직선 위에 중심을 갖는 k-센터를 계산하는 알고리즘

(Algorithms for Computing k-centers on the Line)

나 현 숙 † 신 찬 수 ††
(Hyeon-Suk Na) (Chan-Su Shin)

요 약 이차원 평면에 주어진 점들을 포함하면서 고정된 기울기를 갖는 직선 위에 중심을 갖는 k개의 디스크를 찾는 데, 최대 디스크의 반지름이 최소가 되는 디스크 집합을 계산하는 문제를 다룬다.
키워드 : 알고리즘, 계산기하학, 기하최적화, k-센터

Abstract This paper considers variants of k-center problems to find the set of disks with centers on a given (or moving) line with fixed orientation such that the disks contain a given point set and the maximum radius of the disks is minimized.
Key words : Algorithm, Computational Geometry, Geometry Optimization, k-Center Problem

1. 서 론

기관위치설정 문제(Facility Location Problem)는 계산기하학 뿐만 아니라 Operational Research 분야에서도 매우 중요한 문제로 다루어지고 있다. 가장 대표적인 문제가 k-센터 문제이다. 이차원 평면에 n개의 점이 주어질 때, 이 점을 포함하는 k개의 디스크를 구하는 데, 가장 큰 디스크의 반지름이 최소가 되도록 하는 문제이다.

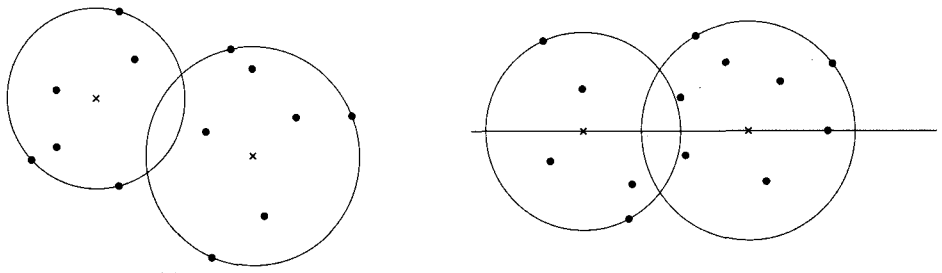
이 문제는 무선 네트워크 분야 또는 센서 네트워크 분야에서 응용될 수 있다. n개의 점이 무선통신 이용자(또는 센서)라 할 때, 무선통신을 위한 기지국 (또는 서버)을 k개 설치하는 문제와 같다. 기지국 또는 서버의 통신 범위는 설치 위치를 중심으로 한 디스크(disk) 형태로 표현되며, 디스크 안에 있는 점은 해당 기지국 또는 서버와 통신할 수 있다. 따라서 디스크의 반지름이 크면 그 만큼 많은 수의 점을 포함할 수 있다는 것을 의미하며, 동시에 높은 구매 비용이 요구된다. 결국, k개의 디스크 중에 가장 큰 디스크의 반지름을 최소로 하는 것이 k개의 서버 설치 비용을 줄이는 가장 중요한 조건이 된다.

현재까지 k-센터 문제에 대한 많은 연구가 이루어졌다. 만약 $k=1$이라면, n개의 점을 포함하는 가장 작은 디스크를 찾는 문제가 된다. 이 문제는 n개의 점에 대한 FVD(Farthest Voronoi Diagram)을 구하여 다이어그램의 정점이나 에지위에 중심이 놓이는 경우를 따져서 $O(n\log n)$시간에 최소 디스크를 계산할 수 있다. 그러나 보다 효율적인 prune-and-search 기법을 사용하면 선형시간인 $O(n)$시간에 계산할 수 있다[1]. 그리고 $k=2$인 경우는 두 개의 디스크로 모든 점을 포함하는 것이고, 두 디스크의 반지름 중 큰 값을 최소화하는 것이다. 그림 1의 (a) 참조. 이 문제에 대한 다양한 알고리즘이 발표되었다. $O(n^2\log^c n)$ 시간 알고리즘이 초기에 발표되었으나, 1996년에 $O(n\log^9 n)$ 시간에 해결할 수 있음이 알려졌다.[2] 이 알고리즘을 수정 보완한 T. M. Chan[3]에 의한 알고리즘은 $O(n\log^2 n\log^2\log n)$ 시간에 동작하는 현재까지 알려진 가장 빠른 알고리즘이다. 그러나 $k\geq 3$인 경우에는 효율적인 알고리즘이 알려져 있지 않다.

· 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음
· 본 연구는 학술진흥재단의 과제 KRF-2005-003-D00278의 지원을 받아 수행되었음

† 종신회원 : 숭실대학교 컴퓨터학부 교수
hsnaa@ssu.ac.kr
†† 종신회원 : 한국외국어대학교 디지털정보공학 교수
cssin@hufs.ac.kr
(Corresponding author임)
논문접수 : 2007년 3월 14일
심사완료 : 2007년 8월 22일

정보과학회논문지: 시스템 및 이론 제34권 제11호(2007.12)



(a) (b)

그림 1 (a) 일반적인 2-센터 문제. (b) 수평선 위에 중심이 있는 2-센터 문제

이 외에도 다양하게 변형된 k-센터 문제가 연구되었다. 센터가 입력 점으로 제한된 이산 k-센터 문제, 점 이외의 선분이나 원과 같은 다른 기하 객체를 포함하는 센터 문제, 고차원 센터 문제, 다른 거리 기준에서의 센터 문제, 군집화(clustering) 문제 등 매우 다양하다.

본 논문에서는 디스크의 중심이 한 고정된 기울기의 직선 위에 놓일 때의 k-센터 문제를 다룬다. 이 문제를 **직선 k-센터 문제**라 부른다. 즉 디스크 중심이 모두 주어진 직선 위에 놓이고 디스크의 합집합이 입력 점을 모두 포함하는 k개 디스크의 중심을 결정하는 데, 가장 큰 디스크의 반지름이 최소가 되도록 하는 문제이다. 그림 1의 (b) 참조. 대부분의 경우 직선의 방향을 하나로 고정한다. 본 논문에서는 수평선으로 제한한다. 이 직선(수평선) k-센터 문제는 다시 수평선이 고정된 경우와 움직이는 경우로 나뉜다. 움직이는 경우의 문제는 최대 디스크의 최소 반지름뿐 만 아니라 디스크의 중심이 놓이는 수평선을 구해야 한다. 본 논문에서는 두 가지 모두 다룬다.

이 문제는 무선통신을 위한 기지국 설치나 센서 네트워크 시스템에서 서버를 설치하는 응용분야에 활용될 수 있다. 예를 들어, 센서 네트워크 시스템에서는 작은 센서를 특정한 장소에 임의로 배치한 후, 센서가 수집한 다양한 종류의 데이타를 무선통신을 통해 서버에게 전달하여 후처리 과정을 수행한다. 센서는 자체 내장 전지에 의해 작동되지만, 서버는 외부 전원으로 동작한다. 따라서 k개 서버의 설치 장소는 외부 전원을 쉽게 사용할 수 있는 곳이어야 하며, 모든 센서가 k개의 서버 중에서 최소 하나 이상의 서버의 통신범위 안에 놓이는 곳이어야 한다. 외부 전원은 직선 형태의 전선으로 표현되므로, k개 서버의 최적 위치는 모두 직선위에 놓이며, k개의 서버 영역(디스크)의 합집합이 n개의 센서(점)를 포함해야 하고, 서버 영역의 최대 반지름이 최소가 되도록 하는 것이 경제적으로 바람직하다. 이 문제는 바로 직선 k-센터 문제와 같다.

