IEICE TRANS. INF. & SYST., VOL.E83–D, NO.3 MARCH 2000

INVITED SURVEY PAPER *Special Issue on Algorithm Engineering: Surveys*

Finding an Optimal Region in One- and Two-Dimensional Arrays

Naoki KATOH[†], ***Member***

SUMMARY Given N real weights $w_1, w_2, \ldots, w_N$ stored in one-dimensional array, we consider the problem for finding an optimal interval $I \subset [1, N]$ under certain criteria. We shall review efficient algorithms developed for solving such problems under several optimality criteria. This problem can be naturally extended to two-dimensional case. Namely, given a $N \times N$ two-dimensional array of N^2 reals, the problem seeks to find a subregion of the array (e.g., rectangular subarray R) that optimizes a certain objective function. We shall also review several algorithms for such problems. We shall also mention applications of these problems to region segmentation in image processing and to data mining.

key words: *optimal interval, combinatorial optimization, inter-class variance, image segmentation, data mining*

1. Introduction

In 1984, Bentley wrote a seminal paper [6] in which he posed the following problem: Given a one-dimensional array of N reals, how can we design an efficient algorithm for finding an interval I with $I \in [1, N]$ such that the sum of weights in I is maximum? At first glance, it looks difficult to design an $o(N^2)$ time algorithm without knowing algorithmic techniques. After giving a trivial $O(N^2)$ algorithm that examines all intervals, he explained an $O(N \log N)$ time algorithm using *divide and conquer* technique. Furthermore, he explained an improved $O(N)$ time algorithm (Kadena's algorithm) based on *dynamic programming*. The paper of [6] was written in order to demonstrate that an algorithmic view of a problem gives insights that may make a program simpler to understand and to write, and that the algorithmic techniques help to dramatically decrease the running time to solve the problem. As was written in [6], the problem arose from pattern recognition, but the original problem was two-dimensional. Therefore, the problem did not seem to have any practical importance except the above-mentioned educational purpose.

The original two-dimensional problem arising from pattern recognition is described as follows. Given a two-dimensional $N \times N$ array of reals, we want to find the rectangular subarray such that the sum of components is maximum. How do you design an efficient algorithm for this problem? Is there any $o(N^3)$ algorithm? Very recently, this was affirmatively solved by Tamaki and Tokuyama [17].

Problems related to the above one- and two-dimensional problems have recently been extensively studied and efficient algorithms for such problems have been developed by combining several nice algorithmic techniques. Such algorithms have been implemented and have been used in software for data mining. Data mining is the process of discovering interesting rules (or patterns) from a huge amount of data accumulated in databases. It has gained a rapidly growing interest in both AI and database communities. We think that the topics dealt with here are good examples in which algorithmic techniques greatly help to solve real-world problems.

This paper surveys recent development of the algorithms for finding an optimal region in one- and two-dimensional arrays and how they are used in related application fields.

In order to illustrate how data mining problems are formulated in terms of the above problems in a one-dimensional array, suppose that we have a database consisting of a numeric attribute A and a binary attribute B. Assume that the domains of A and B are $\{1, 2, \ldots, N\}$ and $\{yes, no\}$, respectively. A is called a *conditional* attribute while B is a *target* attribute. Then, a numeric one-dimensional rule is of the form

$$t[A] \in [u, v] \Rightarrow t[B] = \text{'yes'}$$

which expresses the fact that if the attribute A of tuple t takes the value in the interval $[u, v]$, the attribute B takes value 'yes' with high probability. For instance, if A represents the numeric attribute "Balance" and B the boolean attribute "Card-Loan" ('yes' or 'no'), this rule says that customers whose balance falls in the range between u and v are likely to use card loans. The number of tuples whose A value falls in the range $[u, v]$ is called the *support* of the rule, and among such tuples, the number of those with $t[B]$ = 'yes' is called the *hit* of the rule. The ratio of the hit to the support is called *confidence*. In data mining, one is often interested in finding a rule (called *association rule*) with large support and high confidence. Let s_i be the number of tuples such that $t[A] = i$, and among such tuples let h_i be the one such that $t[B]$ = 'yes'. For an interval $I \subset \{1, 2, \ldots, N\}$, let $s(I) = \sum_{1 \in I} s_i$ and

Manuscript received July 2, 1999.
Manuscript revised October 24, 1999.

