アルゴリズム 48－14
（1995．11．17）

最適区間問題と計算幾何学

福田 剛志　森本 康彦　森下 真一　徳山 豪
日本ＩＢＭ 東京基礎研究所

要旨. データ集合Yを考える。それぞれのデータpに対して、区間$[1,N]$に入るキー$key(p)$と共に幾つかの実数値$u_i(p)$が与えられている。区間$I=[s,t]$を取った時、$u_i(I)=\sum_{key(p)\in I} u_i(p)$とする。このとき、$u_i(I)$たちに関連した幾つかの条件を考え、対応した最適区間を求める事を考える。判りやすく言うと、数理計画法（線形計画法など）の形式を離散データの区間上の関数に対して当てはめた問題である。本論文では、これらの問題を解く計算幾何学的アルゴリズムを提案し、データベースの解析（データマイニング）への応用を述べる。更に、画像処理との関連で最適領域を求める問題への一般化も考える。

Finding optimal intervals using computational geometry

Takeshi Fukuda, Yasuhiko Morimoto, Shinich Morishita, and Takeshi Tokuyama
IBM Japan, Tokyo Research Laboratory

Abstract. We consider a set Y of data. Each data p has a key $key(p)$ and real values $u_i(p)$ $(i=0,1,2,..,h)$. We assume $key(p)$ is an integer in $[1,n]$. For an interval $I=[s,t]$, let $u_i(I)=\sum_{key(p)\in I} u_i(p)$. We consider some conditions and objective function related to $u_i(I)$, and find the optimal interval I. Intuitively, this is a modification of the formulation of mathematical programming (e.g. LP) to functions on intervals of discrete data. We propose efficient geometric algorithms, and discuss its application to data mining. We also consider its two-dimensional extension in application to image processing.

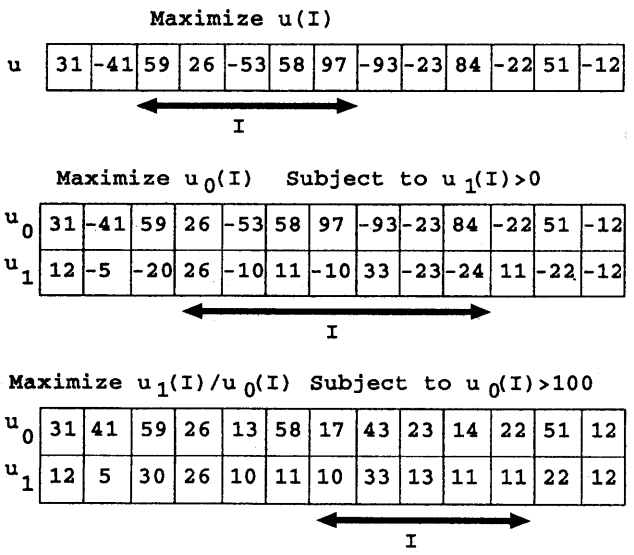

図 1: Optimal interval finding problems

1 Introduction

Suppose we have a set Y of n data, each of which has a primary integral key $key()$ in $[0, N]$ and numerical values $u_i()$ for $i = 0, 1, 2, .., h$. If $u_i(p) \geq 0$ (resp. $u_i(p) \leq 0$) for each $p \in Y$, we say u_i is positive (resp. negative) monotone. Without loss of generality, we assume $N = n$ unless specifically declared. For an interval $I = [s, t]$, let $u_i(I) = \sum_{key(p) \in I} u_i(p)$.

A basic operation in data base searching is the *interval searching*, which is formulated as follows: "For a given interval I, compute $u_i(I)$." This can be trivially done in $O(n)$ time. Moreover, after preprocessing the data in $O(n \log n)$ time, the interval searching can be done in $O(\log n)$ time [7]. In this paper, we consider reversal problems of the interval searching:

Optimal-Interval Finding (OIF): "Given real numbers K_i (i=1,2,..,h), compute the interval I maximizing $u_0(I)$ under the condition that $u_i(I) > K_i$ for $i = 1, 2, .., h$."