이전 연구 결과 중에는 k개 디스크의 반지름의 합을 최소로 하는 문제에 대한 결과가 있다[4]. 고정된 수평선의 경우엔 동적계획법(dynamic programming) 기법을 적용하였으며 $O(n^2 \log n)$시간에 해결하였다. (이 문제에서는 k가 입력의 일부로 주어지지 않는데, 그 이유는 반지름의 합이 최소가 되는 최적 디스크 집합에서의 디스크 개수는 자동적으로 결정되기 때문이다.) 반면에 수평선이 움직이는 경우엔, 디스크의 반지름이 합이 최소가 되는 수평선을 찾아야 한다. 이 문제는 다항시간에 풀기는 매우 어렵다는 것이 [4]에서 증명되었으며, 다양한 근사 알고리즘(approximation algorithm)이 제시되었다. 이 결과를 제외한 다른 관련 결과는 알려져 있지 않다.

본 논문에서는 수평선 k-센터 문제를 $O(\min\{n(\log n)^{\lceil \log k \rceil}, n^2 \log n\})$-시간에 해결하는 알고리즘을 제시한다. (여기서 $k \geq 2$이다. 만약 $k=1$이라면 $O(n \log n)$ 시간에 계산할 수 있다.) 또한 움직이는 k-센터 문제에 대해서는 $O(n^{\min\{4,k\}} \log n)$, $(k \geq 2)$ 시간에 최적 수평선과 반지름을 찾는 알고리즘을 제시한다.

2. 고정된 수평선 위에 중심을 갖는 k-센터 문제

센터의 개수인 k, 디스크의 중심이 놓이는 직선, n개의 점 집합 S가 입력으로 주어진다. 고정된 직선을 x-축으로 대체할 수 있으므로, 설명의 편의상 디스크의 중심이 모두 x-축 위에 놓인다고 가정한다. 점 집합 $S=\{p_1, p_2, \cdots, p_n\}$의 각 점 p_i는 (x_i, y_i) 좌표로 표현된다. 여기서 점들의 x-좌표는 오름차순으로 정렬되어 있다고 가정한다. 만약 정렬되어 있지 않다면 $O(n \log n)$-시간 정렬 알고리즘을 이용하여 미리 정렬한다.

2.1 기하학적 성질 규명

점 집합 S에 대한 k-센터 문제의 최적 해(가장 큰 디스크의 최소 반지름)를 $R_k(S)$라 하자. 다음은 가장 간단한 성질 중 하나이다.

사실 1. 임의의 두 양의 정수 $k' < k$에 대해, $R_{k'}(S) \geq R_k(S)$이 성립하고, 임의의 두 점 집합 $S' \subseteq S$에 대해, $R_k(S') \leq R_k(S)$이 성립한다.

점들의 y-좌표 중 가장 큰 절대 값을 $Y(S)$라 하자. 가장 큰 절대 y-값을 갖는 점도 하나의 디스크에 포함되어야 하므로 그 디스크의 반지름은 $Y(S)$보다 같거나 크게 된다. 결국 디스크의 개수에 상관없이 언제나 $R_k(S) \geq Y(S)$이 성립한다. 또한 $R_n(S) = Y(S)$가 성립한다.

사실 2. 임의의 양의 정수 k에 대해, $R_k(S) \geq Y(S)$이 성립하고 $R_n(S) = Y(S)$이 성립한다.

이젠 $R_k(S) = Y(S)$를 만족하는 최소 k를 찾아보자. 이 최소값을 $k^\star$라 하자. 만약 $k \geq k^\star$면 위의 사실 1과 사실 2에 의해, $R_k(S) = Y(S)$가 성립하고 $k < k^\star$면 $R_k(S) > Y(S)$이 성립함을 쉽게 알 수 있다. 최소값 $k^\star$을 찾는 방법은 다음과 같다.

S의 각 점 p_i를 중심으로 반지름이 $Y(S)$인 디스크를 그린다. 이 디스크와 x-좌표 축과 만나는 구간을 J_i라 하자. 이 구간들의 집합을 $J = \{J_1, J_2, \cdots, J_n\}$라 하자. 이 구간은 모두 1차원 직선인 x-좌표 축 위에서 정의된다. 이제, x-축 위에서 디스크의 중심이 놓이는 몇 개의 점을 선택하려고 한다. 만약 반지름이 $Y(S)$인 디스크의 중심이 구간 J_i 위에 놓인다면, 그 디스크는 점 p_i를 포함하게 된다. 따라서 각 구간이 최소 하나 이상의 디스크 중심을 포함하면, S의 모든 점은 최소 하나 이상의 디스크에 포함되게 된다. 이 성질을 만족하는 x-축 위의 점(또는 핀)의 최소 개수를 최소 관통(stabbing) 수라 부른다. 이 최소 관통 수가 바로 $k^\star$이다.

최소 관통 수는 다음과 같은 그리디(greedy) 전략으로 $O(n\log n)$ 시간에 계산할 수 있다. 우선 구간 J_a가 구간 J_b를 완전히 포함하고 있다면, J_a를 구간 집합 J에서 삭제해도 최소 관통 수에는 변함이 없다. 왜냐하면, J_b를 관통한다면 J_a는 항상 관통할 수 있기 때문이다. 따라서 우리는 J에 속한 어떤 구간도 다른 구간을 완전히 포함하지 않는다고 가정할 수 있다. 다음 과정은 그림 2처럼 J의 구간들을 오른쪽 끝 점의 위치에 따라 오름차순으로 정렬한다. 가장 왼쪽에 오른쪽 끝 점이 놓인 구간을 선택해서 그 구간의 오른쪽 끝 점에 핀을 꽂는다. 이 핀에 의해 관통되는 모든 구간을 제거한 후에, 남은 구간들에 대해서 같은 방법을 적용한다. 이 그리디 방법이 항상 최소 관통 수와 핀의 위치를 결정함을 쉽게 증명할 수 있다. 하나의 핀의 위치를 결정할 때, 그리디 방법과 다르게 선택하는 것이 최소 관통 수를 보장한다면, 그 핀이 관통하는 구간들의 집합이 그리디 방법이 결정한 핀이 관통하는 구간들의 집합에 완전히 포함되기 때문에 그리디 방법보다 더 작은 관통 수를 보장할 수 없다. 따라서 그리디 방법은 항상 최소 관통 수를 계산한다. 수행시간은 정렬 후 선형 시간 탐색이면 충분하므로 $O(n\log n)$이 된다.

사실 3. $R_k(S) = Y(S)$를 만족하는 최소 $k^\star$의 값은 $O(n\log n)$ 시간에 계산할 수 있다.

이제 $k^\star$ 값을 알고 있으므로, 본 논문에서는 $k < k^\star$에 대해서만 고려한다. 또한 $n \leq k$이면 항상 $R_k(S) = Y(S)$이므로 $n > k$라고 가정한다. 점 집합 S를 포함(커버)하는 최적의 k개 디스크를 중심이 놓인 위치(x-좌표)에 따라 가장 왼쪽부터 오른쪽 방향으로 각각 $D_1, D_2, \cdots, D_k$라 하고, 디스크 D_i의 반지름을 $r(D_i)$이라 하자. 어떤 수직선 h의 왼쪽 반평면을 h^-, 오른쪽 반평면을 h^+라 하자. 그러면 아래와 같은 기본 성질이 성립한다.

사실 4. 임의의 $k < k^\star$에 대해서, 점 집합 S를 포함하는 디스크 집합 $D_1, D_2, \cdots, D_k$ 중에서 반지름의 가장 큰 디스크 D라 하고, 그 중심을 지나는 수직선을 h라 하자. 그러면 D의 경계에는 다른 디스크에 포함되지는 않는 최소 두 점이 존재하는데, 그 중 한 점은 h^-에 있으며, 다른 점은 h^+에 속한다.