[†]The author is with the Department of Architecture and Architectural Systems, Graduate School of Engineering, Kyoto University, Kyoto-shi, 606–8501 Japan.

$h(I) = \sum_{1 \in I} h_i$. In data mining, people are often interested in finding an interval I that optimizes one of the following objective functions.

(1) Maximize the confidence of I under the constraint that the support of I (i.e., $s(I)$) is not less than a given threshold.
(2) Maximize the support of I under the constraint that the confidence of I (i.e., $h(I)$) is not less than a given threshold.
(3) Maximize the *gain* of I, i.e, $gain_\theta(I) \equiv h(I) - \theta s(I)$, where θ is a given parameter.

We do not discuss the first and second problems in this paper (e.g., see [12]). The third problem is equivalent to the one studied by Bentley [6].

An important role of such an association rule in data mining is to segment tuples (e.g., customers) into two groups, one such that the confidence is high and the other is low. For this purpose, associations are recursively sought to derive a compact and reliable decision tree [15], [16]. Measuring the goodness of a segmentation is usually based on (i) the entropy gain or (ii) the interclass variance. For an interval $J \subset \{1, 2, \ldots, N\}$, Defining the entropy for an interval J as

$$Ent(J) = -\frac{h(J)}{s(J)} \log \frac{h(J)}{s(J)} - \frac{\bar{h}(J)}{s(J)} \log \frac{\bar{h}(J)}{s(J)},$$

where $\bar{h}(J) = s(J) - h(J)$, the entropy gain for a binary segmentation of $\{1, 2, \ldots, N\}$ into I and $\bar{I}(= \{1, 2, \ldots, N\} - I)$ is defined as follows.

$$\text{Ent\_Gain}(I, \bar{I}) = Ent([1, N]) - \frac{s(I)}{s(I) + s(\bar{I})} Ent(I) - \frac{s(\bar{I})}{s(I) + s(\bar{I})} Ent(\bar{I}). \quad (1)$$

Interclass variance of a segmentation $(I, \bar{I})$ is defined by

$$\text{Var}(I, \bar{I}) = s(I)s(\bar{I}) \cdot \left(\frac{h(I)}{s(I)} - \frac{h(\bar{I})}{s(\bar{I})} \right)^2. \quad (2)$$

We want to find an interval I that maximizes $\text{Ent\_Gain}(I, \bar{I})$ or $\text{Var}(I, \bar{I})$.

As will be seen later, this problem can be solved by using the problem studied by Bentley [6] as a subproblem. Other than the above two criteria, *Gini index* and *χ-squared error* have also been considered.

In this paper, we first review the algorithms by Bentley [6] and consider the following problem which is a simpler version of the one for maximizing (2). We shall explain an $O(N \log N)$ algorithm by Asano et al. [5]. We also explain the algorithms for maximizing (1) and (2).

We then focus on the two-dimensional problem. In this case, the relation to pattern recognition is clear. Suppose that we are given an $N \times N$ digital image, and let g_{ij} be the brightness level of a pixel (i, j) of an image. We want to separate an object from the background. This operation is commonly called *image segmentation*. Asano et al. [4], formulated the problem as the one for finding an optimal region in a two-dimensional array (i.e., a square grid of $N \times N$ in this case) by imposing the constraint that the shape of the object to be extracted is x-monotone. Here, a region S in the grid is said to be *x-monotone* if it is bounded by two x-monotone curves (see Fig. 2). Asano et al. [4] developed an $O(n^2)$ algorithm for finding a connected x-monotone region S_0 that maximizes the interclass variance between S_0 and S_1, where S_1 is the complement of S_0. We shall briefly review their algorithm. This problem is viewed as a variant of the two-dimensional version of the problem studied by Bentley.

In data mining, the problems of finding optimized two-dimensional association rules have recently been studied [10]–[12], [14], [18] and they are closely related in some sense to the above image segmentation problem. However, in the field of data mining, the objective functions such as maximization of entropy gain, χ^2-error and Gini index are often used and classes of shapes of regions other than x-monotone regions have also been considered. Depending on the cases, efficient algorithms have been developed [10]–[12], [18]. We also review such recent developments.

