethodUsingSA함수는 SA을 이용한 가지치기 기법을 구현한 함수로써 객체 e와 SA영역의 겹침 관계에 따라 e의 후보객체 선정 및 선호도 재 정의를 수행한다. 알고리즘 4의 OverlapRelation-Check 함수를 이용하여 e가 SA영역에 포함되는지(contain), 겹치는지(intersect) 또는 완전히 포함되지 않는지를 판별하여 그에 따라 다른 처리과정을 수행한다(알고리즘 2 : line2). 만일 포함관계이면 후보객체로써 e를 C, RSL 배열에 삽입 하고 e에 따라 SA영역을 수정한다(line 3-5). 반대로 완전히 포함되지 않으면 e는 리버스 스카이라인이 될 수 없다. 그러나 후에 Refinement 단계에서 후보객체들을 최종 확인하는데 사용하기 위해 현재 SA영역의 기준이 되는 후보객체 c에 의해 지배되지 안는 객체만 F배열에 유지한다(line 11-15). 만일 e가 SA에 겹쳐진다면, 이를 C, RSL배열에 삽입하기 전에 e의 선호도를 재 계산하고, SA영역을 수정해야 한다. 또한 SA영역에 포함되지 않는 e.Tp(또는 e.Bp)을 SA와 만나는 점인 Tp'(또는 Bp')로 수정해야 한다(line 6-10).

```
알고리즘 2.PruningMethodUsingSA
1:  procedure PruningMethodUsingSA(ResultList RSL,
      CandidateList C, Entry e)
2:    state = OverlapRelationCheck (SA, e);
3:    if state == 'contain' then
4:      insert e into C and RSL list
5:        alter SA with e
6:    else if sate == 'intersect' then
7:      RecomputePreference_1(e,SA);
8:      change Tp which is not contained by SA to Tp' // or
        Bp
9:      alter SA with e
10:     insert e into C and RSL list
11:   else [fully not contained]
12:     if e is not dominated by c owning Sp then
13:       insert e into F list
14:     else
15:       discard e
16:     end if
17:   end if
18: end procedure
```

3.4.3 PruningMethodUsingCA

알고리즘 3의 PruningMethodUsingCA함수는 CA을 이용한 가지치기 기법을 구현한 함수로써 객체 e와 후

```
알고리즘 3.PruningMethodUsingCA
1: procedure PruningMethodUsingCA(ResultList RSL,
     Entry e)
2:   for all entry into RSL then // r ∈ RSL
3:     state = OverlapRelationCheck(r.CA, e);
4:     if state == 'contain' then
5:       discard r with RSL list
6:     else if state == 'intersect' then
7:       add e into r.intersection list
8:     end if
9:   end for
10: end procedure
```

```
알고리즘 4.OverlapRelationCheck
1: procedure OverlapRelationCheck(Region R, Entry e)
2:   if e.Tp ∈ R and e.Bp ∈ R then
3:     return 'contain'
4:   else if e.Tp ∈ R or e.Bp ∈ R then
5:     return 'intersect'
6:   else [e.Tp and e.Bp are not contained into R fully]
7:     return 'non-candidate'
8:   end if
end procedure
```

```
알고리즘 5.RecomputePreference_1
1: procedure RecomputePreference_1(Entry e, Region
     SA)
2:   e.prefernece = a percent contained into SA
3: end procedure
```

```
알고리즘 6.RecomputePreference_2
1: procedure RecomputePreference_2(ResultList RSL)
2:   for all entry into RSL then // r ∈ RSL
3:     Entry e = maxIntersectRegion(r.intersection);
4:     r.preference = r.preference - r.preference×
     PercentOfIntersectRegion(e,r);
5:   end for
6: end procedure
```

보객체 r의 CA영역의 겹침 관계에 따라 r의 후보객체 제거를 수행한다. OverlapRelationCheck을 통해 이들간의 겹침 관계를 판별 후 e가 후보객체 r의 CA영역에 포함되면 r은 후보객체에서 제외고, 포함되지 않으면 유지된다(line 3-5). 그러나 겹쳐진다면, e의 정보를 r의 intersection 배열에 저장함으로써 마지막에(알고리즘 1 : line25) 가장 많이 겹치는 객체만을 대상으로 r의 선호도를 재 계산할 때 사용한다(line 6-7).