LP-type OIF "Under costraints $f_l(I) = a_l^1 u_1(I) + ... + a_l^h u_h(I) > K_l$ ($l = 1, 2, .., m$ and $N >> m >> h$), maximize $u_0(I)$."

Ratio Optimizing Interval Finding (ROIF): "Compute the interval I maximizing $u_1(I)/u_0(I)$ under the condition that $u_0(I) > K$ and u_0 is positive monotone"

Generalized Ratio Optimizing Interval Finding (GROIF): "Compute the interval I maximizing $u_1(I)/u_0(I)$ under the condition that $u_i(I) > K_i$ for $i = 0, 1, 2, 3, ..., h$ and $K_0 \geq 0$"

Functions $u_0(I)$ in OIF and $u_1(I)/u_0(I)$ in ROIF and GROIF are called the *objective functions*, while the other u_i ($i = 1, 2, .., h$) are called the *conditional functions*.

Since there are at most $O(N^2)$ intervals, it is trivial to give $O(N^3)$ time solution for each of problems, and easy to improve it to $O(N^2)$. Our aim is to design better algorithms when h is a small constant. We give an $O(N \log^h N)$ time and $O(N)$ space solution of OIF. If the objective function u_0 is monotone, the time complexity is reduced to $O(N \log^{h-1} N)$ time. If each of the condition functions is (positive or negative) monotone, OIF can be solved in $O(N)$ time. LP-type OIF can be solved in $O(m^{\lfloor h/2 \rfloor} N \log^{h+2} N)$ time and space. ROIF can be solved in $O(N)$ time and space. GRIOF can be solved in $O(N \log^{h+2})$ time and $O(N \log^{h+1})$ space.

We also consider the following two dimensional version in $O(n)$ time. Suppose that each data p has a secondary key $key'(p)$, which is an integer in $\{1, .., N\}$ (here, we do not assume $n = O(N)$). We consider the $N \times N$ grid G. Then, $\mathbf{KEY}(p) = (key(p), key'(p))$ is a lattice point in G.

Optimal Region Finding (ORF): "Find the connected x monotone region R so that $Z(R) = \sum_{\mathbf{KEY}(p) \in R} u_0(p)$ is maximized."

2 Applications

Data mining.

Imagine a database containing data of n customers. For each customer, $key(p)$ is his/her age, and $u_0(p)$ is the number of game softwares which he/she buys a year. A game company want to make a series of new game software, and want to optimize the target customers. For the purpose, a planner wants to find the age interval attaining the maximum software purchase under the condition that at least 0.5 software per customer is bought in average. We define $u_1(p) = u_0(p) - 0.5$ for all p. Then, the problem is formulated as:

"Find an interval maximizing $u_0(I)$ under the condition that $u_1(I) > 0$", which is an OIF problem, where the objective function u_0 is (positive) monotone.

Suppose that he wants to find an age-interval containing at least 10,000 customers, and maximize the profit ratio, assuming that the expence is propotional to the number of customers. Then, defining $u_2(p) = 1$ for all p, we want to solve

"Find an interval maximizing $u_0(I)/u_2(I)$ under the condition that $u_2(I) > 10,000$," which is an ROIF problem.

These two are typical examples of of *data mining*, which is a general concept for rule (*association rule*) finding problems from a data base. Efficient query methods of the optimal intervals are crucial for such data mining systems. (In the above example, "age" is up to 100 or so, but in other applications, N may become much larger).

Next, we consider a data, where $key(p)$ and $key'(p)$ are age and income of the customer p, respectively. We define (for instance) $u_0(p) = t(p) - 2w$ if p has $t(p)$ computers and w is the average number of computers a customer has. Then, the solution of ORF gives the age-income region in which customers are tend to be heavy computer-users.

All of OIF (for $h = 1$ and a monotone objective function), ROIF, and ORF have been already implemented as data mining functions, and tested on real data base [6].