증명. 우선 $k < k^\star$이므로 $R_k(S) > Y(S)$이 성립되어, D의 중심의 바로 위 또는 아래 경계 위에는 점이 놓이지 않는다. 따라서 이 경우는 제외한다. 만약, 최대 디스크 D가 위 성질을 만족하지 않는다고 가정하자. 우선 D의 경계 위에 한 점 p만 놓인다면, D를 p가 D 내부로 들어오고 원래 내부에 포함하던 점들은 계속 포함하도록 움직일 수 있다. 그러면 D 경계 위에는 어떤 점도 오지 않는다. 이것은 D의 반지름을 조금 줄일 수 있다는 것을 의미하여 D가 가장 큰 디스크라는 가정에 모순이다. 따라서 D의 경계 위에는 최소한 두 점 이상이 존재해야 한다. 이제, 두 점이 h의 한 쪽에만 존재한다

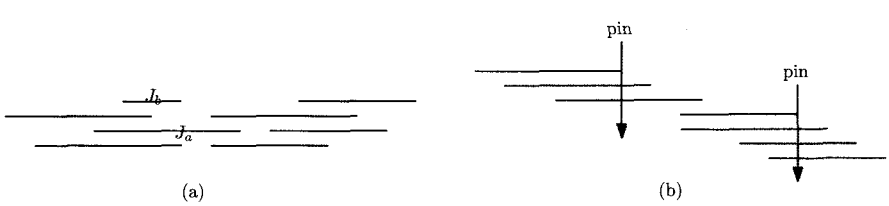


그림 2 (a) 정렬 전. (b) 정렬 후.

고 가정하자. 이 경우 역시 D를 경계위에 점들이 놓여 있는 방향으로 조금 움직이고 반지름을 조금 줄여도 원래 포함된 점을 그대로 포함하도록 할 수 있다. 따라서 h^-와 h^+에 최소 하나 이상의 점이 포함되어야 한다. 마지막으로 h^- (또는 h^+)에 속하며 D의 경계 위에 오는 점이 D 이외의 다른 디스크에도 포함된다고 가정하면 반대쪽 방향으로 D를 움직인 후 반지름을 줄일 수 있다. □

사실 5. 임의의 $2k \le k'$에 대해, $R_{k'}(S) < R_k(S)$이 성립한다.

증명. 최적 디스크 집합 $D_1, D_2, \cdots, D_k$에서, 반지름이 가장 큰 디스크 중 하나를 D라 하자. 사실 4에 의해, D의 경계 위에는 중심을 지나는 수직선 h의 서로 다른 쪽에 오는 점이 최소 하나 이상 존재하게 된다. 이제 D와 같은 크기의 새로운 디스크 D'을 다음과 같이 추가한다. 먼저 D'의 중심을 D의 중심과 일치시킨다. D를 움직이는데, D가 포함하는 h^-에 있는 점들은 그대로 포함하면서 왼쪽으로 조금 이동한다. 반대로 D'은 D가 포함하는 h^+에 있었던 점들을 그대로 포함하면서 오른쪽으로 조금 이동한다. 이 때 이동거리를 적절히 작게 하면, D와 D'의 경계 위에는 어떤 점도 오지 않으면서 원래 D가 포함했던 점들은 여전히 D와 D'에 포함됨을 알 수 있다. 그러면 두 디스크의 반지름을 조금 줄일 수 있고, 두 디스크는 더 이상 가장 크지 않다. 만약 $D_1, D_2, \cdots, D_k$ 중에 가장 큰 디스크가 하나 이상이라면, 같은 방법으로, 하나의 디스크를 추가하여 더 작은 디스크 두 개로 변경할 수 있다. k개 디스크가 모두 반지름이 가장 큰 경우라도 k개의 디스크만 추가하면 모두 더 작은 디스크로 만들 수 있다. 따라서 $R_{2k}(S) < R_k(S)$이 되어, $2k \le k'$에 대해서 $R_{k'}(S) < R_k(S)$이 성립한다. □

사실 6. 다음의 조건을 만족하는 점 집합 S에 대한 k개 디스크 $D_1, D_2, \cdots, D_k$ 집합이 항상 존재한다:

(1) $\max_i \{r(D_i)\} = R_k(S)$이고,

(2) 모든 i에 대해, 다른 디스크에는 포함되지 않고 D_i에만 포함되는 S의 한 점 이상이 존재한다.

증명. 어떤 D_i에 대해, D_i에만 포함된 점이 존재하지 않는다면, D_i를 빼도 남은 디스크들이 S를 여전히 포함한다. 이러한 디스크를 모두 빼고 남은 디스크 집합을 $B_1, B_2, \cdots, B_{k'}$라고 다시 표기한다. 사실 5에 의해 $k < 2k'$이어야 한다. 즉 $k - k' < k'$이 성립한다. 이젠 위의 두 조건을 만족하도록 k'개의 $B_1, B_2, \cdots, B_{k'}$에 $k-k'$개의 디스크를 추가할 수 있음을 보인다. $B_1, B_2, \cdots, B_{k'}$ 중에 가장 큰 디스크를 B라 하면, $r(B) = R_k(S)$이다 B의 경계 위에는 사실 4에 의해 중심을 지나는 수직선의 양 쪽에 다른 디스크에 포함되지 않는 한 점 이상이 각각 존재해야 한다. 그러면 사실 5의 증명에서와 같이, B와 같은 크기의 새로운 디스크 D를 추가하여 B가 원래 포함하고 있었던 점을 D와 나누어 포함하도록 하고, 두 디스크의 크기를 줄인다. 이 두 디스크는 조건 (2)를 만족함을 쉽게 알 수 있다. 이러한 과정을 $B_1, B_2, \cdots, B_{k'}$의 가장 큰 디스크에 대해 반복한다. 최대 디스크가 $k-k'$개 보다 많지 않다면, 우리는 $k-k'$개의 디스크를 추가하여 최대 디스크의 크기를 줄일 수 있으므로 더 작은 $R_k(S)$를 얻게 된다. 이것은 최적성 가정에 모순이므로 가장 큰 디스크의 개수는 $k-k'$개보다 많아야 한다. 결국 추가로 갖고 있는 $k-k'$개의 디스크를 조건 (2)가 만족되도록 추가로 배치하더라도 $B_1, B_2, \cdots, B_{k'}$ 중에 최대 디스크는 여전히 하나 이상 남아 있다. 최종적으로 k개의 디스크로 S를 포함하고 최대 디스크의 크기는 $R_k(S)$와 같다. □

위 사실 6에 의해 우리는 항상 두 조건을 만족하는 k개의 디스크 집합 $D_1, D_2, \cdots, D_k$을 고려하기만 하면 된다. 이러한 디스크 집합을 S에 대한 최적 디스크 집합이라 부른다. 두 디스크 D_i와 D_j의 경계가 두 점에서 교차한다면, 두 교차점을 연결한 수직선을 $\ell(i,j)$라 하자. 디스크 D_i에 대해, D_i의 오른쪽 위치한 디스크 D_j $(j > i)$와의 수직선 $\ell(i,j)$ 중 가장 왼쪽에 위치한 수직선을 h_i라고 정의한다. 만약, D_i가 어떠한 D_j와도 교차하지 않는다면 D_i와 접하는 두 개의 수직선 중에서 오른쪽에서 접하는 수직선으로 정의한다(그림 3 참조).

사실 7. 최적 디스크 집합 $D_1, D_2, \cdots, D_k$와 $k_1 + k_2 = k$를 만족하는 임의의 두 양의 정수 k_1, k_2에 대해, $S = S_1 \cup S_2$, $S_1 \cap S_2 = \varnothing$가 성립하고 S_1의 점들의 x-좌표가 S_2의 점들의 x-좌표보다 작으며, $R_k(S) = \max\{R_{k_1}(S_1), R_{k_2}(S_2)\}$이 성립하는 S의 부분집합 S_1, S_2이 존재한다.

