분기한정법을 이용한 효율적인 리버스 스카이라인 질의 처리

(Efficient Reverse Skyline Processing using Branch-and-Bound)

한 아 † 박 영 배 ††
(Ah Han) (Young Bae Park)

요 약 최근 이슈가 되고 있는 "정보 중심의 서비스"는 정보(정보 제공자)가 질의의 주체가 되어 정보 스스로 자신이 필요할 것 같은 고객을 찾아 제공되는 새로운 서비스 이다. 이러한 서비스는 정보를 사용할 가능성이 높은 특정한 고객들에게만 선택적으로 제공하기 때문에 적은 비용으로 높은 효과를 얻을 수 있다. 정보 중심의 서비스를 처리하기 위해 리버스 스카이라인기법을 제안한다. 리버스 스카이라인 기법 중 RSSA (Reverse Skyline using Skyline Approximations) 기법은 가장 정형화되고 성능이 증명된 방법이다. 그러나 메모리의 낭비와 실행시간의 낭비가 서로 상충작용을 하여 반복적인 한계를 유발하는 문제점이 있다.

본 논문에서는 리버스 스카이라인을 보다 효율적으로 구하기 위한 ERSL(Efficient Reverse Skyline) 알고리즘을 제안한다. ERSL 알고리즘은 BBS(Branch and Bound Skyline) 알고리즘을 발전시킨 새로운 기법으로 메모리와 실행시간의 낭비를 최소화 하고, 객체의 변화에 유연하여 추가적인 처리과정이 필요 없는 장점이 있다. ERSL의 성능을 평가하기 위해 대상객체의 수의 변화와 차원의 변화에 따른 실행시간을 측정하는 모의실험을 수행하였다. 그 결과 ERSL기법은 데이터양과 차원의 변화에 크게 영향을 받지 않고 일정한 성능을 유지하여 가장 효율적인 기법으로 증명되었다.

키워드 : 스카이라인 질의, 리버스 스카이라인 질의

Abstract Recently, "Service of information perspective" that is an important issue is that a company searches customers that interested in certain information and the company offers information to the customers. This service can gain high effects by low cost because of supporting selective information. In most recently, Reverse Skyline using Skyline Approximation(RSSA) is proposed to process services of information provider's perspective. RSSA has problem to defects about waste of processing time and memory.

In this paper, Efficient Reverse Skyline(ERSL) Algorithm is proposed for Efficient processing the Skyline. ERSL is new Algorithm using Branch and Bound Skyline(BBS) reduces the waste of processing time and memory. When we execute the variety experimentation to valuation ERSL algorithm's capacity, It is proved the best efficient algorithm among the others because ERSL is flexibly kept the established capacity.

Key words : Skyline query, Reverse Skyline query

† 학생회원 : 명지대학교 컴퓨터공학과
ahhan.korea@gmail.com
†† 종신회원 : 명지대학교 컴퓨터공학과 교수
parkyb@mju.ac.kr
논문접수 : 2009년 3월 31일
심사완료 : 2009년 10월 14일

정보과학회논문지: 데이타베이스 제37권 제1호(2010.2)

1. 서 론

컴퓨터 및 네트워크 기술의 빠른 발전으로 "정보"는 현대인의 필요조건이 되었다. 그러므로 다양한 분야에서 정보의 사용과 제공방법에 대해 관심을 가지고 접근하고 있다. 그 중 정보의 제공방법에 대한 연구는 중요한 발전을 거듭하여 다양한 정보 서비스를 제안하였다.

각양각색의 정보 서비스는 사용자인 우리에게 유용한 정보를 보다 쉽게 얻도록 지원한다. 그러나 사용자는 이에 만족하지 않고 보다 폭넓은 서비스를 요구하고 있다.

그러므로 정보제공자는 사용자의 요구사항을 만족시키기 위해 "어떤 정보를 제공할 것인가?"처럼 정보의 유형만을 집중하던 과거와 달리, "누구에게 어떻게 제공할 것인가?"처럼 정보를 받을 대상과 제공과정에도 관심을 가지게 되었다.

새로운 고려사항을 반영한 정보서비스 중 우리가 관심을 가지는 서비스는 "정보 중심의 서비스 : Focus is information"이다. 정보 중심의 서비스란 정보(정보제공자)가 질의의 주체가 되는 서비스로써, 정보를 사용하는 고객이 질의의 주체가 되는 "정보사용자 중심의 서비스 : Focus is customers"와 다르다. 이는 사용자가 무수히 많은 정보의 바다에서 자신이 필요한 정보를 찾아가는 것이 아니라, 정보 스스로 자신이 필요할 것 같은 고객을 찾아 제공되는 새로운 서비스인 것이다. 이렇게 해당 고객에게만 필요한 정보를 제공한다면 적은 비용으로 높은 서비스 효과를 얻을 수 있으며 또한, 언제나 유용한 정보를 알고자 하는 고객의 욕구도 만족시킬 수 있는 장점이 있다. 이러한 접근을 "선택적 광고 마케팅", "고객 맞춤 서비스"라고 한다.

사용자가 질의의 주체가 되었던 기존의 서비스는 스카이라인(Skyline) 기법으로 처리하였다. 그러나 정보가 질의의 주체가 되는 정보 중심의 서비스를 스카이라인 기법으로 처리하는 것은 불가능하다. 그러므로 리버스 스카이라인(Reverse Skyline) 기법을 제안한다.

리버스 스카이라인 기법은 스카이라인의 기본 개념이 유지된 기법이다. 즉, 질의자의 정보에 따라 원하는 정보를 제안해주고 선택은 사용자에게 맡기는 기법의 기본 방향은 변함이 없다. 다만 스카이라인 기법에서의 질의자(정보사용자가 원하는 정보)와 객체(제공될 정보/정보제공자)를 바꾸어 질의한다.

리버스 스카이라인 기법은 현재 VLDB(2007)에 발표된 RSSA(Reverse Skyline using Skyline Approximate)기법이 최초이며, 유일한 기법이다[1]. 모든 객체의 스카이라인을 디스크에 유지하고 있어 메모리 낭비가 발생하고 객체의 추가·삭제·수정에 따라 영향을 받는 스카이라인을 수정하기위한 추가적인 처리과정이 필요하다. 메모리 낭비를 절감하기 위해 근삿값 방법을 제안하였으나, 2차원 데이터 조건에서만 제한적으로 적용 가능하여 한계가 있다. 그러므로 RSSA기법은 메모리 최소화 사용과 처리시간 절감의 모든 면에서 만족스러운 성능을 기대하기 힘들 것이라 생각된다.