Image segmentation. Suppose the data Y corresponds to the set of pixels in $N \times N$ pixel grid. Each pixel p has a gray level $g(p)$. We define $key(p)$ and $key'(p)$ corresponding to the coordinates of the pixel, and $u_0(p) = g(p) - \alpha$ for a suitable constant α. The solution of ORF of the above Y is called a *focused image* associated with α, and plays a key role in image segmentation [3].

3 Algorithms for interval finding

3.1 Programming Pearls

A very easy case of OIF can be found in the "Programming Pearls" (column of J. Bentley in C. ACM [4]), in which the problem is used as an example to demonstrate how algorithm design techniques are important to write an efficient program.

Problem PP: "Compute the interval I maximizing $u_0(I)$"

Bentley showed four algorithms, whose time complexities are $O(N^3)$, $O(N^2)$, $O(N \log N)$, and $O(N)$, respectively. The linear time algorithm scans the data with respect to the key. For each i, $MaxEndHere(i) = \max_{s \le i} u_0([s,i])$ and $MaxSofar(i) = \max_{s \le t \le i} u_0([s,t])$. Then, $MaxSofar(N)$ is our answer.

We have relations that $MaxEndHere(i+1) = \max\{0, MaxEndHere(i) + u_0(\{i\})\}$ and $MaxSofar(i+1) = \max\{MaxEndHere(i+1), MaxSofar(i)\}$. Thus, a simple dynamic programming gives $O(N)$ time solution. Unfortunately, this method does not work if we generalize Problem PP to OIF.

3.2 Monotonically-conditioned interval finding

We give a solution of OIF when conditional functions are monotone, based on a fast matrix searching algorithm. First, we note that $u_i(I)$ $(i = 0, 1, .., h)$ can be queried in constant time by giving some preprocessing:

Lemma 1 *After $O(n)$ preprocessing, we can query $u_i(I)$ for each given I in $O(1)$ time.*

Proof: We computes $u_i([0,s])$ for all s in $O(n)$ time, and store them in tables. Then, we can compute $u_i((s,t]) = u_i([0,t]) - u_i([0,s])$ in $O(1)$ time. □

A function $f(I)$ is "convex" if $f(I) + f(I') \le f(I \cup I') + f(I \cap I')$ for every pair of I and I' if $I \cap I' \neq \emptyset$. $u_i(I)$ are convex (moreover, the equality holds instead of the inequality above).

For each s, we define $first_i(s)$ (resp. $last_i(s)$) to be the minimum (resp. maximum) index t such that $u_i([s,t]) > K_i$. If there is no such index, we define $last_i(s) = first_i(s) = N+1$. Based on the following lemma, which is immediate consequence of the monotonicity, we can compute $first_i(s)$ and $last_i(s)$ for $s = 1, 2, .., N$ in $O(N)$ time.

Lemma 2 *For each $i = 1, 2, .., h$, $first_i(s) \ge first_i(s')$ and $last_i(s) \ge last_i(s')$ if $s \ge s'$.*

For each s, $opt(s)$ is the index such that $first_i(s) \leq opt(s) \leq last_i(s)$ and $u_0([s, opt(s)]) \geq u_0([s,t])$ for every $first(s) \leq t \leq last(s)$. The solution of OIF is $\max_{s=1,2,..,N} u_0([s, opt(s)])$.

Lemma 3 *The indices $opt(s)$ $(s = 1, 2, ... N)$ can be computed in $O(N)$ time. Moreover, it can be computed in $O(\log N)$ time using $O(N)$ processors in CREW model.*

Proof: Because of the convexity, we can apply fast matrix searching method in a *Monge matrix* [1, 2]. □

Theorem 1 *OIF can be solved in $O(N)$ time if all conditional functions are monotone. Also, it can be computed in $O(\log N)$ time using $O(N)$ processors.*

3.3 Optimal interval finding

The non-monotonicity of the condition functions gives a drastic change on the solution. Since each data p has $h+1$ values $u_i(p)$ together with the key $key(p)$, it can be considered as a point in $h+2$-dimensional space. So, we consider the OIF as a computational-geometric problem. However, we consider a different point set from $\{(key(p), u_0(p), ..., u_h(p)) : p \in \Upsilon\}$ in order to apply techniques of computational geometry.