증명. D_{k_1}에 대한 h_{k_1}의 왼쪽에 속하는 점들의 집합을 S_1으로 정의하고, 오른쪽에 속하는 점들의 집합을 S_2라 정의한다. 이제 S_1의 모든 점은 $D_1, \cdots, D_{k_1}$의 디스크에 포함됨을 증명하자. 만약 S_1의 어떤 점 p가 $D_1, \cdots, D_{k_1}$의 어떤 디스크에도 속하지 않는다고 가정하자. 그러면 어떤 D_j $(j > k_1)$에 속해야 한다. 디스크 D_j는 D_{k_1}를 완전히 포함할 수 없기 때문에, D_{k_1}과 교차해야 한다. 그런데, 교차점을 연결한 수직선 $\ell(k_1, j)$는 h_{k_1}의 정의에 의해 h_{k_1}의 왼쪽에 오지 못하므로, p를 포함되는 영역

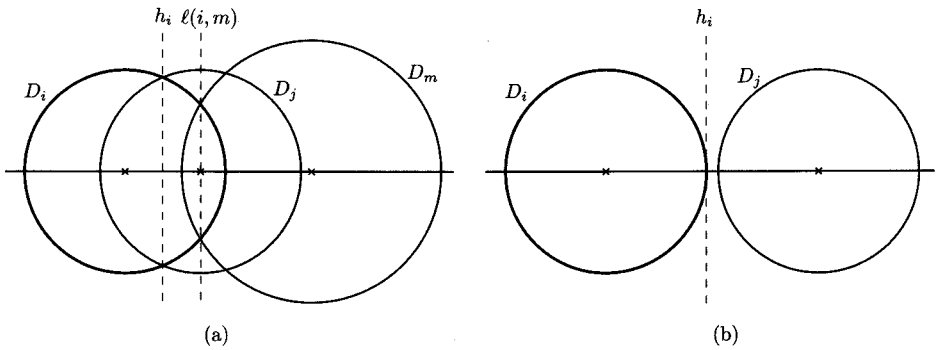


그림 3 h_i의 정의

$D_j \cap h_{k_1}^-$이 D_{k_1}에 완전히 포함되어야 한다. 이것은 p가 D_{k_1}에 포함됨을 의미하는데, 이는 $D_1,\cdots,D_{k_1}$ 중의 어떤 디스크도 p를 포함하지 않는다는 가정에 모순된다. 따라서 S_1의 모든 점은 $D_1,\cdots,D_{k_1}$ 중의 하나에 반드시 포함된다. 이것은 $R_{k_1}(S)$와 $R_{k_2}(S)$의 정의에 의해, $R_{k_1}(S_1) \le \max_{i \le k_1}\{r(D_i)\}$이고 $R_{k_2}(S_2) \le \max_{k_1 < i}\{r(D_i)\}$이 성립한다. 따라서, $\max\{R_{k_1}(S_1), R_{k_2}(S_2)\} \le \max_i\{r(D_i)\} = R_k(S)$이다. $R_k(S)$의 정의에 의해, $\max\{R_{k_1}(S_1), R_{k_2}(S_2)\} \ge R_k(S)$이므로, $R_k(S) = \max\{R_{k_1}(S_1), R_{k_2}(S_2)\}$이 성립한다. □

x-좌표에 대해 오름차순으로 정렬되어 있는 점 집합 $S=\{p_1, p_2, \cdots, p_n\}$의 부분집합 $S(i)$를 첫 i개의 점들로 구성된 집합, 즉 $S(i)=\{p_1, p_2, \cdots, p_i\}$으로 정의한다. 그러면 위의 사실 7은 $R_k(S) = \min_i \max\{R_{k_1}(S(i)), R_{k_2}(S-S(i))\}$을 의미하고, 이를 만족하는 인덱스 i를 찾으면 된다. 이 인덱스를 특별히 $i^\star$라 표기하자. 다음의 자명한 사실은 $i^\star$를 이진탐색으로 찾을 수 있음을 알려준다.

사실 8. $R_k(S(i))$는 i에 대해 감소하지 않는 함수(non-decreasing function)이다.

2.2 알고리즘

$k_1+k_2=k$를 만족하는 임의의 두 자연수 k_1, k_2를 정한다. 각 i에 대해, $S_1 := S(i)$로 하고 $S_2 := S-S(i)$로 정의하고 이를 점진적 계산법으로 계산한다. 이젠, 사실 7에 의해, $R_k(S) = \min_i \max\{R_{k_1}(S(i)), R_{k_2}(S-S(i))\}$이 되고, 이를 만족하는 $i=i^\star$를 찾으면 된다. 그런데 사실 8에 의하면, $R_{k_1}(S(i))$는 i에 대해 증가(non-decreasing)함수이고, $R_{k_2}(S-S(i))$는 감소(non-increasing) 함수가 되어, $i^\star$를 이진탐색으로 찾을 수 있게 도와준다. 현재 이진탐색 단계에서 $S(i)$를 고려할 때, 만약 $R_{k_1}(S(i)) < R_{k_2}(S-S(i))$이라면 $i^\star$는 i보다 크거나 같아야 되며, $R_{k_1}(S(i)) > R_{k_2}(S-S(i))$이라면 i보다 작거나 같아야 한다. 만약 $R_{k_1}(S(i)) = R_{k_2}(S-S(i))$면 $i^\star = i$가 되어 탐색이 끝난다. 탐색이 끝나지 않았다면, 다음 단계에서는 $i^\star$가 존재하는 범위의 중간 인덱스를 i로 정하여 이진탐색을 계속한다. 탐색이 계속되어 탐색 구간이 최종적으로 $[i, i+1]$으로 좁혀졌다면 $i^\star$는 i와 $i+1$ 둘 중 하나가 된다. 또한, $R_{k_1}(S(i)) < R_{k_2}(S-S(i))$이고 $R_{k_1}(S(i+1)) > R_{k_2}(S-S(i+1))$이 성립한다. (그렇지 않다면 함수의 증가, 감소 성질에 의해, 이진탐색이 진행되는 동안 $R_{k_1}(S(i)) = R_{k_2}(S-S(i))$이 성립되는 i가 존재하여 $i^\star$가 결정되었을 것이다.) 결국 $R_{k_2}(S-S(i)) > R_{k_1}(S(i+1))$이면 $i^\star = i+1$이 되고, 반대라면 $i^\star = i$가 된다.

위와 같은 이진탐색을 위해선 $k_1, k_2 < k$에 대해 $R_{k_1}(S)$, $R_{k_2}(S)$를 계산할 수 있어야 한다. 이것은 $k=1$에 대한 알고리즘부터 설계하고, 이를 이용하여 $k>1$에 대한 알고리즘을 재귀적으로 구성하는 방법으로 계산할 수 있다.

2.2.1 $k=1$인 경우

먼저 $k=1$인 경우를 살펴보자. 이 경우에는 모든 점을 포함하는 최소 반지름의 디스크를 계산해야 한다. 최소 디스크의 중심은 S에 대한 $FVD(S)$의 에지와 x-축과의 교차점 중의 하나에 놓이며, 최소 디스크의 반지름 $R_1(S)$는 그 교차점을 정의한 S의 두 점까지의 거리가 됨을 쉽게 알 수 있다. 따라서 $FVD(S)$와 x-축과의 교차구간을 계산하면, 선형시간에 $R_1(S)$을 계산할 수 있다. 교차구간을 계산하는 가장 단순한 방법은 $FVD(S)$를 직접 계산한 후에 x-축과의 교차구간을 구하는 것이다. 그러나 $FVD(S)$를 구하는 알고리즘이 단순하지 않

기 때문에, $FVD(S)$를 명시적으로 계산하지 않고 x-축과의 교차구간을 계산하는 방법을 제시한다.