본 논문에서 제안하는 ERSL(Efficient Reverse Skyline)기법은 RSSA기법의 단점을 최소화한 새로운 기법이다. ERSL기법은 스카이라인을 가장 효율적으로 구하는 BBS(Branch and Bound)기법[2]을 바탕으로 발전시킨 방법으로써 다차원 데이터 환경에서의 리버스 스카이라인 질의처리가 가능하다. 그러므로 결과객체가 포함된 디스크 페이지를 한번만 접근하여 실행시간의 낭비를 최소화 하는 BBS기법의 장점을 유지한다. 또한 필수적인 메모리만을 사용하고, 객체의 변화에도 유연하다. 이러한 ERSL기법의 장점으로 저장 공간이 작은 모바일 시스템에 응용하기 적합하다.

ERSL기법의 향상된 성능을 증명하기 위하여 대상객체의 수의 변화와 차원의 변화에 따른 실행시간을 측정하는 모의실험을 수행하였다.

2. 스카이라인 질의

스카이라인 질의(Skyline Query)란 질의자(정보사용자)의 선호도를 기준으로 예측된 "관심을 가질만한 정보"의 집합이다[3,4]. "관심을 가질만한 정보"는 하나의 객체(정보)의 속성값이 다른 객체보다 나쁘지 않는 객체들로써, 이들 중 질의자가 최종적으로 사용할 정보를 선택하도록 한다[7,8].

"가격이 싸고, 지하철역까지의 거리가 가까운 집"을 구하는 고객이 있다고 가정하자. 일반적으로 지하철역까지의 거리와 가격은 반비례관계이므로 두 조건을 만족하나는 최적의 집을 알고리즘에서 하나만 결정해 줄 수 없다. 그러나 고객이 관심을 가질만한 집들을 몇 가지 제안하여 스스로 알맞은 집을 결정하도록 할 수는 있다.

스카이라인 질의의 기본은 객체들 간의 지배관계를 결정하는 것이다[5,9,10]. 그림 1은 H_9를 기준으로 다른 객체들과의 지배관계를 나타낸 것이다. H_9를 원점으로 하는 직각좌표계에서 1사분면에 위치하는 객체는 H_9가 지배하는 피지배 객체(dominatee)이다. 반면에 3사분면에 위치하는 객체는 H_9를 지배하는 지배 객체(dominator)이다. 또한 2,4분면에 위치하는 객체는 H_9와 비교할 수 없는 객체(incomparable)로써, H_9가 지배하지도 H_9를 지배하지도 않는 객체이다.

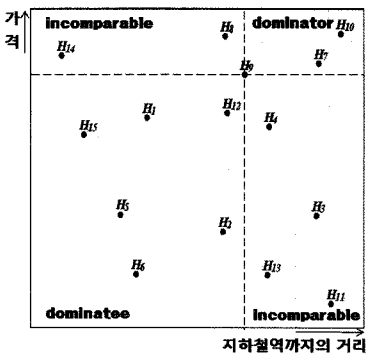
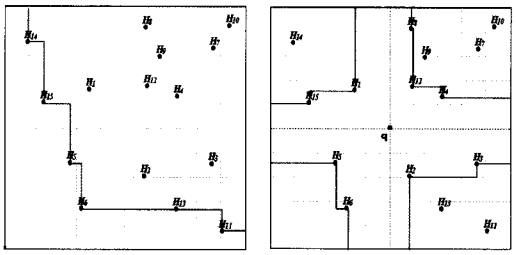


그림 1 집 {H_9}의 지배관계



(a) 오리지널 스카이라인 (b) 글로벌 스카이라인

그림 2 스카이라인

객체의 지배관계를 기준으로 그림 1의 15개의 객체의 스카이라인을 구하면 그림 2와 같다. 그림 2는 <가격, 지하철역까지의 거리>의 두 가지 속성값만을 고려하여 고객이 관심을 가질만한 집들을 스카이라인으로 나타낸 것이다. 이때 고객이 찾는 집의 조건이 "무조건 싸고 가까운 집"으로써 질의점이 원점인 경우 스카이라인은 그림 2(a)와 같이 {H_5,H_6,H_{11},H_{13},H_{14},H_{15}}이다. 그러나 "가격이 5천만원, 거리가 100m인 집"으로써 질의점이 원점이 아닌 경우는 지원하지 않는다. 그러므로 이러한 조건의 환경에서의 스카이라인은 글로벌 스카이라인(Global Skyline : GSL)으로 구할 수 있다[1]. GSL기법은 질의점 기준 각 사분면단위로 각각 스카이라인을 구하는 방법으로써, 예제의 결과는 그림 2(b)와 같이 {H_1,H_2,H_3, H_4,H_5, H_6,H_8,H_{12},H_{15}}이다.

3. 리버스 스카이라인 질의

리버스 스카이라인 질의(Reverse Skyline Query : RSL)는 "제공될 정보에 관심을 가질만한 고객"의 집합이다. "제공될 정보에 관심을 가질만한 고객"은 질의점을 스카이라인으로 포함하는 객체들로써, 정보가 자신을 제공할 고객을 선택하도록 한다. 그러므로 리버스 스카이라인 기법은 스카이라인의 질의점과 객체를 바꾸어 처리하는 방법이다[11].

"가격이 5천만원이고, 지하철역까지의 거리가 100m인 집"을 팔고자 한다고 가정하다. 이 집의 판매수익을 보다 많이 남기기 위해 부동산 중개업자는 집을 구하는 고객들 중 이 집에 관심을 가질만한 고객을 선별한다. 집을 사고자 하는 고객 15명이 원하는 정보가 그림 3과 같을 때, 리버스 스카이라인 결과는 {H_1,H_2,H_5}이다. 이러한 RSL의 결과객체는 질의점에 관심있는 객체들로써 중개업자가 {H_1,H_2,H_5}에 해당하는 고객에게 집을 권한다면 효율적으로 거래가 성사될 것이다. 이런 방법은 모든 고객이 아닌 필요한 고객에게 선택적으로 제공하기 때문에 적은 비용으로 높은 서비스 효과를 얻을 수 있다.