The organization of this paper is as follows. In Sect. 2, we consider the one-dimensional case. We first briefly review the algorithms of [6], and review an $O(N \log N)$ time algorithm for the problem of finding an interval in a one-dimensional array that maximizes the interclass variance. We also discuss the problems related to data mining. In Sect. 3, we consider the two-dimensional case. We review several algorithms depending on the cases of objective functions and the object shapes to be extracted. Section 4 concludes the paper.

2. One-Dimensional Case

2.1 Bentley's Problem

We consider the problem studied by Bentley [6]: Suppose that we are given a one-dimensional array of N reals $w_1, w_2, \ldots, w_N$. We want to find a subarray whose sum of weights is maximum.

$$P_1 : \text{maximize } \{w(I) \mid I \subset \{1, 2, \ldots, N\}\}, \quad (3)$$

where $w(I) = \sum_{i \in I} w_i$. We shall explain $O(N \log N)$ and $O(N)$ algorithms for this problem.

(a) **$O(N \log N)$ algorithm:** The idea is based on *divide and conquer*. We consider the following three subproblems. Let W^* be the optimal objective value.

$$P_{left} : \max\{w(I) \mid I \subset [1, N/2]\}. \quad (4)$$
$$P_{right} : \max\{w(I) \mid I \subset [N/2 + 1, N]\}. \quad (5)$$

$$P_{overlap} : \max\{w(I) \mid I \supset [N/2, N/2+1]\}. \quad (6)$$

Let W^*_{left}, W^*_{right} and $W^*_{overlap}$ denote the optimal objective values of problems P_{left}, P_{right} and $P_{overlap}$, respectively. We then have the following recurrence equation:

$$W^* = \max\{W^*_{left}, W^*_{right}, W^*_{overlap}\}. \quad (7)$$

The first two subproblems are of size $N/2$ and are recursively solved by further considering smaller subproblems. The last problem can be solved in $O(N)$ time by computing

$$w([i^*, N/2]) = \max_{1 \le i \le N/2} w([i, N/2]),$$
$$w([N/2+1, j^*]) = \max_{N/2+1 \le j \le N} w([N/2+1, j]).$$

Then an optimal solution of $P_{overlap}$ is $[i^*, j^*]$ and its objective value $W^*_{overlap}$ is $w([i^*, j^*])$. Let $T(N)$ denote the running time required to solve the original problem. We then have the following recursive formula.

$$T(N) = 2T(N/2) + O(N).$$

From this equation, we have $T(N) = O(N \log N)$.

(b) **$O(N)$ algorithm:** The algorithm is based on *dynamic programming* and scans the data in an array from left to right while maintaining the following information for each i with $1 \le i \le N$.

$$Max(i) = \max_{s \le i} w([s, i]), \quad \text{and}$$
$$Max(\le i) = \max_{s \le t \le i} w([s, t]).$$

When $i = N$, $Max(\le N)$ gives the solution. $Max(i)$ and $Max(\le i)$ are updated by using the following recurrence equation:

$$Max(i+1) = \max\{0, Max(i)\} + w_{i+1}, \quad \text{and}$$
$$Max(\le i+1) = \max\{Max(i+1), Max(\le i)\}.$$

It is clear that this algorithm correctly solves the problem in $O(N)$ time.

2.2 Maximizing Interclass Variance

We consider the problem of finding an interval I that maximizes the interclass variance. Let μ_0 and μ_1 denote the average of weights in I and $[1, N] - I$, respectively, i.e.,

$$\mu = \frac{w([1,N])}{N}, \mu_0 = \frac{w(I)}{|I|}, \mu_1 = \frac{w([1,N]) - w(I)}{N - |I|}.$$

We assume $\mu_0 > \mu$ without loss of generality. We consider the following problem:

$$P_2 : \quad \text{maximize} \quad |I|(\mu - \mu_0)^2 + (N - |I|)(\mu - \mu_1)^2$$
$$\text{subject to} \quad I \subset [1, N]. \quad (8)$$

Since there are $O(N^2)$ different solutions, it is easy to construct $O(N^2)$ time algorithm. We shall show an improved $O(N \log N)$ time algorithm for this. One of the techniques we use is dynamic programming we explained above, and the other is *parametric programming*. We shall explain how problem P_2 can be reduced to the following parametric programming problem.

$$P_\theta([1,N]) : \quad \text{maximize} \quad S_\theta(I) = \sum_{i \in I}(w_i - \theta)$$
$$\text{subject to} \quad I \subseteq [1, N] \quad (9)$$

Theorem 2.1 [4], [13]**:** There exists a parameter θ^* such that an optimal solution of $P_{\theta^*}([1, N])$ is also optimal to P_2.