우리는 이미 $O(n\log n)$ 시간에 S의 점들을 x-좌표에 따라 오름차순으로 정렬해 놓았다. 그러면 $S(1)$부터 시작하여 점진적 계산법에 의해, $S(1), S(2), \cdots, S(n)$에 대한 교차구간을 차례대로 선형시간에 계산할 수 있다. $FVD(S(i))$와 x-축과의 교차구간이 이미 계산되었다고 가정하고, 교차구간의 끝 점들을 왼쪽부터 차례로 $x_1, x_2, \cdots, x_{m_i}$이라 하자. 편의상 $x_0 = -\infty, x_{m_i+1} = +\infty$라고 하자. j번째 교차구간은 $[x_j, x_{j+1}]$로 정의되고, 그 구간에 대해 가장 먼 점을 $p_{\sigma(j)}$라 하자. 그러면 교차점 x_{j+1}은 두 점 $p_{\sigma(j)}, p_{\sigma(j+1)}$의 수직이등분선 $b(p_{\sigma(j)}, p_{\sigma(j+1)})$과 x-축과의 교점으로 정의된다.

이제 $S(i)$에 점 p_{i+1}를 포함시킨 $S(i+1)$에 대한 $FVD(S(i+1))$의 교차구간을 어떻게 계산하는지 살펴보자.

점 p_i는 $S(i)$에 있는 점 중에서 x-좌표가 가장 크기 때문에, 가장 왼쪽의 교차구간인 $[x_0, x_1]$에서 가장 먼 점이 된다. 즉 $p_{\sigma(0)} = p_i$이다. 같은 이유로 $FVD(S(i+1))$와의 교차구간 중 가장 왼쪽 구간은 p_{i+1}의 구간임을 쉽게 알 수 있다. 그 구간을 $[x'_0, x'_1]$이라 하자. 교차구간의 왼쪽 끝 점은 $x'_0 = -\infty$이므로 오른쪽 끝 점 x_1'만을 결정하면 된다. 이를 위해, $FVD(S(i))$와의 가장 왼쪽 교차구간부터 차례대로 다음과 같이 살펴본다. $j=0$부터 시작한다.

(1) $b(p_{\sigma(j)}, p_{i+1})$과 x-축과의 교차점 z를 계산한다.

(2) z가 $[x_j, x_{j+1}]$에 포함된다면, $[x_j, z]$ 구간이 p_{i+1}의 구간에 포함되고, $[z, x_{j+1}]$은 $p_{\sigma(j)}$를 위한 구간이 된다. 즉 $x'_1 = z$가 된다. p_{i+1}에 대한 구간을 얻었으므로 $FVD(S(i+1))$에 대한 교차구간이 계산이 완성되었다.

(3) z가 $[x_j, x_{j+1}]$에 포함되지 않는다면 $p_{\sigma(j)}$에 대한 구간은 $FVD(S(i+1))$에 더 이상 존재하지 않게 된다. 따라서 다음 교차구간 $[x_{j+1}, x_{j+2}]$에 대해 (1)번 단계부터 반복한다. (즉 j를 하나 증가시키고 (1)번 단계부터 반복한다.)

이 과정에서 z가 현재 검사하는 교차구간에 포함되지 않는다고 하면, 그 구간은 $FVD(S(j))$ (모든 $j>i$)에 대한 교차구간에 나타나지 않게 된다. 즉 한 번 교차구간에서 제외되면 다시는 등장하지 않게 된다. 따라서 점진적으로 교차구간을 계산하는 알고리즘은 총 $O(n)$ 시간에 동작한다. S의 점을 x-좌표에 따라 정렬하는 시간을 합하면 $O(n\log n)$ 시간이면 충분하다.

소정리 1. S에 대한 최적 디스크와 $R_1(S)$는 $O(n\log n)$ 시간에 계산가능하다.

2.2.2 $k=2$인 경우

이젠 $k=2$인 경우를 생각해보자. 사실 7과 사실 8을 이용하는데, $k_1=1, k_2=1$이라 정의하고, $R_2(S) = \min_i \max R_2(S) = \min_i \max\{R_1(S(i)), R_1(S-S(i))\}$을 만족하는 $i=i^*$를 이진탐색을 이용하여 찾는다. 이진탐색을 시작하기 전에 $k=1$일 때 사용한 방법을 이용하여, 모든 i에 대해 $R_1(S(i))$와 $R_1(S-S(i))$를 $O(n\log n)$ 시간에 계산한다. 모든 i에 대한 $R_1(S(i))$를 얻기 위해선 $S(1), S(2), \cdots, S(n)$의 순서로 p_1부터 p_n까지 점을 하나씩 차례대로 추가하는 점진적인 방법으로 계산하고, $R_1(S-S(i))$를 위해선 $S-S(n-1), \cdots, S-S(1)$의 순서로 역순으로 p_n부터 p_1까지 점을 하나씩 추가하면서 계산하면 된다.

이제, $i=n/2$으로 하여 이미 계산되어 있는 $R_1(S(i))$와 $R_1(S-S(i))$를 비교하여 재귀적 이진탐색을 수행한다. 이진탐색은 $O(\log n)$번 수행되고, 매 탐색마다 이미 계산된 $R_1(S(i))$와 $R_1(S-S(i))$를 참조하므로 상수시간이면 충분하다. 따라서 이진탐색 단계는 $O(\log n)$ 시간이 소요된다. 따라서 전체 수행시간은 $O(n\log n)$이 된다.

소정리 2. S에 대한 최적 디스크 집합과 $R_2(S)$는 $O(n\log n)$ 시간에 계산가능하다.

2.2.3 $k>2$인 경우

$k>2$에 대해서도 이진탐색을 그대로 적용한다. 모든 $k'<k$에 대해, $R_{k'}(S)$를 $T(n,k')$에 계산할 수 있다고 가정한다. 우리는 이미 $k=1,2$인 경우는 모두 $O(n\log n)$ 시간에 계산할 수 있음을 증명했다. 이제 k에 대한 $R_k(S)$를 계산하는 데 필요한 시간을 알아보자. 이를 위해, $k_1 = \lfloor k/2 \rfloor$, $k_2 = \lceil k/2 \rceil$로 한다. 이진탐색으로 $R_k(S) = \min_i \max\{R_{k_1}(S(i)), R_{k_2}(S-S(i))\}$이 되는 $i=i^*$를 찾는다. 우리는 모든 $k'<k$에 대해, $R_{k'}(S)$를 $T(n,k')$에 계산할 수 있다고 가정하면, 한 번의 이진탐색 단계는

$$\begin{aligned} &T(S(i), \lfloor k/2 \rfloor) + T(S-S(i), \lceil k/2 \rceil) + O(1) \\ &= T(i, \lfloor k/2 \rfloor) + T(n-i, \lceil k/2 \rceil) + O(1) \\ &\le T(n, \lceil k/2 \rceil) + O(1) \end{aligned}$$

시간이 소요된다. 여기서 함수 $T(X,y)$는 $|X|$와 y에 관한 증가함수이고 $T(X,y) \ge |X|$이므로, 공통 원소가 없는 두 집합 X_1과 X_2에 대해서, $T(X_1,y) + T(X_2,y) \le T(X_1 \cup X_2, y)$가 성립한다. 따라서 위의 식에 나타난 부등식이 성립한다. 이를 정리하면 아래 재귀식과 같다.

$$T(n,k) \le \log n (T(n, \lceil k/2 \rceil) + O(1)).$$

이 식을 풀어보자. 우선 $k=2$인 경우는 소정리 2에 의해 $O(n\log n)$ 시간에 계산가능하다. $k=3$인 경우는 $k_1=1, k_2=2$로 정해서 이진탐색을 수행한다. 매번 이진탐색을 할 때마다 1-센터와 2-센터 알고리즘을 각각 호출하게 된다. 따라서 매번 $O(n\log n)$ 시간이 필요하게 되고, 총 $O(\log n)\times O(n\log n)=O(n\log^2 n)$ 시간이 소요된다. $k=4$인 경우에도 역시 $k_1=2, k_2=2$가 되므로, 한 번의 이진탐색을 위해 $O(n\log n)$ 시간이 필요하게 되어, 전체적으로 $O(n\log^2 n)$ 시간이 소요된다. 이 방법을 반복하면, $k=3,4,\cdots 8$인 경우에는 모두 $O(n\log^3 n)$이 되며, $k=9,10,\cdots,15$인 경우에는 $O(n\log^4 n)$이 된다. 따라서 위의 재귀식을 풀어 일반해를 구하면, 아래와 같은 결과를 얻을 수 있음을 쉽게 확인할 수 있다.

$$T(n,k)=n(\log n)^{\lceil \log k \rceil}$$

이 된다.