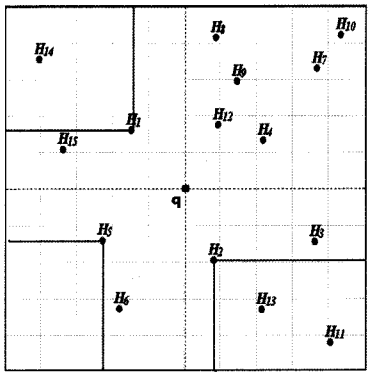


그림 3 리버스 스카이라인

3.1 Greedy Reverse Skyline 기법

Greedy Reverse Skyline(GRSL)기법은 리버스 스카이라인을 구하는 가장 기초적인 방법이다. GRSL기법은 대상객체의 스카이라인에 질의점이 포함되는지의 여부로 리버스 스카이라인을 판별한다[1]. 즉, 객체 p의 스카이라인에 질의점이 포함되면 p는 리버스 스카이라인이고 반대이면 리버스 스카이라인이 아닌 것이다. GRSL의 개념은 단순하지만 존재하는 객체 모두에 대한 스카이라인 생성 및 질의점 포함유무 판별과정이 필요하여 실행시간이 낭비되는 단점이 있다.

3.2 Branch and Bound Reverse Skyine 기법

Branch and Bound Reverse Skyline(BBRS)기법은 GRSL기법의 단점인 실행시간의 낭비를 절감하기 위해 제안된 방법이다[1]. BRSL기법은 리버스 스카이라인이 될 가능성이 있는 후보객체들만을 대상으로 판별과정을 수행하기 때문에 실행시간을 절약할 수 있다.

BRSL기법은 그림 4와 같이 GSL의 결과객체인 {H_1,H_2, H_3,H_4,H_5, H_6,H_{12},H_{15}} 10개의 객체만을 대상으로 윈도우 질의(window query)를 수행한다. 총 15개의 객체중 5개의 객체가 가지치기(pruning)된 것이다. 윈도우는 질의점과 후보객체가 이루는 거리의 두 배로 생성하며, 윈도우 안에 다른 객체가 존재하면 후보객체는 리버스 스카이라인에서 제외된다. 리버스 스카이라인의 결과는 그림 3과 같다.

BBRS기법은 방법이 단순하지만 후보객체의 위치에 따라 달라지는 윈도우의 크기를 측정하고, 영역질의를 수행하는 과정에서 실행시간이 낭비되는 단점이 있다.

3.3 Reverse Skyline using Skyline Approximations 기법

Reverse Skyline using Skyline Approximation (RSSA)기법은 VLDB(2007)에 발표된 가장 최신기법으로써 BBRS기법에서 발전된 방법이다[1]. RSSA기법은 BBRS기법의 단점의 원인이 되는 윈도우 질의의 횟수

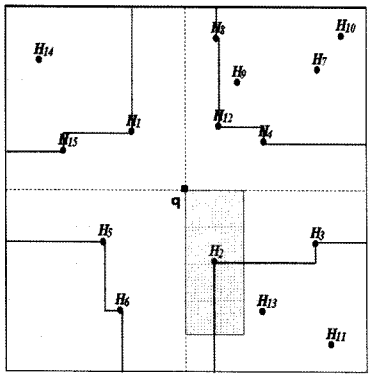


그림 4 BBRS의 윈도우 질의

를 최소화하는 것을 목표로 한다. DDR/DADR영역을 제안하여 질의점의 위치판별 만으로 리버스 스카이라인을 결정할 수 있어 불필요한 윈도우 질의의 횟수를 절감한다. 이를 위해 모든 객체에 대한 다이나믹 스카이라인(Dynamic Skyline : DSL)[6]을 선처리하여 디스크에 유지한다. 그러나 DDR/DADR영역으로도 판별되지 않는 객체는 최종적으로 윈도우 질의를 수행하여야 하며, 메모리 절감을 위한 근삿값 방법 또한 2차원 환경에서만 제한적으로 적용되므로 한계가 있다.

4. 제안하는 효율적인 리버스 스카이라인 질의 처리 기법

본 논문에서 제안하는 Efficient Reverse Skyline (ERSL)기법은 BBS기법을 기반으로 발전시킨 기법으로써 질의점으로 부터 가까운 순서대로 객체를 선택하여 해당 객체가 리버스 스카이라인인지 판별한다. 그러므로 BBS기법의 장점인 페이지의 접근의 최소화를 유지할 뿐만 아니라 한번의 R*-tree 순회로 질의점을 기준으로 한 사분면영역의 리버스 스카이라인을 구할 수 있다. 즉, 총 4번의 순회로 리버스 스카이라인 결과객체를 구할 수 있다. 그러므로 실행시간과 메모리의 사용의 모든 면에서 만족스러운 결과를 얻을 수 있으며, 객체의 추가·삭제·수정으로 인한 변화에 추가적인 처리과정 없이 재 질의 하면 된다.

다음 4.1장에서는 해당객체가 리버스 스카이라인인지 판별하기 위한 기준이 되는 중요한 영역들을 정의하고, 4.2장에서는 정의된 영역들을 이용하여 객체의 형태에 따라 처리되는 조건을 명세한다. 마지막으로 4.3장에서는 예제를 통해 ERSL기법의 처리과정을 설명한다.

4.1 리버스 스카이라인 판별조건

필요 없는 처리과정의 낭비를 최소화하기 위해 리버스 스카이라인이 될 가능성이 있는 후보객체를 선별하는 것이 중요하다.

• 정리 1. RSL(q) ⊆ GSL(q)

x∉GSL(q)를 지배하는 y가 y∈GSL(q)라고 가정하자. 모든 차원에서 i∈{1...d}: |qi-xi|≥|qi-yi| 를 만족하거나 또는 적어도 한차원에서 i∈{1...,d}: |qi-xi|≥|qi-yi| 를 만족할 때, x∉RSL(q)이다. 즉, GSL(q) 결과에 포함되지 않는 객체는 RSL(q)이 될 수 없다.

GSL의 객체는 그렇지 않은 객체를 지배한다. 이는 스카이라인기법의 기본 개념으로써, 스카이라인 객체가 다른 객체보다 조건이 우월하므로 이들만 대상객체로 고려하면 된다. 객체 20개가 그림 5와 같이 존재할 때, GSL은 {H_2,H_3,H_9,H_{10},H_{11},H_{12},H_{14},H_{18},H_{20}}로써 4개의 실선으로 표현된다. 그 중 {H_9}는 GSL으로 {H_5,H_6,H_7,H_8}을 지배한다. 그러므로 {H_5,H_6,H_7,H_8}의 대표로 {H_9}에 대해서만 리버스 스카이라인 판별과정을 수행하면 된다.