Let us consider a set $S = \{q_i : i = 1, 2, ..N\}$ of points in $h+2$-dimensional space defined by $q_i = (x_i^{(0)}, x_i^{(1)}, ...x_i^{(h)}, x_i^{(h+1)}) = (v([0,i]), u_1([0,i]), ..., u_h([0,i]), i)$.

A point $q = (x^{(0)}, .., x^{(h+1)})$ dominates other point $q = (x'^{(0)}, .., x'^{(h+1)})$ if $x^{(i)} \geq x'^{(i)}$ for each $i = 0, 1, ..., h+1$. We say a point q is dominated by a set A of points if there exists a point $q' \in A$ dominating q. A point q_j of S is a maximal (resp. minimal) point of S if there is no other point q_l of S dominating (resp. dominated by) q_j, for $i = 1, 2, .., h$. The set of maximal (resp. minimal) points of S is called the maxima (resp. minima).

From a point q in the space, let $F(q)$ be the unique point of S (if exists) which has the maximum $x^{(1)}$-coordinate value among the points dominating q. It is easy to see that if there is a point of S dominating q, $F(q)$ exists, and it is in the maxima of S. We define $\mathbf{K} = (0, K_1, K_2, ..., K_h, 0)$, and $q + \mathbf{K}$ is the point associated with the vector sum of q and $\mathbf{K}$.

Lemma 4 *If $I = (s,t]$ is the solution of the OIF, q_s is a minimal point, q_t is a maximal point, and $q_t = F(q_s + \mathbf{K})$.*

So, it is sufficient to compute $F(q_s + \mathbf{K})$ for all $s = 1, 2, .., N$ in order to solve OIF. Although it is not easy to compute $F(q)$ for each q in polylogarithmic time using a linear space data structure, we have an efficient algorithm to compute $F(q)$ for all $q \in S$ together.

Lemma 5 *$F(q_s)$ for all $s = 1, 2, .., N$ can be computed in $O(N \log^h N)$ time and $O(N)$ space.*

Proof: For simplicity, we assume that $N = 2^m$ for an integer m. We sort the list $q_1, q_2, ..., q_N$ (originally sorted with respect to the $h+2$-th coordinate) into a sorted list $\tilde{q}_1, ..., \tilde{q}_N$ with respect to the first-coordinate (corresoonding to $v = u_0$). We define $S_{L,b} = \{q_{2^bL+1}, q_{2^bL+2} \cdots, q_{2^b(L+1)}\}$.

Then, we can compute $F(\tilde{q}_s + \mathbf{K})$ for $s = 1, 2, .., N$ as follows: Let $S + \mathbf{K} = \{q + \mathbf{K} : q \in S\}$. First, we compute the subset $U(1)$ consisting points of $S + \mathbf{K}$ whose projection images by pr are dominated by $pr(S_{1,m-1})$. We define $U(2) = S - U(1)$ (set difference). Then, $F(z)$ for $z \in U(1)$ is in $S_{1,m-1}$, whereas that for $z \in U(2)$ is in $S_{0,m-1}$, if there exists a point of S dominating z.

We next compute the subset $U(1,1)$ consisting of points of U_1 whose projection images are dominated by $pr(S_{3,m-2})$, and define $U(1,2) = U(1) - U(1,1)$. Similarly, we compute the subset $U(2,1)$ consisting of points of $U(2)$ whose projection images are dominated by $pr(S_{1,m-2})$, and define $U(2,2) = U(2) - U(2,1)$. We continue this operation until we compute $U(e_1, ..., e_m)$ for all $e_i \in \{1, 2\}$. Then, $F(z)$ for a point z in $U(e_1, ..., e_m)$ is q_{M+1} where $M = e_1 2^{m-1} + e_2 2^{m-2} + ... + e_m$ if it dominates z (otherwise, no point of S dominates z).

