アルゴリズム 81－7
（2001．11．27）

K - レベルとパラメトリック最小木の極値列挙について

加藤 直樹 (京都大学)　徳山 豪 (東北大学)

要旨

平面上の直線アレンジメントのK - レベルは、その組合せ複雑度が未解決であり、組合せ幾何学で注目されている研究対象である。 また、パラメトリック問題として見た時、選択問題（K番目に小さい元の計算）のパラメトリック化と考えることができる。 K－レベルにおける一つの大きな特徴は、高々K個の極大点しか持たないという点であり、これが数々の性質の基となっている。 本研究では、これらの極大点をすべて列挙する高速アルゴリズムを提案する。 アルゴリズムの計算時間は $O(n \log n) + \tilde{O}((kn)^{2/3})$ 時間であり、また、大きいほうから τ 個の極大点だけならば $O(n \log^2 n) + \tilde{O}((\tau n)^{2/3})$ 時間で列挙を行う．さらに、k－レベル問題はパラメトリックマトロイド問題の特殊な場合と捉えることができるが、パラメトリック最小木問題では、k－レベルのy座標に対応するものは、辺の重みが線形に変化するグラフでの最小木の最大重み辺（ボトルネック辺）になる。この問題にたいしては、ボトルネック辺を最大／最小にするパラメタ値を求める高速アルゴリズムを提案する。 アルゴリズムでは、単体領域探索に用いられるMatoušekの点集合分割を利用する。

Notes on computing peaks in k-levels and parametric spanning trees

Naoki Katoh[†], Takeshi Tokuyama[‡]

[†] Department of Architecture, Kyoto University, naoki@archi.kyoto-u.ac.jp,

[‡] Graduate School of Information Sciences, Tohoku University, tokuyama@dais.is.tohoku.ac.jp

Abstract

The k-level of an arrangement of lines is one of popular geometric objects in computational geometry [7]. Moreover, the k-level is a special case of the locus of the largest element of the minimum base of a parametric matroid [11, 8] with one parameter. We give an algorithm to compute all the local peaks in the k-level of an arrangement of n lines in $O(n \log n) + \tilde{O}((kn)^{2/3})$ time. We can also find τ largest peaks in $O(n \log^2 n) + \tilde{O}((\tau n)^{2/3})$ time. Moreover, we consider the longest edge in a parametric minimum spanning tree (in other words, a bottleneck edge for connectivity), and give an algorithm to compute the parameter value (within a given interval) maximizing/minimizing the length of the longest edge in MST. The time complexity is $\tilde{O}(n^{8/7}k^{1/7} + nk^{1/3})$.

1 Introduction

The k-level is the union of k-th lowest (closed) line-segments of the arrangement, and it can be considered as the trajectory of the k-th smallest element in a set of n data each of which depends on a parameter x linearly. From the viewpoing of combinatorial optimization, the k-level is the locus of the maximum element in the minimum base of a parametric uniform matroid of rank k. It is known that the complexity $g_k(n)$ of the k-level of an arrangement of n lines is $O(k^{1/3}n)$ [6] and $2^{\Omega(\sqrt{\log k})}n$ [16]. The upper bound holds for any parametric matroid (with a linear parameter) of rank k in n elements [8]. Moreover, it is known that the k-level can be computed in $O(g_k(n)\log^2 n)$ [4].

However, we often need a compact "outline" of a trajectory of a parametric problem by using a small number of characteristic points on it. Such an outline, generally speaking, will be useful as a compact data to control parametric problems, and possibly utilized in designing kinetic data structures [3]. Local peaks in the trajectory are considered to be natural characteristic points. A key observation to investigate the k-level is that it has at most $2k-1$ local peaks (at most k maximal peaks and at most $k-1$ minimal peaks) [2, 8]. One interesting problem is to compute all the local peaks efficiently. This enables us to give a decomposition of k-level into monotone chains, and hence create an outline with a size $O(k)$ of the k-level.