사실 4에 의해, 최적 디스크 집합에서 가장 큰 디스크의 경계에는 반드시 두 점 이상이 위치한다. 만약 하나의 점 쌍 (p_i,p_j)이 정해지면, 두 점을 경계에 포함한 디스크의 반지름 $r(p_i,p_j)$는 쉽게 계산된다. 우리는 모든 점 쌍 (p_i,p_j)에 대해, 반지름 $r(p_i,p_j)$을 계산한다. 이 반지름 중에서 $R_k(S)$가 존재한다. 총 $O(n^2)$개의 반지름을 오름차순으로 정렬한 후, 이진탐색을 수행하여 $R_k(S)$를 찾는다. 각 이진탐색에서는 주어진 반지름을 갖는 k개의 디스크로 점들을 모두 포함할 수 있는지의 여부만 판별하면 된다. 이것은 앞에서 설명한 그리디 알고리즘으로 $O(n\log n)$시간에 알 수 있다. 따라서 총 $O(n^2\log n)$ 시간에 k 값에 관계없이 $R_k(S)$를 계산할 수 있다.

정리 1. 임의의 $k\geq 2$, 디스크 중심이 위치할 수평선이 주어질 때, 점 집합 S에 대한 최적 디스크 집합과 $R_k(S)$는 $O(\min\{n(\log n)^{\lceil \log k \rceil}, n^2\log n\})$ 시간에 계산할 수 있다. $k=1$인 경우엔 $O(n\log n)$ 시간에 계산할 수 있다.

3. 움직이는 수평선 위에 중심을 갖는 k-센터 문제

이제는 디스크의 중심이 고정된 수평선 위에 있는 것이 아니라, 움직이는 수평선 위에 있을 때, 최적의 디스크 집합을 찾는 문제를 고려하자. 정확한 문제 정의는, 점 집합 S를 포함하면서 디스크 중심이 모두 특정 수평선 ℓ위에 있는 k개의 디스크 집합 $D_1,D_2,\cdots,D_k$을 찾는 것이다. 즉, 모든 수평선 ℓ 중에서 $\max_i\{r_i(D_i)\}$이 최소가 되는 **최적 수평선** ℓ^*를 찾고 ℓ^*위에 중심을 갖는 디스크의 최소 반지름 $R_k(S)$을 찾는 것이 목표다.

수평선 $\ell(y):=y$가 고정되어 있다면, 2절에서 설명한 최적 디스크 집합에 관한 다양한 성질이 그대로 성립한다. 이 외에도 몇 가지의 기하학적 성질들이 성립한다.

3.1 기하학적 성질 규명

고정된 수평선 $\ell(y)$에 대한 최적 디스크 집합을 $D_1^y,D_2^y,\cdots,D_k^y$라 하자. 이 중 가장 큰 디스크의 중심을 $c(y)=(x(y),y)$라 하고, 반지름을 $R_k^y(S)$로 하자. 그러면 $R_k(S):=\min_y R_k^y(S)$로 정의된다. $\ell(y)$가 움직이면, 즉 수평선의 y-좌표가 변하면, $R_k^y(S)$와 $x(y)$의 값도 따라서 변하게 된다. 즉, $R_k^y(S)$와 $x(y)$ 모두 y에 관한 함수이다.

우선 $k=1$인 경우를 살펴보자. S의 모든 점을 포함하는 가장 작은 디스크 하나를 찾는 문제이다. 디스크의 개수가 하나이므로 디스크 중심은 이차원 평면 위의 어느 점이라도 상관없다. 따라서 일반적인 1-센터 문제와 동일하다. 최적 디스크의 중심은 $FVD(S)$의 정점 (또는 에지 위의 한 점)에 놓이므로, $FVD(S)$의 에지와 정점을 방문하면서 $O(n\log n)$시간에 최적 디스크의 중심과 반지름을 찾을 수 있다. (그러나 [1]에서 제시한 prune-and-search 기법을 이용하면 $O(n)$ 시간에 최적 반지름을 찾을 수 있다.) 최적 디스크의 중심이 놓이는 $FVD(S)$의 정점 또는 에지위의 한 점을 z라 표기하면, $FVD(S)$는 z를 루트로 하는 트리(tree) 구조로 정의된다. z에서 출발하여 $FVD(S)$의 에지 위를 이동하면서 S를 포함하는 디스크를 그리면, 디스크의 반지름이 z에서 멀어질수록 지속적으로 커진다는 사실을 쉽게 확인할 수 있다. 이를 포함한 $FVD(S)$의 기본적인 기하학적 성질들은 [5]에 설명되어 있다.

사실 9. 수평선 $\ell(y)$에 대해, $\ell(y)$에 중심이 있으면서 S를 포함하는 최소 디스크의 중심 $c(y)$는 유일하다. 즉, 최소 디스크는 유일하다.

증명. 만약 유일하지 않다면, 즉 반지름이 같은 두 디스크 D와 D'이 모두 S를 포함한다고 한다면 S의 모든 점은 D와 D'의 공통영역(lune)에 포함되어야 한다. 그런데 이 공통영역을 포함하는 더 작은 디스크 D''이 항상 존재하므로 D와 D'이 가장 작은 반지름을 갖는다는 가정에 위배된다. □

특정 $\ell(y)$에 대한 $R_1^y(S)$는 $\ell(y)$와 교차하는 $FVD(S)$의 에지 e에 대해, $R_1^y(S)=\min_e\{r^y(e)\}$으로 정의되는데, 위의 사실 9를 이용하면 다음의 사실도 증명할 수 있다.

사실 10. 수평선 $\ell(y)$이 $y=-\infty$에서부터 $y=+\infty$까

지 이동할 때, $\ell(y)$에 중심이 있으면서 S를 포함하는 최소 디스크의 중심 $c(y)$의 자취는 트리 $FVD(S)$에서 z를 통과하는 연결된(connected) 경로(path)가 된다.

증명. 두 수평선 $\ell(y_1)$와 $\ell(y_2)$을 고려하자. 여기서 y_2은 y_1보다 약간 큰 값으로 가정한다. 이 두 수평선에 모두 교차하는 $FVD(S)$의 에지 중에서 두 에지를 e_1과 e_2라 하고, 이 중 하나의 에지 위에 구간 $y_1 \le y \le y_2$에서의 최소 디스크의 중심이 놓인다고 가정하자. 이제, $y_1 \le y \le y_2$에 대해, $r^y(e_1)$을 $\ell(y) \cap e_1$위에 중심을 두고 S를 포함하는 디스크의 반지름으로 정의한다. 그러면 $r^y(e_1)$은 상수 최고 차수를 갖는 다항식임을 쉽게 알 수 있다. 그 이유는 다음과 같다. 에지 e_1을 포함하는 직선은 S의 특정 두 점 p, q를 연결하는 선분의 수직이등분선으로 $y=mx+b$ 형식의 1차 함수이고, 함수 $r^y(e_1)$은 직선 위의 점 (x,y)와 p (또는 q 점)까지의 거리로 $r^y(e_1) := \sqrt{(x-p_x)^2+(y-p_y)^2} = \sqrt{(x-p_x)^2+((mx+b)-p_y)^2}$이다. 이 함수를 편의상 제곱하여 사용해도 되기 때문에, 거리 함수는 최고 차수가 2인 다항식으로 가정할 수 있다.