알고리즘 5는 정리 3에 따라 SA에 겹치는 면적에 따른 선호도 재 정의 과정을, 알고리즘 6은 정리 4에 따라 CA에 겹치는 면적에 따른 선호도 재 정의 과정을 나타낸다.

3.4.4 Refinement

알고리즘 7의 Refinement함수는 후보객체들을 한번 더 확인하여 최종적으로 결과객체로 선정하기 위해 F배열에 저장된 객체들과 후보객체의 CA을 확인하는 과정을 나타낸다. 이 단계는 질의점과 가까운 순서대로 엔트리를 처리하는 본 기법의 구조상, 놓칠 수 있는 CA포함 관계를 한번 더 확인함으로써 올바른 결과객체를 선정할 수 있다.

```
알고리즘 7.Refinement
1: procedure Refinement (ResultList RSL, List F)
2:   for all entry into F then // f ∈ F
3:     PruningMethodUsingCA(RSL,f);
4:   end for
5: end procedure
```

4. 실험 및 결과

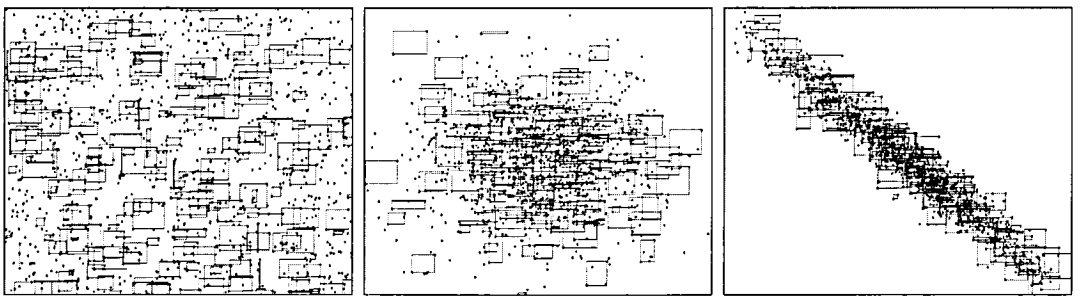
RSRD기법은 영역객체의 리버스 스카이라인 질의 처리를 위한 최초의 접근으로써 그 비교 대상이 되는 기법이 존재하지 않는다. 그러므로 RSRD기법이 영역객체 환경에서의 리버스 스카이라인 질의를 명확히 수행하는지 확인하고, 전체 데이터셋에서 영역객체와 점객체와의 비율이 성능에 미치는 영향을 분석한다. 또한 본 기법의 핵심이 되는 영역 겹침 현상에 따른 선호도 재 평가로 인한 결과객체의 차이를 분석하는데 중점을 두었다.

실험에 필요한 데이터셋은 그림 11과 같이 일반 점객체와 영역객체를 혼합하여 uniform, Gaussian, anti-correlate분포, 다양한 차원으로 생성하여 실험하였다. 이는 자체적인 데이터 생성기를 구현하여 사용하였으며, 객체들의 조건은 랜덤하게 정의하였다. 단, 영역객체의 영역범위는 객체가 조건으로 가질 수 있는 조건 값의 범위의 1/10로 고정하였다. 즉, 본 실험의 경우 한 차원에서 객체 조건의 min, max 차는 10만 이하이다.

모든 실험은 Intel Quad Q6600 2.4GHz CPU, SATA ST340062 400GB hard disk, 4GB메모리를 가지는 32bit window XP 환경의 컴퓨터에서 수행하였다.

4.1 리버스 스카이라인 결과 개수 비교

표 1은 다양한 분포와 차원을 가지는 환경에서의 리버스 스카이라인 결과객체의 개수 차를 보여준다. 이때, 실세계의 데이터의 경우 영역객체와 점객체가 조화될 것이라 예상되므로, 전체 데이터 셋 중 20%만을 영역객체로 정의하고 실험하였다. 그 결과 표 1과 같이 데이터 분포, 차원에 따른 결과객체 개수의 차이를 확인할 수 있었다. 전체 차원에서 gaussian분포의 결과가 가장 적고, uniform 분포가 가장 많으며, 저 차원에서는 적은 개수를 갖다가 고차원이 될수록 개수가 급증하는 anti-correlate 분포의



(a) Uniform (b) Gaussian (c) Anti-correlate

그림 11 실험 데이터 셋의 분포 예제 (1000개, 영역객체 50% 분포)

표 1 리버스 스카이라인 개수 비교(n=100K)

dimension	uniform	gaussian	anti_corr
2	1	2	0
3	5	5	1
4	20	18	9
5	61	40	38
6	184	87	83
7	369	165	247
8	584	297	556
9	1708	475	1194
10	2947	661	2350

이러한 변화는 이전의 점객체 환경에서의 결과와 동일하다.