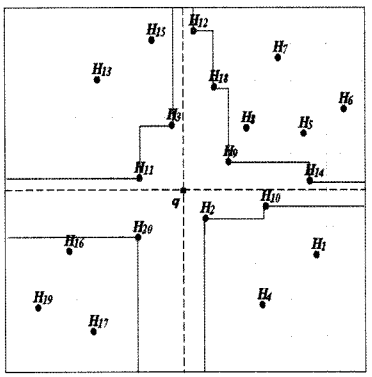


그림 5 예제를 위한 20개 객체

ERSL기법은 정의 1의 GSL을 구하면서 다른 판별과정을 동시에 진행하는 효율적인 기법이다. 그러므로 GSL을 구하기 위해 BBS기법을 적용하고, 구해진 GSL 객체 중 리버스 스카이라인을 가려내기위한 새로운 영역을 제안한다.

• 정리 2. Reverse Range 영역

x∈GSL(q), y∈GSL(q)일 때, x가 첫 번째 GSL(q)이면 모든 차원에서 i∈{1...d}: {RR|(xi+qi)/2>yi}를 만족하거나, p가 첫 번째 GSL(q)가 아니면 적어도 한 차원 이상에서 i∈{1...d}: {RR|(xi+qi)/2>yi}을 만족할 때, y∈RSL(q)이다. 즉, x의 RR영역 안에 존재하는 y만이 RSL(q)이 된다.

Reverse Range(RR)영역은 GSL객체 중 한번 더 후보객체를 절감하기 위한 영역이다. RR영역은 정의 1에 의해 선정된 후보객체를 기준으로 생성되며, 다음 정의 1에 의해 선정될 객체를 후보객체에 포함시킬 것인지를 판별하는데 사용한다. 즉, 정의 1를 만족하더라도 리버스 스카이라인이 될 수 없는 객체를 미리 가지치기(pruning)하는 것이다.

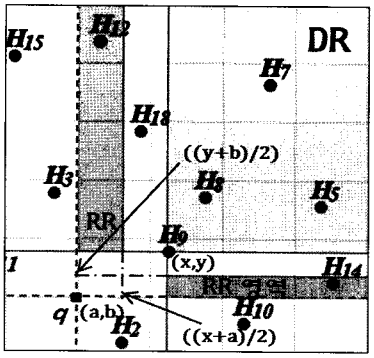


그림 6 {H_9}에 대한 RR영역

그림 6은 {H_9}가 첫 번째 GSL객체로 선정되었을 때 측정할 수 있는 RR영역을 나타낸 것이다. {H_9}의 RR영역 안에 존재하지 않는 객체는 GSL객체라도 리버스 스카이라인이 될 수 없다. $(RR \cup DR)^{-1}$영역에 위치하는 객체의 스카이라인에는 질의점 대신 질의점을 지배하는 {H_9}가 포함되기 때문이다. 그러므로 $(RR \cup DR)^{-1}$영역에 위치하는 {H_{18}}은 정의 2에 의해서 리버스 스카이라인에서 제외된다.

• 정리 3. Inclusion Decision Range 영역

$p \in P$, $s \in P$ 일 때, $i \in \{1...d\}$: $|p_i - q_i| < |p_i - s_i|$ 의 조건을 만족하는 p만이 RSL(q)이 된다. 여기서 객체 s는 p를 제외한 나머지 객체이다.

Inclusion Decision Range(IDR)영역은 정의 1,2에 의해 선정된 후보객체를 리버스 스카이라인으로 확정하기 위한 영역이다. 만일 IDR영역 안에 다른객체 s가 존재하면 p는 리버스 스카이라인에서 제외되고, 존재하지 않으면 p는 리버스 스카이라인으로 확정된다. p가 질의점보다 s의 속성값과 하나라도 더 가까우면, p의 스카이라인에 질의점 대신 질의점을 지배하는 s가 속하기 때문이다.

IDR영역은 후보객체 p를 기준으로 질의점과의 모든 차원축(2차원의 경우 x축, y축) 각각의 거리에 2배를 하여 생성한다. 그림 7은 {H_9}가 후보객체로 선정되었을 때 측정할 수 있는 IDR영역을 나타낸 것으로 {H_9}의 좌표가 (x,y)이고 질의점의 좌표가 (a,b)일 때, 질의점과 (2x-a, 2y-b)점이 이루는 사각형 모양의 영역으로 표현된다.

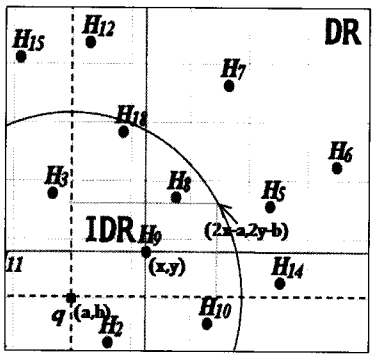


그림 7 {H_9}에 대한 IDR 영역

• 정리 4. IDR영역 고려 범위

후보객체 p의 IDR영역에 객체 s가 존재하는지에 대한 판별과정(정리 3)은 $p \in P$, $s \in P$일 때, $i \in \{1...d\}$: $2|pi-qi| > |si-qi|$ 조건이 만족하는 범위까지만 수행한다.

정리 3의 과정은 후보객체를 리버스 스카이라인 객체로 확정하는 필수적이고 중요한 과정이다. 그러나 ERSL기법이 종료될 때까지 수행할 필요는 없다. ERSL기법이 종료하기 전이라도 해당 후보객체 p가 리버스 스카이라인인지 판별되면, p의 IDR영역은 더 이상 고려하지 않아도 된다.

정의 4는 ERSL기법이 질의점과의 거리를 기반으로 생성한 우선순위 큐(heap)를 사용한다는 특징을 이용한 방법으로써 힙의 선행노드 s의 위치가 중요하다. s가 IDR영역의 대각선을 반지름으로 하는 범위(그림 7의 둥근영역) 밖에 처음 존재할 때까지 IDR영역 안에 어떠한 객체도 존재하지 않으면 이후 객체는 모두 IDR영역 안에 존재할 수 없다. 그러므로 IDR영역 안에 다른객체가 존재하면 그 즉시 p는 리버스 스카이라인이 아니며, 적어도 s가 IDR영역 대각선 범위 밖에 존재하면 p는 리버스 스카이라인으로 결정되어 IDR영역은 더 이상 의미가 없어진다.