We give an algorithm to compute all the local peaks in the k-level of an arrangement of n lines in $O(n\log n)+\tilde{O}((kn)^{2/3})$, where $\tilde{O}$ is the big-O notation ignoring polylogarithmic factors. The current estimate for the polylogarithmic factor of the second term is less than $\log^5 n$; however we do not give it explicitly in this paper, since it is probably loose and will confuse readers. The time complexity is better than $O(g_k(n)\log^2 n)$ for some restricted range of k even if the current lower bound of $g_k(n)$ by Tóth [16] is tight. If we substitute the current $O(k^{1/3}n)$ upper bound to $g_k(n)$, the time complexity is better than $O(g_k(n)\log^2 n)$ if $k = O(n/\log^c n)$, where c is a suitable constant.

Another interesting question is how fast we can compute τ largest maximal peaks for $\tau \leq k$. If $\tau = 1$, Roos and Widmayer [14] gave a neat method to compute the maximum point in the k-level in $O(n\log n + (n-k)\log^2(n-k))$ time by using an efficient slope selection algorithm. We can compute τ largest peaks in $O(n\log^2 n) + \tilde{O}((\tau n)^{2/3})$ time by combining Roos and Widmayer's technique and the above mentioned method for computing all the peaks.

Finally, we investigate whether we can analogously treat some parametric matroids: Compute peaks in the trajectory of the largest element in the minimum weight base of a parametric matroid. In particular, the graphic matroid is of wide interest: Given a weighted undirected connected graph $G(x)$ with k nodes and n edges, such that each edge has a parametric weight that is linear in a parameter x. Here, k and n becomes the rank and size of the graphic matroid, respectively. Let $T(x)$ be the minimum weight spanning tree of $G(x)$ and consider the longest edge $e(x)$ in $T(x)$. Note that the minimum weight spanning tree becomes a spanning tree that minimizes the length of the longest edge. We call the edge $e(x)$ the *spanning bottleneck edge*(SBE), and write $SBE(x)$ and $w_{SBE}(x)$ for $e(x)$ and its weight, respectively. The naming comes from the fact that $w_{SBE}(x)$ is the minimum value of w such that the subgraph of $G(x)$ consisting of edges with weights less than or equal to w is connected.

The following problems are important in sensitivity analysis: (1). Compute the maximum value and the minimum value of $w_{SBE}(x)$ for $x \in I$, where I is a given interval. (2). Compute all peaks of the trajectory $y = w_{SBE}(x)$.

For example, imagine a system represented by the graph G where a link represented by an edge with a weight larger than a (controllable) threshold value becomes unreliable, and the edge weight depends on a parameter x linearly within an interval I. For a given subinterval $J \subset I$, we want to know the threshold value of the edge weights so that the graph remains connected for every $x \in J$. This can be reduced to the problem (1). Moreover, if we have computed all peaks in I as a preprocessing (problem (2)), we can efficiently query for the threshold value, provided that we have an efficient method (shown in Section 4) to compute $w_{SBE}(x)$ at endpoints of J.

Both of problems (1) and (2) can be solved by computing the whole picture of the transitions of the minimum spanning trees, and the time complexity of the current best algorithm is $O(kn \log n)$ [10]. Roos and Widmayer's method can be directly applied to the first problem. By using dynamic maintenance algorithms [9] of a minimum spanning tree, the time complexity becomes $O(\sqrt{k}n \log n)$. Combined with range searching techniques, we improve the time complexity to $\tilde{O}(n^{8/7}k^{1/7} + nk^{1/3})$. We give some discussion on the second problem, although theoretical improvement on the $O(kn \log n)$ time method remains open.

2 Preliminaries

2.1 Roos and Widamyer's algorithm

Given a set $\mathcal{H}$ of n lines in the x-y plane, let $\mathcal{L}_k$ be the k-level of the arrangement of $\mathcal{H}$. Let $\boldsymbol{p}$ be a point on $\mathcal{L}_k$ that has the maximum y-value y_{max}. Without loss of generality, we assume that such a point is unique. For any given value α, one can decide whether $y_{max} \geq \alpha$ or not in $O(n \log n)$ time: We sweep on the line $h : y = \alpha$ from the leftmost intersection point to the right to compute the levels of all intersection points on h with lines in the arrangement. If all intersection points are above the k-level, $\alpha > y_{max}$; otherwise, $\alpha \leq y_{max}$. By using this decision method, a binary search algorithm works to compute $\boldsymbol{p}$, and a weakly polynomial time algorithm with a time complexity $O(n \log n \log \Gamma)$ can be obtained, if each coefficient of the lines is a quotient number of integers with $\log \Gamma$ bits. Roos and Widmayer[14] applied an efficient slope selection method to transform the algorithm into strongly polynomial, and gave an $O(n \log^2 n)$ time algorithm. They further improved the time complexity to $O(n \log n + k \log^2 k)$ for computing the minimum and $O(n \log n + (n-k) \log^2(n-k))$ for computing the maximum.