Outline of the Proof: Let us assume $\mu = 0$ without loss of generality because translating each w_i by a constant does not change the objective value. From this assumption, the objective function of P_2 can be rewritten as

$$\frac{N(w(I))^2}{|I|(N - |I|)}, \quad (10)$$

(see [4] for details). Since $w(I) > 0$ from $\mu_0 > \mu$, and N is constant, maximizing (10) is equivalent to maximizing

$$U(I) = (|I|(N - |I|))^{-1/2} w(I) \quad (11)$$

Since $f(x, y) = y/\sqrt{x(N-x)}$ is convex in x and y with $0 < x < N$, maximizing (11) reduces to solving $P_\theta([1, N])$ for some θ by applying the following theorem developed by Katoh and Ibaraki [13], proving the theorem. □

Theorem 2.2 [13]**:** Suppose that a function $f(x, y) : R^2 \to R$ defined over an appropriate region $S \in R^2$ is convex, and continuously differentiable in x and y. (1) If a maximizer (x^*, y^*) of $f(x, y)$ over $(x, y) \in S$ satisfies $\partial f(x^*, y^*)/\partial x > 0$, it can be characterized by the following parametric problem:

$$\text{maximize } \{x + \theta y \mid (x, y) \in S\}, \quad (12)$$

where θ is a real parameter. (2) If $\partial f(x^*, y^*)/\partial x < 0$, (x^*, y^*) can be characterized by the following parametric problem:

$$\text{maximize } \{-x + \theta y \mid (x, y) \in S\}, \quad (13)$$

In other words, there exists θ^* for which an optimal solution of the above parametric problem is also a maximizer of $f(x, y)$.

Theorem 2.1 can be interpreted as follows. Let us consider the point set $\{(|I|, w(I)) \mid I \subset [1, N]\}$ in the plane (black dots in Fig. 1). The theorem says that an optimal solution of P_2 is on the upper chain of convex

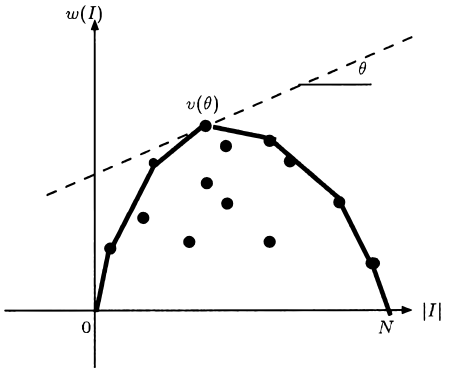

Fig. 1 Illustration of Theorem 2.1.

hull of the point set. We need an algorithm for solving $P_\theta([1,N])$ for a given slope θ without any knowledge of the locations of the point set. Such an algorithm computes the tangent line with slope θ to the upper chain of convex hull together with the tangent point $v(\theta)$, and is called a *touching oracle* [4]. The following general theorem is useful for efficiently solving problems dealt with in this paper.

Theorem 2.3 [8], [9]**:** Let P be a point set in the plane, $conv(P)$ be the convex hull of P, and $V(P)$ be the vertices on the upper chain of $conv(P)$. Then $V(P)$ can be computed by calling a touching oracle $O(|V(P)|)$ times.

Since problem $P_\theta([1,N])$ is equivalent to P_1, we can employ $O(N)$ or $O(N \log N)$ time algorithm as touching oracle. Since there are $O(N)$ different values of $|I|$ among $w(I) - \theta|I|$ for all $I \subset [1,N]$, $|V(P)| = O(N)$ holds. Therefore, from Theorem 2.3, we can obtain an $O(N^2)$ time algorithm for P_2. However, it is a trivial task to design an $O(N^2)$ time algorithm that checks all intervals. In order to improve the running time, we take the following different approach. We apply the divide and conquer technique used for P_1, and we simultaneously maintain a set of solutions of $P_\theta([1,N])$ for the entire range of θ. As in Sect. 2.1, we consider the problems $P_{left}(\theta)$, $P_{right}(\theta)$ and $P_{overlap}(\theta)$:

$$P_{left}(\theta) : \max\{w(I) - \theta|I| \mid I \subset [1, N/2]\},$$
$$P_{right}(\theta) : \max\{w(I) - \theta|I| \mid I \subset [N/2+1, N]\},$$
$$P_{overlap}(\theta) : \max\{w(I) - \theta|I| \mid I \supset [N/2, N/2+1]\}.$$