4.2 리버스 스카이라인 판별과정

ERSL기법에서는 RR영역, RR^{-1}영역(RR영역을 제외한 영역), IDR영역을 새롭게 정의하였다. R^*-tree로 구성되어진 객체들의 정보를 바탕으로 힙의 선행노드부터 꺼내어(pop) 정의한 영역들 중 어디에 위치하는지 판별하고 다음과 같이 처리하면 효율적으로 리버스 스카이라인을 구할 수 있다. 단, 힙의 선행노드가 R^*-tree의 중간노드인지 단말노드인지에 따라 적용되는 영역과 처리방법이 달라지므로 유의해야한다.

ERSL기법은 질의점을 기준으로 각 사분면 각각 동일한 방법으로 반복 수행하므로 이해를 돕기 위해 1사분면만을 예로들어 설명한다.

우선순위 큐의 선행노드가 트리의 중간노드(Minimum Boundary Rectangle : MBR)일 경우 그림 8과 같이 4가지 경우로 나누어 처리한다.

① MBR이 DR영역 안에만 존재하는 경우:
- 중간노드의 엔트리를 삭제

② MBR이 DR영역과 겹치거나 DR영역안에 존재하며, IDR영역과 겹치거나 IDR영역안에 존재하는 경우:

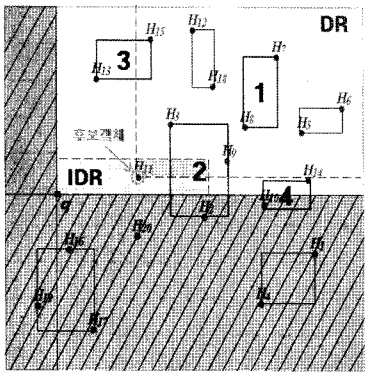


그림 8 선행노드가 중간노드인 경우

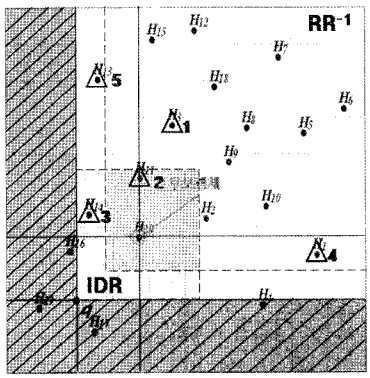


그림 9 선행노드가 단말노드일 경우

- 중간노드의 자식노드를 확장
- IDR영역 안에 존재하는 자식노드만 힙에 삽입

③ MBR이 DR영역과 겹치는 경우:
- 중간노드의 자식노드를 확장
- DR영역 밖에 존재하는 자식노드만 힙에 삽입

④ MBR이 DR영역 밖에 존재하며, 고려중인 사분면 안에 존재하거나 겹치는 경우 :
- 중간노드의 자식노드를 확장
- 현재 고려중인 사분면 (질의점을 중심으로 1사분면)안에 위치하는 자식노드만 힙에 삽입

우선순위 큐의 선행노드가 트리의 단말노드(Object)일 경우 그림 9와 같이 5가지 경우로 나누어 처리한다.

① 객체가 RR^{-1}영역 안에만 존재하는 경우(H_3):
- H_3을 힙에서 삭제

② 객체가 IDR영역 안에 존재하는 경우(H_{11}):
- H_{20}이 리버스 스카이라인이 아님이 확정.
- H_{11}을 힙에서 삭제

③ 객체가 (IDR-RR^{-1})영역 안에 존재하는 경우(H_{14}):
- H_{20}이 리버스 스카이라인이 아님이 확정.
- H_{14}를 리버스 스카이라인 후보객체로 선정

④ 객체가 (RR^{-1}-DR)영역 안에 존재하는 경우(H_1):
- H_1을 힙에서 삭제.
- H_1은 최종 RSL 후보군의 IDR 존재 유무 판별에 사용

⑤ 객체가 RR^{-1}영역 밖에 존재하는 경우(H_{13}):
- H_{13}을 리버스 스카이라인 후보객체로 선정

4.3 ERSL기법 예제 결과

ERSL기법의 처리과정을 자세히 설명하기 위해 그림 5에서 생성한 20개의 모의 객체를 대상으로 리버스 스카이라인을 구해보자. 그림 10은 20개의 대상객체를 R^*-tree로 색인하여 구조화한 것이다. 이렇게 생성한 R^*-tree의 루트노드를 힙에 삽입하고 선행노드부터 차례로 꺼내(pop)여 리버스 스카이라인 판별과정을 수행한다.

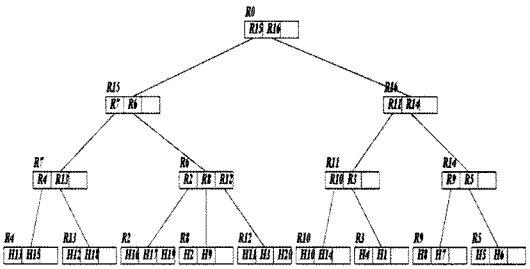


그림 10 ERSL기법 예제를 위한 20개 객체의 R*-tree

표 1 우선순위 큐(heap)의 처리과정

	action	Heap contents	RSL List
1	access root	<R15,0> <R16,3.4> <R6,R7>	RSL_Candidate={Ø} RSL_Result={Ø}
2	expand R15	<R6,0> <R16,3.4> <R7,5.3> <R12,R8,R2>	RSL_Candidate={Ø} RSL_Result={Ø}
3	expand R6 delete R2,R12	<R8,1.1> <R16,3.4> <R7,5.3> <H2,H9>	RSL_Candidate={Ø} RSL_Result={Ø}
4	expand R8 delete H2	<H9,2.7> <R16,3.4> <R7,5.3>	RSL_Candidate={Ø} RSL_Result={Ø}
5	pop H9	<R16,3.4> <R7,5.3> <R11,R14>	RSL_Candidate={H9} RSL_Result={H9}
6	expand R16	<R11,4.1> <R14,4.5> <R7,5.3> <R10,R3>	RSL_Candidate={H9} RSL_Result={H9}
7	expand R11 delete R3	<R10,4.3> <R14,4.5> <R7,5.3> <H10,H14>	RSL_Candidate={H9} RSL_Result={H9}
8	expand R10 delete H10	<R14,4.5> <R7,5.3> <H14,6.7> <R8,R5>	RSL_Candidate={H9} RSL_Result={H9}
9	expand R14 Delete R9,R5	<R7,5.3> <H14,6.7> <R4,R13>	RSL_Candidate={H9} RSL_Result={H9}
10	expand R7 delete R4	<R13,5.4> <H14,6.7> <H13,H12>	RSL_Candidate={H9} RSL_Result={H9}
11	expand R13 delete H13	<H14,6.7> <H12,8.3>	RSL_Candidate={H9} RSL_Result={H9}
12	pop H14	<H12,8.3>	RSL_Candidate={H9,H14} RSL_Result={H9,H14}
13	pop H12	Ø	RSL_Candidate={H9,H14,H12} RSL_Result={H9,H14,H12}