4.2 영역객체와 점객체 비율에 따른 결과 개수 비교

그림 12는 전체 데이터셋 중 영역객체가 차지하는 비율에 따른 리버스 스카이라인 결과객체의 개수 차를 보여준다. 정확한 결과를 위하여 10개의 새로운 데이터셋

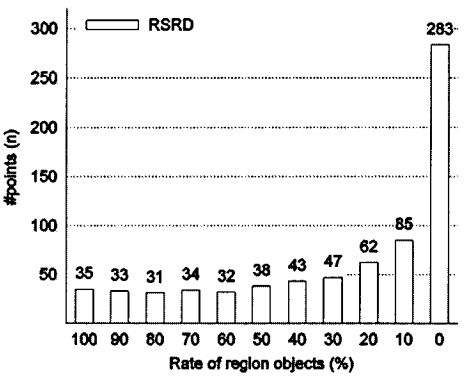


그림 12 전체 데이터셋 중 영역객체와 점객체 비율에 따른 리버스 스카이라인 결과객체(uniform, 5d, 100K)

을 생성하고 서로 다른 질의를 각각 10번씩 수행한 평균을 구하였으나, 질의점의 조건에 따라 영역객체의 겹치는 경우를 예측할 수 없어, 영역객체가 100%~50%

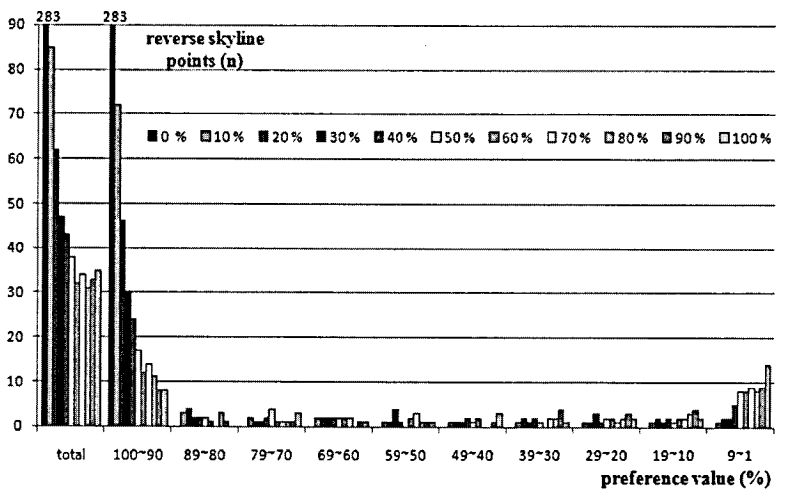


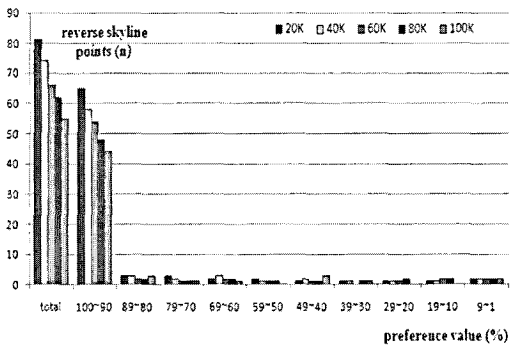
그림 13 전체 데이터셋 중 영역객체의 비율에 따른 결과객체의 선호도 분포(uniform, 5d, 100K)

포함되는 환경에서 비슷한 결과 개수가 확인되었다. 그러나, 점객체의 포함확률이 높아질수록 질의 결과 개수가 많아지는 것을 확실히 알 수 있다.