또한 두 함수 $r^y(e_1)$와 $r^y(e_2)$는 $y_1 \le y \le y_2$ 범위에서 서로 만나지 않는다. 만약 만난다면, 즉 $r^y(e_1)=r^y(e_2)$가 만족하는 특정 y가 존재한다면, $\ell(y)$에서 최소 디스크가 두 개가 되어 사실 9에 위배된다. 두 함수가 만나지 않는다면 하나의 함수가 다른 함수의 아래에 존재한다는 의미이다. 즉 $y_1 \le y \le y_2$의 모든 범위에서 $r^y(e_1) < r^y(e_2)$이거나 $r^y(e_1) > r^y(e_2)$이다. 이것은 $y_1 \le y \le y_2$ 범위에선 언제나 $R_1^y(S) = r^y(e_1)$이거나 $R_1^y(S) = r^y(e_2)$라는 것이므로, $c(y)$의 자취가 하나의 에지 위에서 연결된다는 것을 의미한다. 이 논리를 모든 y에 대해 확장하면, $c(y)$의 자취는 트리 $FVD(S)$에서 연결됨을 보일 수 있다. 마지막으로, 이 연결된 자취가 z를 통과한다는 사실과 $FVD(S)$가 트리라는 사실로부터 $c(y)$의 자취가 경로가 됨을 쉽게 확인할 수 있다. □

사실 10이 의미하는 것은 결국 $\ell(y)$이 $y=-\infty$에서부터 $y=+\infty$까지 이동할 때, 함수 $R_1^y(S)$는 해당 경로에 포함된 에지를 따라 정의된 곡선 조각들로 구성되었음을 나타내고, 그 값이 감소하다가 z의 y-좌표에서 최소가 되며 그 이후에 계속 증가하는 성질을 갖는다는 것이다. 이렇게 감소와 증가를 최대 한번 씩만 하는 함수를 strict uni-modal하다고 정의한다. 따라서 다음의 소정리가 성립한다.

소정리 3. $R_1^y(S)$는 $O(n)$개의 상수 차수의 곡선으로 이루어졌으며 strict uni-modal 함수이다.

3.2 최적 수평선 $\ell(y^\star)$ 및 $R_k(S)$를 찾는 알고리즘

3.2.1 $k=2$인 경우

최적 수평선 $\ell(y^\star)$을 고려하자. 이때의 최적 디스크 D_1과 D_2는 2절에서 설명한 기하학적 성질을 모두 만족한다. 즉 $R_2^{y^\star}(S) = \max\left\{R_1^{y^\star}(S(i)), R_1^{y^\star}(S-S(i))\right\}$를 만족하는 S의 두 부분집합 $S(i), S-S(i)$이 존재해야 한다. 이러한 분할은 최대 $n-1$가지가 존재한다. 따라서 $R_2^{y^\star}(S) = \min_i \min_y \max\left\{R_1^y(S(i)), R_1^y(S-S(i))\right\}$이 성립한다. 이젠 특정 분할 $S(i), S-S(i)$에 대해, $\min_y \max\left\{R_1^y(S(i)), R_1^y(S-S(i))\right\}$ 값을 구해보자.

우선, $FVD(S(i))$와 $FVD(S-S(i))$를 $O(n\log n)$ 시간에 계산한다. 다음으로, 함수 $R_1^y(S(i))$와 $R_1^y(S-S(i))$를 계산한 후, 두 함수의 $\max\left\{R_1^y(S(i)), R_1^y(S-S(i))\right\}$를 직접 계산한다. 소정리 3에 의해 $R_1^y(S(i))$와 $R_1^y(S-S(i))$이 모두 strict modal 함수이므로 $f_i(y) = \max\left\{R_1^y(S(i)), R_1^y(S-S(i))\right\}$도 $O(n)$개의 곡선으로 구성된 strict uni-modal 함수이다. 따라서 이 과정은 선형시간에 이루어진다. 또한 $f_i(y)$를 구성하면서 $f_i(y)$의 최소값을 동시에 찾을 수 있다. 결국, $\min_y \max\left\{R_1^y(S(i)), R_1^y(S-S(i))\right\}$를 $O(n\log n)$ 시간에 계산할 수 있다. 모든 분할을 고려하면 다음 정리를 얻을 수 있다.

소정리 4. $k=2$인 경우 $\ell(y^\star)$와 $R_2^{y^\star}(S)$는 $O(n^2\log n)$ 시간에 계산할 수 있다.

3.2.2 $k=3$인 경우

$k=2$인 경우의 알고리즘과 거의 같다. 다른 점은 세 개의 디스크를 고려하기 때문에 S를 세 개의 부분집합 S_1, S_2, S_3로 분할해야 한다는 점이다. 이러한 분할의 가짓수는 $O(n^2)$이다. 각 분할 (S_1, S_2, S_3)에 대해, FVD를 먼저 계산하고, strict unimodal 함수 $R_1^y(S_1)$, $R_1^y(S_2)$, $R_1^y(S_3)$를 계산한 후, $f(S_1,S_2,S_3) = \min_y \max\left\{R_1^y(S_1), R_1^y(S_2), R_1^y(S_3)\right\}$를 계산한다. $f(S_1,S_2,S_3)$ 또한 $O(n)$개의 곡선으로 구성된 strict unimodal 함수이므로 $f(S_1,S_2,S_3)$의 최소 값 역시 선형시간에 계산가능하다. 따라서 다음의 소정리가 성립한다.

소정리 4. $k=3$인 경우 $\ell(y^\star)$와 $R_3^{y^\star}(S)$는 $O(n^3\log n)$ 시간에 계산할 수 있다.

3.2.3 $k \ge 4$인 경우

위의 $k=2,3$인 경우와 같은 방식으로 문제를 풀면 $O(n^k\log n)$ 시간에 해결할 수 있다. 이 보다 빠른 k에 종속되지 않는 $O(n^4\log n)$ 시간에 동작하는 알고리즘을 제시한다.

최적 수평선 $\ell(y^\star)$위의 중심을 갖는 최적 디스크 집합을 고려하자. 가장 큰 디스크를 D라 하자. 당연히 D의 경계 위에는 최소 두 점이 놓여야 하고, 왼쪽 반 원위에 한 점과 오른쪽 반 원위에 한 점씩 놓인다.