표 1은 질의점을 기준으로 1사분면만을 유효영역으로 선정하고 ERSL기법을 진행하는 과정을 나타낸 것이다. 제 1사분면의 결과는 {H_9,H_{12},H_{14}}이다. 동일한 방법으로 각 사분면의 리버스 스카이라인을 구한 결과는 그림 11과 같다.

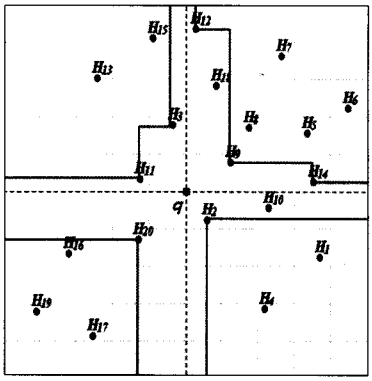


그림 11 예제를 위한 객체의 리버스 스카이라인

5. 실험 및 성능 평가

ERSL기법의 성능평가를 위해 대상객체의 모의 데이터를 생성하여 GRSL기법과 RSSA기법을 비교대상으로 실험하였다. 모든 기법은 동일하게 Intel(R) Pentium(R) 4 CPU 2.4GHz 프로세서와 1GB의 메인 메모리, 80GB의 하드 디스크를 가진 Windows XP 운영체제의 Desktop PC 환경에서 자바언어(JSDK 1.5)를 이용하여 메모리 기반으로 구현하였다.

성능비교의 기준은 알고리즘 실행시간(Processing time)으로써 micro second(ms)단위로 측정하였다. 실행시간이란 객체(정보)를 R*-tree 색인 후, 질의가 요청된 후부터 결과가 구해질 때까지의 순수 질의시간을 말한다.

대상객체는 Uniform분포로 100개에서 1000개까지 100개단위로 증가시키고, 대상객체의 변화에 따라 100번의 질의를 랜덤하게 생성하여 실행시간을 측정하였다. 또한 정확한 실험값을 위해 100번의 질의를 각각 20번 반복수행하고, min/max 값을 5개씩 제외한 총 10번의 측정값의 평균을 한회의 질의 수행시간으로 결정하였다. 즉, 하나의 결과는 2000번의 수행 후 1000번의 평균으로 구한 것이다.

5.1 대상객체 수의 변화에 따른 성능 평가

대상객체의 수의 변화에 따른 GRSL, RSSA, ERSL 기법의 성능을 평가하기 위해 표 2와 같이 실행시간의 증가율을 측정하였다. 표 2를 보면, 모든 차원에서 ERSL기법이 RSSA기법보다 대상객체의 변화에 따른 증가율이 높은 것을 알 수 있다. 그러나 이러한 결과는 대상객체가 100~300개일 때의 증가율로 지배된 결과로써, 처음부터 작은 실행시간(ERSL기법의 2차원의 경우 100개~200개의 실행시간 변화가 1002ms에서 1809ms)의 변화로 80.54% 대상객체의 변화에 민감하다고 판단하기는 모호하다. 그러나 대상객체의 차원이 증가할수록 ERSL기법의 실행시간 증가율은 23.84%로써, 73.8%인 RSSA기법보다 대상객체의 변화에 덜 민감하다는 것을 알 수 있다. 반면에 ERSL기법은 -2.06%의 증가율로 차원의 변화가 대상객체의 변화에 따른 증가율에는 가장 덜 민감하지만, 각각의 실행시간 증가율은 가장 높아 꾸준히 증가된다.

그림 12는 3차원 객체의 실행시간 측정결과를 그래프로 표현한 것이다. 그림 13은 동일한 조건으로 5차원 객체의 결과를 그래프로 표현한 것이다. 그림 12와 13에서 RSSA, ERSL기법을 비교해보면, 객체가 증가할수록 실행시간 측정값의 차이가 조금씩 가까워지는 3차원 결과와 달리, 일정한 차이를 유지하는 5차원 결과를 확인할 수 있다. 이는 ERSL기법이 다른 기법들보다 차원의 증

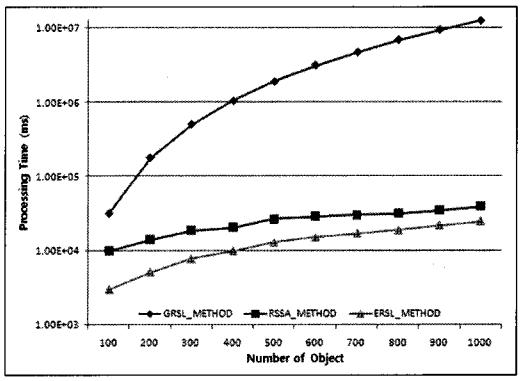


그림 12 객체 수의 변화에 따른 실행시간(Uniform-3차원)