2.2 Range query and Matoušek's point set decomposition

We use well-known (although sophisticated) simplex range query data structures [1]: We construct a data structure for a set S of n points in a plane such that given a query halfplane H, we can compute the number of points in S located in H efficiently. If we spend $O(m)$ time for constructing the data space for $n \log n \leq m \leq n^2$, the query time is $\tilde{O}(n/m^{1/2})$. The query can be done in polylogarithmic time by using $O(n/m^{1/2})$ processors. The data structure uses $\tilde{O}(m)$ space, although we do not discuss space complexity in this paper. Moreover, we can query the number of points in the intersection of two (or three) halfplanes in the same query time if we ignore a polylogarithmic factor. We can also do reporting query if we spend additional $O(N)$ time if the region contains N points.

Given a set $\mathcal{H}$ of n lines in a plane, we consider the set $\mathcal{D}(\mathcal{H})$ of their dual points: The dual point of a line $y = ax - b$ is (a, b). We construct a range searching data structure for $\mathcal{D}(\mathcal{H})$. Given a point $p = (x_0, y_0)$, the set of dual points of lines below p is

the set of points in $\mathcal{D}(\mathcal{H})$ located below the line $Y = x_0X - y_0$, where X and Y correspond to coordinates of the dual plane. Thus, we can compute the level of the point p in the arrangement of n lines by using half-plane range searching. Moreover, we have the highest line below p in the same query time. Also, we can query the number of lines which lie below both of a pair of query points.

A main building block for the range query is the point set decomposition structure of Matoušek, which we also need to utilize directly (we only describe its two-dimensional version):

Theorem 1 (Matoušek) *Given a set S of n points in the plane, for any given $r < n$, we can subdivide S into r disjoint subsets S_i $(i = 1, 2, \ldots, r)$ such that $|S_i| \leq 2n/r$ satisfying the following condition: Each S_i is enclosed in a triangle σ_i, and any line in the plane cuts at most $cr^{1/2}$ triangles among $\sigma_1, \sigma_2, \ldots, \sigma_r$ where c is a constant independent of n and r. Such a decomposition can be constructed in $O(n \log n)$ time.*

3 Computing all peaks in k-level

We assume $k \leq n/2$ for simplicity from now on; if $k > n/2$, replace k by $n - k$ and exchange maximal and minimal in the statements. A key observation for the k-level is that it is a subset of a union of k concave chains such that all concave vertices of the k-level are vertices of these concave chains [2]; thus, a k-level has at most k maximal peaks and $k - 1$ minimal peaks. We want to compute all the local peaks in a given interval I of the x-coordinate value. Without loss of generality, we assume that no line in the arrangement is horizontal nor vertical.

We prepare two key-subroutines: *one-shot query* and *peak counting*: Let $\boldsymbol{p}(x_0)$ be the point on the k-level at the x-coordinate value x_0. Let $\ell_k^-(x_0)$ (resp. $\ell_k^+(x_0)$) be the line in the k-level at the x-coordinate value $x_0 - \epsilon$ (resp. $x_0 + \epsilon$) for infinitesimally small $\epsilon > 0$. If x_0 is not an x-coordinate value of a vertex on the k-level, $\ell_k^-(x_0) = \ell_k^+(x_0)$. The above operation to compute the point (together with lines containing the point) on $\mathcal{L}_k$ at a given x-coordinate value is called *one-shot query*. One-shot query is an analogue of *ray shooting* [1], and thus the following lemma is basically well-known. The complexity $q(n, m)$ given in the lemma is called *one-shot query-time* for the k-level:

Lemma 2 *If we preprocess the lines in $\mathcal{H}$ with $O(m)$ time for $n \log n < m < n^2$, given an x-coordinate values x_0, we can compute $\boldsymbol{p}(x_0)$, $\ell_k^-(x_0)$ and $\ell_k^+(x_0)$ in $q(n, m) = \tilde{O}(n/m^{1/2})$ time, and also in polylogarithmic time by using $O(n/m^{1/2})$ processors.*

Proof By using the method given in the preliminary section, we can compute the level of any given point (x_0, y_0) in polylogarithmic time by using $O(n/m^{1/2})$ processors. We now apply parametric searching [15] to have the sequential time bound to compute the point $\boldsymbol{p}(x_0)$. □

The peak-counting is a routine to compute the number of peaks of the k-level in a given interval $J = [x_0, x_1]$ of x-coordinate values efficiently. The following elementary lemma is essential:

Lemma 3 *Let $f(x_0)$ and $f(x_1)$ are numbers of positive slope lines below or on the k-level at x_0 and x_1, respectively. Then, the number of maximal peaks of $\mathcal{L}_k$ in the interval J is $f(x_0) - f(x_1)$.*

Proof At-most-k-level (the part of the arrangement below $k + 1$-level) is a union of k concave chains such that all concave peaks in the chains appear in the k-level [2]. If a concave chain among them has a peak in J, the slope of the chain must be changed from positive to negative. Thus, the number of maximal peaks within J is the difference between the numbers of positive slope lines at two endpoints. □

Lemma 4 *For a given interval J of x-coordinate value, the number $\kappa(J)$ of peaks of $\mathcal{L}_k$ in J can be computed in $O(q(n,m))$ time if we preprocess the lines with $O(m)$ time. Also, the number of maximal peaks can be computed in $O(q(n,m))$ time.*

Proof If we construct the dual of range search data structure for the set of lines with positive slopes, the number of positive slope lines below a given point (x_0, y_0) can be computed in $O(q(n,m))$ time. Hence, $f(x_0)$ and $f(x_1)$ can be computed in $O(q(n,m))$ time, and the number of maximal peaks can be computed by using Lemma 3. The number of minimal peaks is easily computed from the number of maximal peaks and slopes of the k-level at endpoints. □

Now, we can apply a binary search paradigm to design a weakly-polynomial time algorithm. First, for the input interval I, we compute the number of peaks $\kappa = \kappa(I)$ within the interval. The time complexity for this initialization is negligible, and obviously $\kappa \leq 2k-1$. Next, we construct a data structure for the one-shot query in $O(m)$ time, where the choice of m will be explained later. Suppose that coefficients of the equations of lines are quotient numbers of $\log \Gamma$ bit integers. We apply κ-branching binary search to find all peaks; At each stage of the binary search, we have at most κ subintervals which has at least one local peak of $\mathcal{L}_k$ (such a subinterval is called an *active interval*), and we recursively search in active subintervals. Thus, after examining $\kappa \log \Gamma$ candidates of x-coordinate values, we can find all of the peaks. To make the complexity into strongly polynomial, we apply parametric search [15] by using the parallel algorithm for the one-shot query given in Lemma 3 as its *guide algorithm* (we omit details in this version).

Theorem 5 *All the peaks on $\mathcal{L}_k$ within an interval I can be computed in $O(n \log n) + \tilde{O}((\kappa n)^{2/3})$ time if I has κ local peaks.*

Since $\kappa \leq 2k$, we have the following:

Corollary 6 *All the peaks on $\mathcal{L}_k$ can be computed in $O(n \log n) + \tilde{O}((kn)^{2/3})$ time.*

When κ is large, it may be too expensive to compute all the local peaks. Suppose that we want to compute τ largest maximal peaks in the input interval I for $\tau \ll \kappa$ more efficiently than computing all the peaks. This can be done by combining Roos and Widmayer's algorithm and the algorithm given above.

Theorem 7 *We can compute τ largest/smallest maximal/minimal peaks of $\mathcal{L}_k$ in an interval I in $O(n \log^2 n) + \tilde{O}((\tau n)^{2/3})$ time. We can also compute τ largest local peaks (including both maximal and minimal peaks) in the same time complexity.*

4 Bottleneck edge length in a parametric spanning tree

Next, we consider the parametric spanning tree problem. Consider a connected graph $G = (V, E)$ with k nodes and n edges. Because of the connectivity, $k - 1 \leq n \leq k(k+1)/2$. For each edge $e \in E$, we associate a weight function $w_e(x)$, which is linear on a parameter x. We assume that the arrangement generated by lines $y = w_e(x) : e \in E$ is simple, i.e., no three lines intersect at a point. We can remove this assumption by giving a symbolic perturbation. G is denoted by $G(x)$ if it is considered as a weighted graph with parametric weights. For a given value x, we consider the minimum spanning tree $T(x)$ of $G(x)$.