It is known that for point sets A and B in R^d ($d \geq 2$) of total size M, respectively, we can find the subset consisting of points of A dominated by B in $O(M \log^{d-2} M)$ time [5, 7] and linear space, if the point sets are sorted with respect to a coordinate. (this operation is called *filtering* in [7], and used as a subroutine of computing maxima of a point set). In parallel model, the operation can be done in $O(\log M)$ time using $O(M \log^{d-2} M)$ processors.

Hence, $U(1)$ can be computed in $O(N \log^{h-1} N)$ time, and for each fixed $l \leq m = \log N$, $U(e_1, e_2, ..., e_{l-1}, 1)$ can be computed from $U(e_1, e_2, ..., e_{l-1})$ in $O((2^{m-l+1} + |U(e_1, e_2, ..., e_{l-1})|) \log^{h-1} N)$ time.

Since $\sum_{e_i \in \{1,2\}} |U(e_1, e_2, ..., e_{l-1})| = N$, the time complexity of our algoithm is $O(N \log^h N)$, and the space complexity of the algorithm is $O(N)$. □

The above method is named *group ray shooting*, which is a multidimensional divide & conquer algorithm.

When the objective function $v = u_0$ is monotone, then, $i < j$ if and only if $x_i^{(0)} < x_j^{(0)}$. Thus, we can ignore the last coodinate of q_i. Hence, we have the following:

Theorem 2 *OIF with h conditional functions can be solved in $O(N \log^h N)$ time and $O(N)$ space. Moreover, it can be solved in $O(\log^2 N)$ time using $O(N \log^{h-1} N)$ processors. If the objective function is monotone, the sequential time complexity and the processor number in parallel computation can be reduced by a factor of $\log N$.*

Remark. If $h = 1$ and the objective function is monotone, the above theorem claims $O(N)$ time solution, which cannot be obtained by using group ray shooting naively. In this case, we construct the 2-dimensional maxima of sorted point set in linear time, compute $F(z)$ by using it, and solve OIF in linear time.

3.4 LP-type OIF

Because of space limitation, we omit details of the LP-type OIF problem. Based on modified orthogonal range searching method [8, 7], we have the following theorem:

Theorem 3 *If h is a small constant, and $h << m << N$, LP-type OIF can be solved in $O(m^{\lfloor h/2 \rfloor} N \log^{h+2} N)$ time and space. If the objective function is monotone, the complexities can be reduced by a factor of $\log N$. Moreover, if $h = 2$ and the objective function is monotone, the problem can be solved in $O(mN \log^2 N)$ time and $O(mN \log N)$ space.*

3.5 Algorithm for ROIF

We use a point set $S = \{q_i = (u_0([0, i]), u_1([0, i]))\}$ in a plane. Note that we use the symbols q_i and S for different ones from the previous section. For $I = [s, t]$, $u_1(I)/u_0(I) = (y_t - y_s)/(x_t - x_s)$ is the slope of the segment between q_s and q_t. For each point q_i, let $S_K^+(i) = \{q_j : x_j - x_i > K\}$. Let $UConv(S_K(i))$ be the upper convex hull (upper chain of the convex hull) of $S_K^+(i)$. We also consider a point set $S(i) = \{q_j : x_j \leq x_i\}$ and its lower convex hull $LConv(S(i))$. Let $slope(i)$ be the slope of the common tangent line $line(i)$ of $LConv(S(i))$ and $UConv(S_K(i))$.

Lemma 6 *If $I = (s, t]$ is the solution of RIOF, q_t is a vertex of $UConv(S_K(s))$, and $q_s q_t$ is a common tangent segment to convex chains $LConv(S(s))$ and $UConv(S_K(s))$. Moreover, slope of $q_s q_t$ is $\max_i\{slope(i)\}$.*

Thus, it suffices to compute the maximum of $slope(i)$, together with its associated common tangent segment. Let us consider a dynamic algorithm which updates $LConv(S(i))$, $UConv(S_K(i))$, and $line(i)$ in an on-line fashion by increasing i from 1 to N. We omit the proof of the following lemma in this version:

Lemma 7 *The updating time is $O(1)$ amortized time for each i.*

Theorem 4 *The ROIF can be solved in $O(N)$ time.*

3.6 Algorithm for GROIF

Let $q_i = (u_0([0,i]),...,u_h([0,i]))$, $pr_1(q_i) = (u_0([0,i]), u_1([0,i]))$, $S = \{q_i : i = 1,2,..,N\}$, and $S(< i) = \{q_i, q_{i+1},..., q_N\}$.