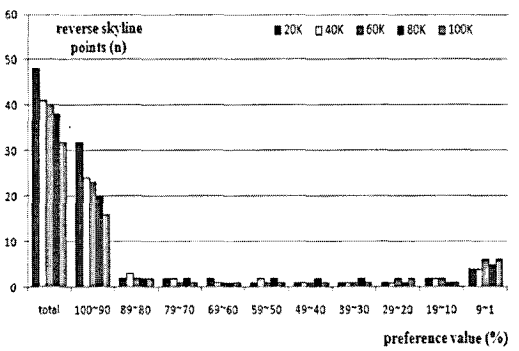
4.3 영역객체 비율에 따른 결과객체의 선호도 분포

그림 12이 영역객체 점유비율에 따른 총 결과객체의 개수를 보여준다면, 그림 13에서는 결과객체 각각의 겹침 현상에 따라 재 계산된 선호도 값의 분포 비율을 확인할 수 있다. 이때 그림 13의 total은 그림 12의 개수와 같고. 이 실험을 보면, 영역객체의 비율이 많아질수록 총 결과객체의 수는 점점 줄어들고, 겹침 현상이 보다 많이 발생하여 낮은 선호도를 갖는 객체가 결과로 포함될 확률은 높아진다. 그러나 전체 선호도 분포에 규칙성 있는 영향을 주지는 않는다.

이러한 특징은 그림 14의 영역객체의 20% 점유 환경과, 50% 환경에서 데이터셋의 크기를 다르게 했을 때 선호도 분포 실험 결과를 보면 보다 명확하게 확인할 수 있다. 또한 데이터셋의 크기가 결과객체 수에 영향을 미치지만, 전체 선호도 분포에는 관련이 없는 것을 알 수 있다.



(a) 전체 데이터셋 중 영역객체가 차지하는 비율 20%



(b) 전체 데이터셋 중 영역객체가 차지하는 비율 50%

그림 14 전체 리버스 스카이라인 결과객체 중 선호도에 따른 차이(uniform, 5d)

4.4 질의처리 시간

그림 15는 영역객체의 점유비율에 따른 순수 질의처리 시간을 보여준다. 동일 분포, 동일 차원, 동일한 크기의 데이터셋을 영역개체의 점유비율만 다르게 하여 실험을 수행한 결과, 전체 데이터셋에 영역객체가 많이 포함될수록 질의처리 시간이 증가는 것을 확인할 수 있다. 즉, 영역객체가 0% 포함되는 경우(4.5s)와, 100% 포함되는 경우(22.7s) 질의시간이 약 5배 차이 나는 것을 알 수 있다. 이러한 현상의 원인은 영역객체가 많아질수록 겹침 관계에 따른 정의 1와 2의 처리 횟수가 증가하며, 선호도 재 계산 비용 또한 늘어나기 때문이다. 즉, 영역객체의 비율이 늘어날수록 질의 결과의 수는 줄어들지만(그림 12참조), 질의처리 시간은 증가한다.

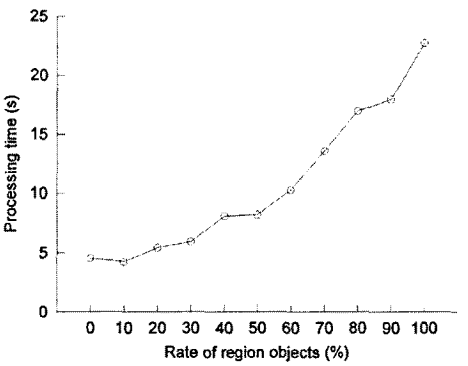
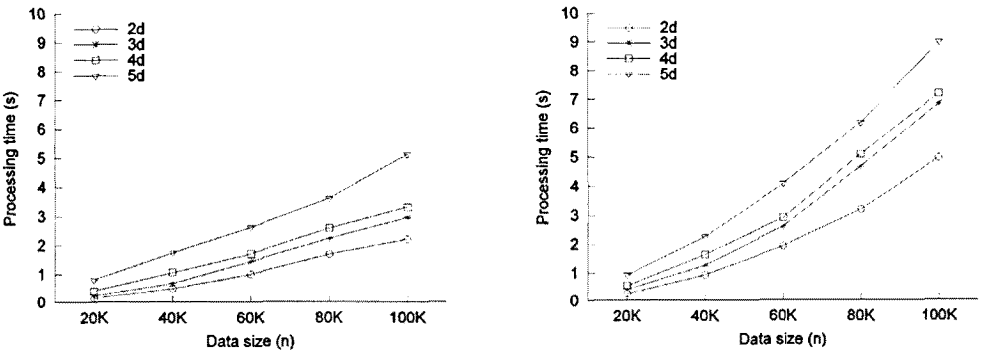