It is known that the number of transitions of the structure of the minimum spanning tree $T(x)$ is $O(k^{1/3}n)$, and all the transitions can be computed in $O(kn \log n)$ time [10]. Moreover, the average edge weight in the minimum spanning tree is a concave function in x, and the value of x maximizing

the average edge weight of $T(x)$ can be computed in $O(n \log n)$ time [10].

As parametric matroid problems, the average edge weight is a counterpart of the average of y-values of k lines below (or on) the k-level. A natural counterpart of the k-level itself in the minimum spanning tree is the longest (i.e. maximum weight) edge in the minimum spanning tree. The edge is also called the *spanning bottleneck edge* at x ($SBE(x)$ in short), and its weight is denoted by $w_{SBE}(x)$. It is easily observed that $W_{SBE}(x)$ is the minimum value of w such that the subgraph of $G(x)$ constructed from the set of edges whose weights are less than or equal to w is connected.

It is natural and important problem in sensitivity analysis [11] to trace the trajectory $y = w_{SBE}(x)$ of the weight of $SBE(x)$. Analogously to the k-level, there are at most k maximal peaks in the trajectory $y = w_{SBE}(x)$. We want to compute peaks in the trajectory.

4.1 One-shot query for the longest edge in MST

We first consider efficient query for $SBE(x_0)$ at any given value of x_0 of the parameter. This query is called *one-shot query for the SBE*. A naive method is the following: First construct $T(x_0)$ in $O(n)$ time, and select its longest edge. Instead, we use the Matoušek's set partition. In the dual space, the dual points of n weight functions of the edges are partitioned into r subsets of size $O(n/r)$. Each subset is contained in a triangle, and $O(n^{1/2})$ triangles are cut by any query line.

Accordingly, we partition the set of n edges into r subsets each has $O(n/r)$ edges. For each subset, we compute a spanning forest (irrelevant to edge weights) and store the connected components except singletons. Thus, each component has a forest with $O(\min\{k, n/r\})$ edges. This computation can be done in $O(n)$ additional time.

If we are given a parameter value x_0, we sort $O(r)$ vertices of the triangles with respect to the inner product of them with the vector $(x_0, 1)$. We do binary search on this sorting list. We guess a vertex v, and consider a line $\ell : Y = x_0 X + c$ which goes through v. We recognize the triangles which is below ℓ; thus, the edges in the subsets associated with the triangles has weights which is less than c. We construct a spanning forest F of the union of forests in these subsets: since they have $O(rk)$ edges, this can be done in $O(rk)$ time. If the forest F is a spanning tree, we decide that v is too large in the sorting list, and continue the binary search.

Otherwise, we consider the subsets associated with the triangles cut by ℓ. They contain $O(n/r^{1/2})$ points in total. We sort them with respect to the weights, and greedily insert them into F until we have a spanning tree. If we do not have a spanning tree, we decide v is too small, and continue the binary search. If we have a spanning tree, we decide v is a candidate, but it may be too large, and continue to search for the lowest vertex v satisfying the above condition, and return the longest edge in the tree for that v. Note that the spanning tree is not a minimum spanning tree in general; however, we correctly recognize the longest edge in a minimum spanning tree.