표 2 대상객체 수의 변화에 따른 실행시간 증가율

실행시간	GRSL_METHOD				RSSA_METHOD				ERSL_METHOD			
증가율	2차원	3차원	4차원	5차원	2차원	3차원	4차원	5차원	2차원	3차원	4차원	5차원
100~200	462.29	461.44	460.67	473.75	23.09	40.36	102.74	127.52	80.54	72.66	126.29	107.43
200~300	190.40	181.74	178.82	185.52	24.10	32.96	35.27	60.93	55.72	51.87	33.59	46.59
300~400	116.65	110.97	108.99	112.06	1.57	10.47	27.45	45.68	11.32	27.00	32.47	38.59
400~500	83.94	81.92	77.40	77.05	6.93	30.20	23.15	15.36	15.69	31.13	26.86	12.73
500~600	65.02	61.92	60.13	60.66	5.08	8.08	7.55	23.30	11.36	16.88	12.28	18.43
600~700	53.48	51.87	50.22	51.14	17.56	4.61	16.89	25.69	21.83	11.59	21.78	24.58
700~800	45.88	44.05	41.90	41.15	5.81	4.44	2.44	22.75	8.47	11.19	7.01	23.84
800~900	39.06	38.45	36.56	36.45	4.99	10.52	10.64	9.64	10.90	14.59	14.49	6.08
900~1000	35.17	33.57	32.71	32.04	1.08	12.90	29.01	13.43	4.37	14.55	19.06	10.87
평균	121.32	118.44	116.38	118.87	10.02	17.17	28.35	38.26	24.47	27.94	32.65	32.13
증가율	-2.06				73.80				23.84			

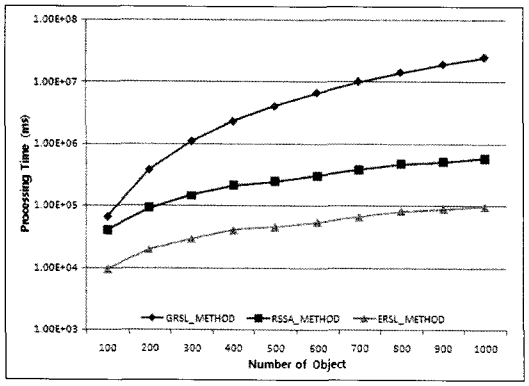


그림 13 객체 수의 변화에 따른 실행시간(Uniform-5차원)

가에 따른 대상객체 수의 변화에 점점 덜 민감해진다는 것을 확인할 수 있다.

5.2 차원 변화에 따른 성능 평가

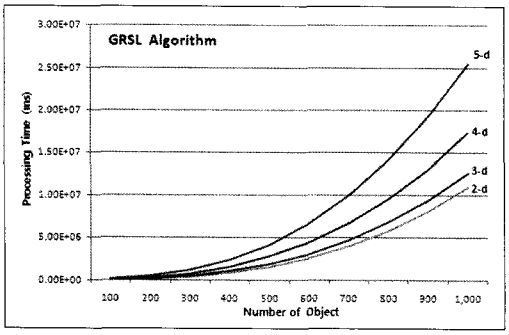
고려하는 대상객체의 속성값의 수가 증가(다차원)함에 따른 GRSL,RSSA,ERSL기법의 성능을 평가하기 위해 실행시간을 측정하였다. 그림 14는 2~5차원 객체에 대한 각 기법의 실행시간을 측정한 결과를 그래프로 표현한 것이다.

차원변화에 따른 GRSL기법의 실행시간 실험결과는 그림 14(a)와 같다. GRSL기법은 모든 차원에서 지수함수 그래프를 그리며 실행시간이 급격히 증가한다. 또한 차원이 증가할수록 증가율이 높아져 대상객체가 많은 경우 뿐 아니라 다차원인 경우에서도 비효율적인 방법임을 알 수 있다.

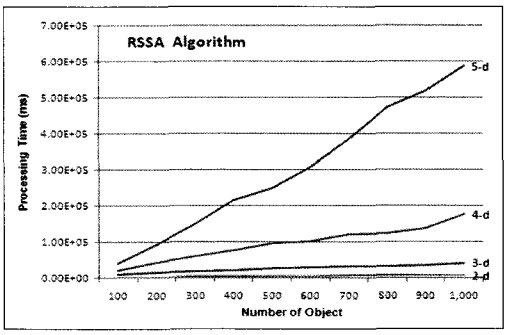
차원변화에 따른 RSSA기법의 실행시간 실험결과는 그림 14(b)와 같다. RSSA기법은 2,3차원에서는 큰 변화 없이 성능이 유지된다. 그러나 4차원에서는 저차원 실험결과와 달리 실행시간이 급격하게 증가하여, 5차원에서는 확연한 증가를 보인다. 즉, RRSA기법은 저차원에서는 성능이 비교적 유지되지만 차원이 증가할수록 비효율적인 방법임을 알 수 있다.

차원변화에 따른 ERSL기법의 실행시간 실험결과는 그림 14(c)와 같다. ERSL기법은 GRSL,RSSA기법과 달리 2차원에서 5차원까지 차원이 증가하여도 일정한 성능을 유지하는 것을 볼 수 있다. 그러므로 ERSL기법은 대상객체의 수가 많은 다차원 환경에서 가장 효율적인 방법임을 알 수 있다.

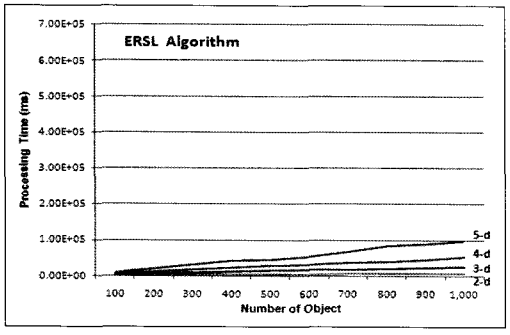
다차원 데이터를 고려한 경우 ERSL기법의 성능을 보다 확실히 보이기 위해 그림 14(d)에서는 대상객체가



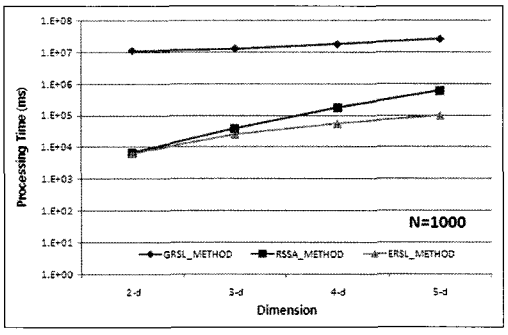
(a) GRSL기법의 다차원 객체에 따른 실행시간



(b) RSSA기법의 다차원 객체에 따른 실행시간



(c) ERSL기법의 다차원 객체에 따른 실행시간



(d) 차원의 변화에 따른 기법들의 실행시간 (객체=1000개)