Let $W^*_{left}(\theta)$, $W^*_{right}(\theta)$ and $W^*_{overlap}(\theta)$ denote the optimal values of $P_{left}(\theta)$, $P_{right}(\theta)$ and $P_{overlap}(\theta)$, respectively. An optimal objective value of $P_\theta([1,N])$ is denoted by $W^*(\theta)$. Similarly to (7), we have the following recurrence equation.

$$W^*(\theta) = \max\{W^*_{left}(\theta), W^*_{right}(\theta), W^*_{overlap}(\theta)\}. \tag{14}$$

$W^*_{left}(\theta)$ is a piecewise-linear convex function in θ and has at most $N/2$ breakpoints because the size of interval $I \subset [1, N/2]$ is at most $N/2$ and hence there exist at most $N/2$ different slopes. The same remark holds for $W^*_{right}(\theta)$ and $W^*_{overlap}(\theta)$. Supposing that three functions of $W^*_{left}(\theta)$, $W^*_{right}(\theta)$ and $W^*_{overlap}(\theta)$ are known, we can determine in $O(N)$ time the function $W^*(\theta)$ from (14) in the same manner as list-merge algorithm if we represent breakpoints of each function as a list structure. What remains is to show that $W^*_{overlap}(\theta)$ can be computed in linear time. For this, let $w_\theta(I) = w(I) - \theta|I|$. We compute

$$f_{left}(\theta) = \max_{1 \le i \le N/2} w_\theta([i, N/2]),$$
$$f_{right}(\theta) = \max_{N/2+1 \le j \le N} w_\theta([N/2+1, j]).$$

Computation of $f_{left}(\theta)$ is done as follows. We first apply point-line duality mapping to $N/2$ lines corresponding to $w_\theta([i, N/2])$ with $1 \le i \le N/2$. The problem then reduces to finding a convex hull of points where $x-$coordinates of points are already sorted. Thus, we can compute $f_{left}(\theta)$ in $O(N)$ time. Similarly, $f_{right}(\theta)$ can be computed in $O(N)$ time. Since

$$W^*_{overlap}(\theta) = f_{left}(\theta) + f_{right}(\theta),$$

we can compute $W^*_{overlap}(\theta)$ in $O(N)$ time.

2.3 Data Mining Applications

As mentioned in Sect. 1, when we want to find a good segmentation in data mining applications, we consider the problem of maximizing $\mathrm{Ent\_Gain}(I, \bar{I})$ of (1) or $\mathrm{Var}(I, \bar{I})$ of (2), which can be viewed as the generalized version of the one studied in the previous subsection (the case of $s_i = 1$ corresponds to the one studied in the previous subsection). We first consider the case of maximizing $\mathrm{Var}(\cdot,\cdot)$. Let M be the number of tuples, and $H = \sum_{i=1}^{N} h_i$. Let

$$\mu = \frac{H}{M}, \mu_0 = \frac{h(I)}{s(I)}, \mu_1 = \frac{H - h(I)}{M - s(I)}.$$

Letting $h_i = h_i - \mu$ does not change the value of $\mathrm{Var}(I, \bar{I})$, we can rewrite the objective function as

$$\text{maximize } \frac{h(I)}{\sqrt{s(I)s(\bar{I})}}. \tag{15}$$

Therefore, since the function $f(x,y) = y/\sqrt{x(M-x)}$ is convex in (x,y) for $0 < x < M$ and $y \ge 0$ as mentioned in Sect. 2.2, we can again apply Theorem 2.2. Namely, an optimal solution can be found by solving the following parametric problem

$$\text{maximize}_{I \subset [1,N]} s(I) - \theta \cdot h(I) \tag{16}$$

for the entire range of θ. The only difference lies in that the number of vertices on the convex hull of the point set $\{(s(I), h(I)) \mid I \subset [1, 2, \ldots, N]\}$ is not bounded by $O(N)$ but by $O(M)$. Although the details are omitted, we can obtain the following result.