This process needs $O(rk + n/r^{1/2})$ time, and we do this process $O(\log r)$ times during the binary search. Thus, if we set $r = (n/k)^{2/3}$, the time complexity is $O(n^{2/3}k^{1/3}\log(n/k))$, which is slightly better than $O(n)$ if $k = o(n)$. By applying a hierarchical subdivision, we can further improve it: We fist start $r = r_1$, and decompose the subset of size $O(r_1 n)$ into r_2 smaller subsets, where $r_2 = r_1^{1/2}$, and we further continue the refinement for $r_i = r_{i-1}^{1/2}$ until r_i becomes below a constant. The query time becomes

$$k(r_1 + r_1^{1/2} r_2 + \ldots + (r_1 r_2 \ldots r_{i-1})^{1/2} r_i) + n/(r_1 r_2 \ldots r_i)^{1/2}.$$

Setting $r_1 = (n/k)^{1/2}$, this enables $\tilde{O}((nk)^{1/2})$ time computation for $SBE(x_0)$. Similarly to the case of halfspace range searching, we can combine hierarchical cutting of the arrangement to have a preprocess-query trade-off (we omit details in this version). Indeed, we have the following proposition:

Proposition 8 *If we spend $\tilde{O}(m)$ preprocessing time for $n \leq m$, we can do the one-shot query for SBE in $\tilde{O}(n/(m/k)^{1/2} + k)$ time.*

Moreover, we will later use the following *two-shot reporting query* for a spanning forest, which reports a spanning forest consisting of edges whose weight functions are below both of given two query points (x_0, y_0) and (x_1, y_1). This can be done similarly to one-shot query (this is a counterpart of the simplex range searching if the one-shot query is a counterpart of the halfplane range searching).

Proposition 9 *If we spend $\tilde{O}(m)$ preprocessing time for $n \leq m$, we can do the two-shot reporting query in $\tilde{O}(n/(m/k)^{1/2} + k)$ time.*

4.2 Computing the maximum peak

Let us consider the problem of computing the maximum peak in I. First, we straightforwardly apply Roos-Widmayer's algorithm. For a given y-value y_0, we want to decide whether $Max_{x \in I} w_{SBE}(x) \leq y_0$ or not. We dynamically update the spanning forest associated with edges with weight below y_0 from $x = x_0$ to $x = x_1$ if $I = [x_0, x_1]$. If we find a value $x \in I$ such that the spanning forest becomes a spanning tree, we know $Max_{x \in I} w_{SBE}(x) \leq y_0$. It costs $O(k^{1/2})$ time to update a minimum spanning forest for insertion and deletion of edges. Suppose that we sweep on the line $y = y_0$ updating the minimum spanning forest. The line $y = y_0$ has at most n intersections with lines associated with weight functions, and hence the method needs $O(nk^{1/2})$ time for the decision. Thus, the maximum peak can be found in $O(nk^{1/2} \log n)$ time.

We try to improve the above time complexity. We subdivide the line $y = y_0$ into $\lceil n/s \rceil$ intervals such that each interval contains at most s intersection points. For each interval $I_i = [x_i, x_{i+1}]$, we perform the two-shot reporting query at (x_i, y_0) and (x_{i+1}, y_0). The reported forest F is constructed from edges whose weight is less than y_0 both at $x = x_i$ and $x = x_{i+1}$. If the forest has more than $s + 1$ connected components, it is impossible that $w_{SBE}(t) \leq y_0$ for a $t \in I_i$. Otherwise, we dynamically maintain the spanning tree, where we contract nodes into at most $s + 1$ super nodes each of which associate with a connected component of the forest F. Our graph has only s edges, and hence the update can be done $O(\sqrt{s})$ time per intersection.

Hence, total time complexity becomes $\tilde{O}(n\sqrt{s} + (n/s)[n/(m/k)^{1/2} + k] + m)$. If we optimize this, we have $\tilde{O}(n^{8/7}k^{1/7} + nk^{1/3})$. This is an improvement over $O(nk^{1/2})$, since $n \leq k(k+1)/2$.

The minimum of $w_{SBE}(x)$ can be analogously computed. Hence, we have the following theorem:

Theorem 10 *For a given interval I of the parameter value x, we can compute both the maximum and the minimum of $w_{SBE}(x)$ for $x \in I$ in $\tilde{O}(n^{8/7}k^{1/7} + nk^{1/3})$ time.*

We can generalize the above theorem for the *truncated matroid* of the graphic matroid to obtain the following proposition (we omit the proof):

Proposition 11 *For a constant c, let $w_{SBE-c}(x)$ be the minimum value of w such that the set of edges in $G(x)$ with weights less than or equal to w has at most c connected components. We can compute both the maximum and the minimum of $w_{SBE-c}(x)$ in $\tilde{O}(n^{8/7}k^{1/7} + nk^{1/3})$ time.*