그림 14 다차원 객체에 따른 실행시간

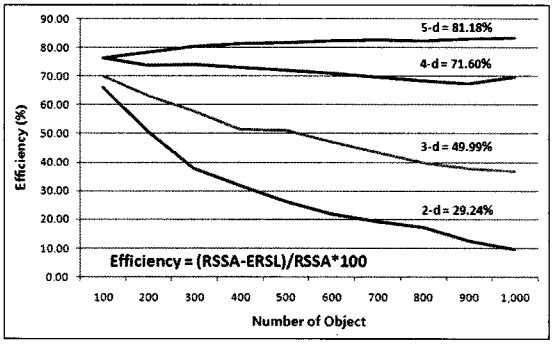


그림 15 다차원 객체에 따른 RSSA, ERSL기법의 효율성

1000개일 때 각 기법의 실행시간을 그래프로 표현하였다. 그림 14(d)를 보면, 차원이 증가할수록 ERSL기법이 GRSL, RSSA기법보다 실행시간이 적에 걸린다. 이러한 성향은 차원이 증가할수록 안정되어 일정한 성능의 차이가 유지될 것임을 예측 가능하다.

그림 15는 ERSL기법과 현재까지 가장 좋은 성능으로 증명된 RSSA기법과의 효율성을 비교한 그래프이다. 그림 15를 보면, 2차원의 경우 ERSL기법이 RSSA기법보다 29.24% 효율적인 반면에 5차원인 경우 81.18%로 효율성이 증가되었다. 뿐만 아니라 차원이 높아질수록 대상객체 수에 영향을 받지 않는 높은 효율성을 유지한다. 즉, ERSL기법은 다차원 데이터를 고려한 리버스 스카이라인 질의 처리에서 가장 뛰어나며, 고차원이 될수록 보다 뚜렷이 확인 가능하다.

5.3 메모리 성능 평가

모든 대상객체의 스카이라인을 메모리에 저장하여 유지하는 RSSA기법과 달리 ERSL기법은 객체를 색인하여 저장하는 R^*-tree와 힙 구성에 필요한 필수적인 메모리만을 사용한다. 즉, 적은 메모리의 사용으로 낭비가 발생하지 않는다. 그러므로 ERSL기법은 저장 공간이 작은 모바일 시스템에 적용하는 것이 적합하다. 그러나 질의시점에 모든 계산이 실행되므로 캐시메모리의 이용률이 높아 장치에 무리를 줄 수 있다. 보다 확실한 실험결과는 본 논문에서와 같이 메모리 기반이 아닌, 디스크 기반으로 구현하고 실험하여 증명하는 것이 좋다.

6. 결 론

정보 중심의 서비스가 새로운 이슈가 됨에 따라 이를 지원하는 리버스 스카이라인 기법에 대한 연수 또한 활발히 진행되고 있다. 그 중 VLDB(2007)에 제안된 RSSA 기법은 리버스 스카이라인 기법 중 성능이 인정된 가장 정형화된 방법이다. 그러나 실행시간과 메모리의 낭비가 반복적으로 발생하고, 처리과정이 복잡하여 여러 분야에 응용하는데 어려움이 있다.

본 논문에서 제안하는 ERSL기법은 RSSA기법의 문제가 되었던 객체들의 스카이라인 저장 및 윈도우 질의를 적용하지 않는 새로운 방법이다. 그러므로 RSSA기법의 단점인 실행시간과 메모리의 낭비를 절감할 뿐만 아니라, 객체의 추가 삭제로 인한 변화에 추가적인 처리과정이 필요 없어 방법이 간단하다. 또한 BBS기법을 바탕으로 진행하기 때문에 페이지 접근을 최소화하는 장점이 있다.

ERSL기법의 성능평가를 위해 대상객체 수의 변화와 차원의 변화에 따른 실행시간을 측정하는 모의실험을 수행하였다. 실험결과 대상객체 수가 증가하고 차원이 높아질수록 ERSL기법은 RSSA기법보다 효율적이다. 즉, 차원이 높아질수록 대상객체의 변화가 실행시간에 영향을 줄 확율은 RSSA기법이 64.65%, ERSL기법이 25.05%로써 ERSL기법이 보다 덜 민감하다. 또한 ERSL기법과 RSSA기법의 효율성을 비교해 볼 때 2차원의 경우 29.24%였던 효율성이 4차원의 경우 71.6%, 5차원인 경우 81.18%로 증가하여 다차원일수록 ERSL기법이 더 효율적임이 증명되었다.

ERSL기법은 기존의 리버스 스카이라인 처리 기법에 비해 시간적, 공간적으로 효율적인 기법이다. 그러나 이는 대상객체의 정적 속성값 만을 고려한 질의방법으로써 응용에 제한이 있다. 그러므로 향후 ERSL기법을 발전시켜 "나와 가장 가까운 거리에 있는 고객"과 같이 나의 위치(질의점)가 변화함에 따라서 계속적으로 변화하는, 동적속성을 고려한 리버스 스카이라인 연구도 진행할 예정이다. 또한 대상객체의 분포도에 따라 비슷한 조건을 가지는 객체 군을 묶어(Clustering) 질의를 수행하는 리버스 스카이라인 연구도 진행할 수 있다. 이러한 여러 가지 접근으로 향상된 리버스 스카이라인은 보다 넓은 분야에 적용되어 유용하게 사용될 것이다.

참 고 문 헌

[1] E. Dellis and B. Seeger, "Efficient Computation of Reverse Skyline Queries," *VLDB*, pp.291-301, 2007.

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

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

[4] J. Chomicki, "Preference Formulas in Relational Queries," *ACM TODS*, pp.427-466, 2003.

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

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

[7] J. Kim, Y. Park, "A progressive Skyline Region Decision Method," *Journal of KISS ; Database*, vol.34, no.6, pp.70-83, 2007. (in Korean)
[8] J. Kim, Y. Park, "Continuous Skyline Query Processing for Moving Objects in Variable Subspaces," 2008. (in Korean)
[9] Z. Li, Y. Park, "Efficient Processing using Static Validity Circle for Continuous Skyline Queries," *Journal of KISS ; Databases*, vol.33, no.6, pp.631-643, 2006. (in Korean)
[10] J. Kim, 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)
[11] A. Han, 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)

한 아

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

박 영 배

1993년 서울대학교 대학원 컴퓨터공학과(공학박사). 1981년~현재 명지대학교 컴퓨터공학과 교수. 관심분야는 Mobile DB, Spatial DB, 한국어 정보처리, 대용량 지문 DB, Skyline queries