그림 15 영역객체의 점유율에 따른 질의처리시간 (uniform, 5d, 100K)

그림 16은 영역객체의 비율이 각각 20%, 50%일 때 데이터셋의 크기가 증가함에 따른 질의처리 시간을 보여준다. 두 환경 모두 각 차원에서 데이터셋의 크기가 증가할수록 질의처리 시간이 증가한다. 다만, 그림 16에서 확인한 바와 같이 영역객체의 점유율이 증가한 만큼 (20% < 50%)의 질의처리 시간의 차가 존재하였다.

그림 17은 기존의 점객체만을 고려한 리버스 스카이라인 질의처리 기법인 RSSA기법과 RSRD기법의 실행시간을 보여준다. RSSA기법은 차원이 높아질수록 처리시간이 급격히 증가하는 반면, RSRD기법은 전체 데이터셋의 영역객체 포함비율과 상관없이 일정한 성능을 유지하는 것을 볼 수 있다. 즉, 점객체만으로 이루어진 데이터셋 환경(RSRD-0%)에서는 모든 차원에서 RSSA기법보다 우수하고, 영역객체만으로 이루어진 환경(RSRD-100%)에서는 저 차원에서는 성능이 떨어지나 고차원에서도 일정한 성능을 유지하기 때문에, 결과적으로 RSSA기법보다 좋은 성능을 가지는 것을 볼 수 있다. 이때 RSRD-0%인 환경에서의 리버스 스카이라인 처리 기법



(a) 영역객체 점유 비율 20% (b) 영역객체 점유 비율 50%

그림 16 질의처리 시간(uniform, 100K)

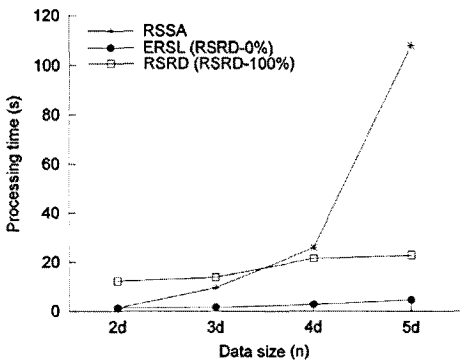


그림 17 RSSA와 RSRD기법의 질의처리 시간 (uniform, 100K)

은 ERSL과 동일하다. 기존의 점객체환경만을 지원하는 RSSA, ERSL기법과의 비교로 제안하는 기법의 성능향상을 증명할 수는 없으나, 영역객체를 지원함에도 불구하고 기존의 리버스 스카이라인 질의처리 기법과 비교하여 손색이 없음을 알 수 있다.

5. 결 론

RSSA기법을 시작으로 리버스 스카이라인 질의 처리를 위한 다양한 기법들이 제안되었다. 그러나 이러한 기법들은 점객체 즉, 객체가 각 차원 별로 하나의 조건 값을 가지는 환경만을 고려하였기 때문에 조건이 범위로 주어지는 영역객체 환경에서의 리버스 스카이라인 질의 처리에는 한계가 있다.

본 논문에서는 영역객체 환경에서의 효율적인 리버스 스카이라인 질의 처리를 지원하기 위하여 RSRD기법을 제안하였다. 이는 이전의 점객체 환경에서 효율이 좋은 ERSL기법을 영역객체 환경도 지원 가능하도록 확장한 기법이다. 그러므로 ERSL기법과 동일한 방법으로 R*-tree로 색인된 데이터셋에 대하여 질의점과의 거리를 기준으로 정렬되는 우선순위 큐(Heap)를 이용하여 결과를 도출한다. 이는 리버스 스카이라인 결과객체가 포함되어 있는 디스크 페이지만 접근하기 때문에 낭비가 적어 좋은 성능을 기대할 수 있다. 또한 SA영역과 CA영역을 이용한 가지치기 기법을 제안하여 최소한의 후보객체를 선정하기 때문에 실행시간의 낭비를 절감할 수 있다. 그 밖에도 영역객체의 겹침 관계에 따른 선호도 재 정의 기법을 제안하여, 결과객체 각각의 선호도 값에 차별을 두어 질의자가 선택적으로 결과객체를 사용하도록 선택권을 제공한다는 장점이 있다.