만약 D가 독자적으로 $\ell(y^\star)$와 $R_k^{y^\star}(S)$를 결정한다면 두 가지 경우만 존재한다. (1) D의 지름을 결정하는 두 점이 D의 경계 위에 놓이는 경우와 (2) 세 점이 D의 경계 위에 놓이는 경우이다. 두 번째 경우에는 세 점이 이루는 삼각형이 반드시 D의 중심을 포함해야 한다. 그렇지 않으면 직선을 조금 움직여서 디스크 반지름을 조금 줄일 수 있게 된다. 그러나 위의 두 경우와 달리 D가 단독으로 $\ell(y^\star)$와 $R_k^{y^\star}(S)$를 결정하지 못하고 두 번째로 큰 디스크를 D'과 함께 결정하는 경우도 존재한다. 그림 4(a)에서처럼, 두 디스크의 경계 위에 각각 최소 두 점 이상이 놓이는 경우이다. 이 경우에는 반드시 D 또는 D'의 경계에는 최소 한 점 이상이, $\ell(y^\star)$의 아래쪽에 있는 D 또는 D'의 경계에도 한 점 이상이 존재해야 한다. 그렇지 않으면 직선을 이동하여 두 디스크의 반지름을 줄일 수 있게 된다. 여기서 쉽게 확인할 수 있는 사실은 이럴 경우 두 디스크의 반지름이 서로 같아야 한다는 것이다. 그렇지 않다면, 그림 4(a)와 그림 4(b)에서처럼 D'의 크기를 (D의 반지름보단 작게) 조금 늘려 경계 위에 어떤 점도 오지 않게 한 후, 직선을 조금 움직여 D의 크기를 줄일 수 있게 된다. 이러한 모순은 두 디스크의 크기가 같아야 발생하지 않는다. 따라서 아래와 같이 정리할 수 있다.

사실 11. $k \geq 4$인 경우, $\ell(y^\star)$와 $R_k^{y^\star}(S)$는 다음의 세 가지 경우에 의해 결정된다. (1) D의 지름을 결정하는 두 점이 D의 경계에 오거나, (2) 세 점이 D의 경계 위에 오거나, (3) D와 같은 크기의 다른 디스크 D'이 존재하여, D 경계 위의 최대 두 점과 D' 경계 위의 최대 두 점이 오는 경우이다.

위의 사실은 결국 최대 네 점이 $\ell(y^\star)$와 $R_k^{y^\star}(S)$을 결정함을 의미한다. 먼저, S의 네 점의 조합에 의해 결정되는 $\ell(y)$와 $R_k^y(S)$을 계산한다. 각 조합에 대한 수평선과 반지름 쌍을 반지름에 따라 오름차순으로 정렬하여 $(\ell_1, r_1), (\ell_2, r_2), \cdots, (\ell_m, r_m)$라 하자. 여기서 $m = {}_nC_4 = O(n^4)$이다. 이 과정은 모두 $O(n^4 \log n)$ 시간이 필요하다. 그러면 m개의 수평선과 반지름 쌍 중에서 최적 쌍 $(\ell(y^\star), R_k^{y^\star}(S))$이 포함되어 있음을 사실 11에 의해 알 수 있다. 이 최적 쌍은 반지름에 대한 이진탐색으로 찾는다. 우선 $i = m/2$번째 쌍에 대해, 각 점에서 반지름이 r_i인 디스크를 그려 ℓ_i와의 교차 구간을 계산한 후, 교차 구간들의 최소 관통수(minimum stabbing number)가 k 이하인지를 판별한다. 만약 k 이하라면 $R_k^{y^\star}(S) \leq r_i$가 되어, 왼쪽 구간에서 이진탐색을 진행하고, k보다 크다면 $R_k^{y^\star}(S) > r_i$가 되어 오른쪽 구간에서 탐색을 계속한다. 매번 탐색 단계마다 $O(n \log n)$ 시간에 최소 관통수를 계산하기 때문에 전체 이진탐색 시간은 $O(n \log^2 n)$이 된다. 따라서 전체 알고리즘 수행시간은 $O(n^4 \log n)$이다.

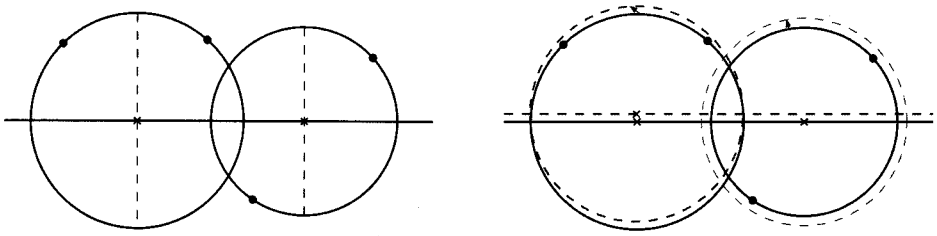
소정리 5. $k \geq 4$인 경우 $\ell(y^\star)$와 $R_k^{y^\star}(S)$는 $O(n^4 \log n)$ 시간에 계산할 수 있다.

정리 2. 임의의 $k \geq 2$와 점 집합 S가 주어질 때, 최적의 수평선 $\ell(y^\star)$와 최적 반지름 $R_k^{y^\star}(S)$는 $O(n^{\min\{4,k\}} \log n)$ 시간에 계산가능하다.

4. 결론 및 향후 연구 과제

본 논문에서는 이차원 평면에 주어진 점들을 기울기가 고정된 직선 위에 중심을 둔 k개의 디스크로 포함하는 데 필요한 최대 디스크의 반지름을 최소화하는 문제를 직선이 고정된 경우와 움직이는 경우로 나누어 각각 $O(\min\{n(\log n)^{\lceil \log k \rceil}, n^2 \log n\})$ 시간과 $O(n^{\min\{4,k\}} \log n)$ 시간에 해결할 수 있음을 보였다.

향후 연구과제는 본 논문에서 제시한 두 알고리즘보다 빠른 알고리즘을 설계하는 것이다. 특히, 직선이 움



(a) (b)

그림 4 $\ell(y^\star)$와 $R_k^{y^\star}(S)$를 결정하는 세 번째 경우에 대한 설명

직이는 경우의 알고리즘은 수행시간 단축 뿐 아니라 성능 좋은 근사 알고리즘을 설계하는 것도 의미 있는 과제라고 판단된다. $k \geq 4$인 경우에 사용된 단순한 접근법이 아닌 더 풍부한 기하학적 성질을 찾아내어 활용하는 방안이 강구되어야 한다. 또한, 임의의 기울기를 갖는 직선 위에 중심을 갖는 k-센터 문제도 고려할 수 있으며, 두 개 이상의 직선 위에 중심을 갖는 k-센터 문제도 흥미로운 문제이다.

참 고 문 헌

[1] N. Meggido, Applying parallel computation algorithms in the design of serial algorithms, Journal of ACM, 30, pp. 852-865, 1983.
[2] M. Sharir, A Near-Linear Algorithm for the Planar 2-Center Problem, Proc., 12th ACM Symp. on Computational Geometry, pp. 106-112, 1996.
[3] T. M. Chan, More planar two-center algorithms, Computational Geometry: Theory and Applications, 13, pp. 189-198, 1999.
[4] H. Alt, E. M. Arkin, H. Bronnimann, J. Erickson, S. P. Fekete, C. Knauer, J. Lenchner, J. S. B. Mitchell, K. Whittlesey, Minimum-Cost Coverage of Point Sets by Disks, 22th ACM Symp. on Computational Geometry, pp. 449-458, 2006[6].
[5] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, Compuational Geometry: Algorithms and Applications, 2nd Ed. Springer, 2000[6].

나 현 숙

1993년 서울대학교 수학과(학사). 1995년 포항공과대학교 수학과(석사). 2002년 포항공과대학교 수학과(박사). 2001년 3월~2002년 8월 프랑스 INRIA Post Doc. 2002년 9월~2003년 2월 홍콩과학기술대(HKUST) 전산학과 Post Doc. 2003년 3월~현재 숭실대학교 IT대학 컴퓨터학부(조교수). 관심분야는 알고리즘, 계산기하학, 정보이론

신 찬 수

1991년 서울대학교 계산통계학과(학사) 1993년 한국과학기술원 전산학과(석사) 1998년 한국과학기술원 전산학과(박사) 1998년~2001년 한국과학기술원 전산학과 BK21 연구 교수. 1999년~2000년 홍콩과학기술대(HKUST) 전산학과 Post Doc. 2001년~현재 한국외국어대학교 전자정보공학부(부교수). 관심분야는 알고리즘, 계산기하학, 컴퓨터그래픽스