Corollary 12 *We can compute the maximum minimal peak of $w_{SBE}(x)$ in $\tilde{O}(n^{8/7}k^{1/7} + nk^{1/3})$ time.*

Proof A minimal peak of $w_{SBE}(x)$ corresponds to a maximal peak in $w_{SBE-2}(x)$ in an one-to-one fashion; hence we have the corollary. □

5 Concluding remarks

The number of maximal peaks in a k-level is known to be at most ${}_kC_d$ if we have d dimensions [5]. Hence, this is much smaller than the complexity of whole arrangement, especially if k is much smaller than n. However, to the author's knowledge, the problem of computing peaks in the k-level for the three dimensional arrangement is open, since the structure is much more complicated than the planar case [12]. One necessary constituent is to develop a counterpart of Lemma 3: Given an arrangement of n hyperplanes in the three-dimensional space, preprocess it, and for any given three points A, B, and C in the plane $z = 0$, decide whether the triangle ABC contains (a projection) of a peak in the k-level or not efficiently. For the purpose, we probably need a counterpart of Lemma 3: Give a criterion of the existence of a peak from the information of the set of hyperplanes below k-level at each of A, B, and C. In two-dimensional space, the lines are classified into positive slope lines and negative slope lines, while this kind of natural discrete classification of planes in the space does not exist.

参考文献

[1] P. Agarwal, "Range Searching," Section 31 of *Handbook of Discrete and Computational Geometry,* (1997), 575–598, CRC Press.

[2] P. Agarwal, B. Aronov, T. Chan, and M. Sharir, "On Levels in Arrangement of Lines, Segments, Planes, and Triangles," *Discrete & Comput. Geom.,* **19** (1998), pp. 315-331.

[3] J. Bash, L. Guibas, and H. Hershberger, "Data Structures for Mobile Data," *Proc. 8th ACM-SIAM Symp. on Disc. Alg.,* (1997), pp. 747-756.

[4] R. Cole, M. Sharir, and C. Yap, "On k-hulls and related problems," *SIAM J. Comput.,* **16** (1987), pp. 61-77.

[5] K. Clarkson, "A Bound on Local Minima of Arrangement That Implies the Upper Bound Theorem," *Discrete & Comput. Geom.,* 10 (1993), pp. 427-433.

[6] T. Dey, "Improved Bound on Planar K-Sets and Related Problems," *Discrete & Comput. Geom.,* 19 (1998), pp. 373-383.

[7] H. Edelsbrunner, *Algorithms in Combinatorial Geometry,* ETACS Monographs on TCS 10, Springer-Verlag, 1987.

[8] D. Eppstein, "Geometric Lower Bounds for Parametric Matroid Optimization," *Discrete & Comput. Geom.,* (1998), pp. 463-476.

[9] D. Eppstein, Z. Galil, G.F.Italiano, and A. Nissenzweig, "Sparcification–A Technique for Speeding Up Dynamic Graph Algorithms," *J. ACM,* **44-5** (1997), pp. 669-696.

[10] D. Fernandez-Baca, G. Slutki, and D. Eppstein, " Using Sparcification for Parametric Minimum Spanning Tree Problems," *Nordic J. Computing,* **3-4** (1996), pp. 352-366.

[11] D. Gusfield, *Sensitivity Analysis for Combinatorial Optimization,* Ph. D. Thesis, Memorandum No. UCB/ERL M80/22, U.C. Berkeley, 1980.

[12] N. Katoh and T. Tokuyama, "Lovász's Theorem for the K-level of an Arrangement of Concave Surfaces and Its Applications," *Proceedings of 40th IEEE FOCS,* (1999), pp. 389-398.

[13] J. Matoušek, "Efficient Partition Trees," *Discrete & Comput. Geom.,* **10** (1992), pp. 315–334.

[14] T. Roos and P. Widmayer, "K-Violation Linear Programming," *Information Processing Letters,* **52** (1994), pp. 109-114.

[15] J. Salowe, Parametric Search, Section 37 of *Handbook of Discrete and Computational Geometry,* (1997), 683–695, CRC Press.

[16] G. Tóth, "Point Sets with Many k-sets," *Proc. 16th SOCG,* (2000), pp. 37-42.