We also define $q_i + \mathbf{K} = (u_0([0,i]) + K_0,..., u_h([0,i]) + K_h)$. Let $W(s,m) = \{q_j \in S(< m) : q_s + \mathbf{K}$ *is dominated by* $q_j\}$. We consider a $(h+1)$-dimensional dynamic orthogonal range search data structure with $O(N\log^{h+1} N)$ space, which stores *principal subsets* of $S(< m)$, so that $W(s,m)$ is reported as a union of $O(\log^{h+1} N)$ principal subsets. The data structure is based on range search tree [8]. (Note that the above preformance is worse by a $\log N$ factor in both space and time complexity than that of counting query model of Willard [8], since we must store a principal subset and a secondary structure shown below in each nodes of the range search tree.)

For each principal subset A, we precompute 2-dimensional upper convex hull of $pr_2(A)$, so that we can compute the tangent from $pr_2(q_s)$ to $pr_2(A)$ in $O(\log N)$ time. If we decrement m by one, the subset can be updated in $O(\log^{h+2} N)$ time.

For each s, we query $W(s,s)$; then, we can compute the interval $I(s) = [s, opt(s)]$ which maximize $u_1(I)/u_0(I)$ in $O(\log^{h+2} N)$ time. Thus, we have the following lemma:

Theorem 5 *GROIF can be solved in* $O(N\log^{h+2} N)$ *time and* $O(N\log^{h+1} N)$ *space.*

4 Optimal region finding problem

Here, we do not assume the data size n to be $O(N)$; On the contrary, n can be as large as N^2. In his "Programming Pearls", Bentley posed a two-dimensional extension of Problem PP, which is, "compute a rectangle region R of the grid G so that $\sum_{\mathbf{KEY}(p)\in R} v(p)$ is maximized". Unfortunately, the current best algorithm for solving this problem needs $O(N^3)$ time, which is $O(n^{1.5})$ even if $n = N^2$. Our ORF problem is another way of extending Problem PP to a two-dimensional problem.

Theorem 6 *ORF can be solved in* $O(n)$ *time.*

Proof: An $O(N^2)$ time solution is given in [3] (as a subroutine of an image segmentation algorithm), and it is easy to modify it to be $O(n)$ time. □

参考文献

[1] A. Aggarwal, M.M. Klawe, S. Moran, P.W. Shor, and R. Wilber: "Geometric applications of a matrix-searching algorithm," *Algorithmica*, Vol.2, pp.195-208, 1987.

[2] M. J. Atallah and S. R. Kosaraju, "An Efficient Parallel ALgorithm for the Row Minima of a Totally Monotone Matrix" *J. Algorithms* 13 (1992) 393-413.

[3] Te. Asano, D.-Z. Chen, N. Katoh, and T. Tokuyama, "Polynomial-Time Solutions to Image Segmentation", Preprint, 1995.

[4] Jon Bentley, "Programming Pearls - Algorithm Design Techniques" *Communications of ACM* 27-9 (1984), 865-871.

[5] H. T. Kung, F. Luccio, and F. P. Preparata, "On Finding the Maxima of a Set of Vectors", *JACM* 22-4 (1975), 469-476.

[6] T. Fukuda, M. Morimoto, S. Morishita, T. Tokuyama "Mining Association Rules between Numerical and Boolean Attributes" Working paper, 1995.

[7] F. P. Preparata and M. I. Shamos, *Computational Geometry, an Introduction*, 2nd edition (1988)

[8] D. E. Willard, "New Data Structures for Othogonal Range Queries", *SIAM J. Comp.* vol 14 (1985) 232-253.