RSRD기법은 영역객체를 위한 최초의 기법이기 때문에 성능을 비교하기 위한 다른 대상기법들이 존재하지 않는다. 그러므로 RSRD기법의 성능 및 질의결과에 영향을 주는 조건이 무엇이며, 그에 따라 소모되는 실행시간을 측정하였다. 그 결과 전체 데이터셋 중 점객체보다 영역객체의 포함비율이 높아질수록 결과객체의 개수는 줄어들고, 그 중 낮은 선호도를 갖는 객체의 수가 증가하였으며, 질의처리 시간은 더 오래 걸리는 것을 확인하였다. 즉, 결과적으로 RSRD기법은 데이터의 개수, 차원, 분포조건에 상관없이 높은 성능을 유지하나, 데이터셋의 점객체와 영역객체의 비율에 따라 전체 성능에 영향을 받는다는 것을 확인하였다.

영역객체 환경에서의 리버스 스카이라인 질의는 영역의 겹침 현상에 따라 결과의 개수 및 실행시간이 모두 결정되므로 기법의 정량화된 성능을 측정하기 어렵다. 그러나 이러한 접근이 최초이며, 점객체와 영역객체가 적절한 비율로 통합되는 환경에서의 알맞은 적용에 크게 의미 있을 것이라 예상된다. 또한 기존의 기법과 달리 결과객체에 차별을 두어 질의자가 선택적으로 정보를 사용할 수 있도록 지원한다는 특징이 있어 또 다른 이점을 기대할 수 있다.

참 고 문 헌

[1] S. Borzsonyi, D. Kossmann, and K. Stocher, "The Skyline Operator," *ICDE*, pp.421-430, 2001.

[2] D. Papadias, Y. Tao, G. Fu and B. Seeger, "An optimal and progressive algorithm for skyline queries," *ACM SIGMOD*, pp.467-478, 2003.

[3] D. Papadias, Y. Tao, G. Fu and B. Seeger, "Progressive skyline computation in database systems," *ACM TODS*, pp.41-82, 2005.

[4] X. Lian and L. Chen, "Dynamic Skyline Queries in Metric Spaces," *ACM ICPS*, vol.261, pp.333-343, 2008.

[5] J. Kim and Y. Park, "An Efficient Pruning Method for Subspace Skyline Queries of Moving Objects," *Journal of KIISE: Databases*, vol.35, no.2, pp.182-191, 2008. (in Korean)

[6] A. Han and Y. Park, "Efficient Reverse Skyline Query Processing of Companies perspective," *Proc. of the 35th KIISE Fall Conference*, vol.2, no.2(C), pp.49-54, 2008. (in Korean)

[7] A. Han and Y. Park, "Efficient Reverse Skyline Processing using Branch-and-Bound," *Journal of KISSE: Database*, vol.37, no.1, pp.12-21, 2010. (in Korean)

[8] E. Dellis and B. Seeger, "Efficient Computation of Reverse Skyline Queries," *VLDB*, pp.291-302, 2007.

[9] X. Lian and L. Chen, "Monochromatic and Bichromatic Reverse Skyline Search over Uncertain Databases," *ACM SIGMOD*, pp.213-226, 2008.

[10] C. Li, B. B. Ooi, A. K. H. Tung and S. Wang, "DADA: a Data Cube for Dominant Relationship Analysis," *ACM SIGMOD*, pp.659-670, 2006.

한 아

2007년 명지대학교 정보공학과(공학사) 2009년 명지대학교 컴퓨터공학과(공학석사). 2009년~현재 명지대학교 컴퓨터공학과(박사과정). 관심분야는 Mobile DB, Spatial DB, Information retrieval, Query optimization, High performance computing, Skyline queries, Reverse Skyline queries

이 종 혁

2006년 명지대학교 컴퓨터공학과(공학석사). 2009년~현재 명지대학교 컴퓨터공학과(박사과정). 관심분야는 Mobile DB, 위치기반 서비스, Skyline queries

박 영 배

1993년 서울대학교 대학원 컴퓨터공학과(공학석사). 1990년~1992년 명지대학교 전자계산소장. 1981년~현재 명지대학교 컴퓨터공학과 교수. 관심분야는 Mobile DB, Spatial DB, 한국어 정보처리, Skyline queries, Reverse Skyline queries