Theorem 2.4: An optimal segmentation for M tuples based on interclass variance can be obtained in $O(M \log N)$ time.

For the case of maximizing Ent_Gain$(I, \bar{I})$, we first show that the optimal solution can also be characterized by parametric optimization. For this, let

$$f(x,y) = x \log \frac{x}{x+y} + y \log \frac{y}{x+y}.$$

It is easy to see that $f(x,y)$ for $x > 0$ and $y > 0$ is concave in (x,y). Therefore,

$$E(x,y) = -\frac{f(x,y) + f(s(I)-x, s(\bar{I})-y)}{M}$$

is convex for (x,y) with $0 < x < s(I)$ and $0 < y < s(\bar{I})$ (see [14]). Therefore, again from Theorem 2.2, it follows that there exists a parameter θ^* such that an optimal solution of the parametric problem (16) with $\theta = \theta^*$ maximizes the entropy gain.

Theorem 2.5: An optimal segmentation of M tuples that maximizes the entropy gain can be obtained in $O(M \log N)$ time.

3. Two-Dimensional Case

As mentioned in Sect. 1, problems in two-dimensional case are classified into several cases according to objective functions and the shape of regions to be extracted. We first consider the case where the objective function is to maximize the interclass variance and the shape is x-monotone region. This problem arises from image segmentation [4].

3.1 Finding an Optimal x-Monotone Region

We are given an $N \times N$ two-dimensional array, and Let w_{ij} be the weight of (i,j) component. In the application of image segmentation, w_{ij} represents the grey scale level at pixel (i,j) in an $N \times N$ digital image. We want to find a connected x-monotone region in the array that maximizes the interclass variance. Let S_0 be the region we want to find and S_1 be its complement. Note that the intersection of an x-monotone region and any pixel column is a (possibly empty) connected region. We call a region *admissible* if it is x-monotone and connected (Fig. 2).

The problem discussed here is formulated as follows. Let $n = N \times N$. Let $\mu = \frac{1}{n}\sum_{(i,j)\in G} w_{ij}$. Let $n_0 = |S_0|$, $n_1 = |S_1|$, $\mu_0 = \frac{1}{n_0}\sum_{(i,j)\in S_0} w_{ij}$, and $\mu_1 = \frac{1}{n_1}\sum_{(i,j)\in S_1} w_{ij}$. The objective function that we use is to

$$\text{maximize } V(S_0, S_1) = n_0(\mu - \mu_0)^2 + n_1(\mu - \mu_1)^2.$$

As in Sect. 2.2, we assume $\mu = 0$ without loss of generality. We define

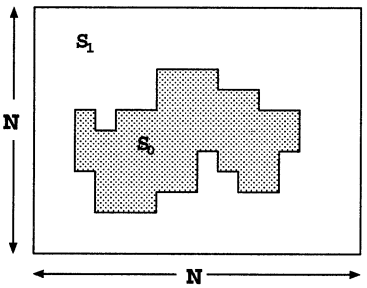

Fig. 2 An admissible region bounded by two x-monotone curves.

$$w(S_0) = \sum_{(i,j)\in S_0} w_{ij}.$$

Similarly to the discussion of Sect. 2.2, we rewrite the objective function as

$$\text{maximize } D(S_0, S_1) \equiv (n_0 n_1)^{-1/2} w(S_0).$$

Let $P(k)$ denote the subproblem of maximizing $D(S_0, S_1)$ under the constraint that n_0 is fixed to k. When k is fixed, the term $(n_0 n_1)^{-1/2}$ becomes constant. Thus, $P(k)$ can be formulated as:

> Find an admissible object S_0 maximizing $w(S_0)$ under the condition that $|S_0| = k$.

We define $F(k)$ to be the maximum value of $w(S_0)$ with $|S_0| = k$. We first consider the segmentation by using one x-monotone chain into a connected region S_0 and its complement S_1. Without loss of generality, we assume that S_0 is below the separating chain.

We define M_m to be the m-th column of the input matrix, and $M_{\le m} = \cup_{i \le m} M_i$. Let $F(k,m)$ be the maximum of $w(S_0)$ under the conditions that $|S_0| = k$, $S_0 \subseteq M_{\le m}$, and $S_0 \cap M_m \neq \emptyset$. Naturally, $F(k) = \max_{1 \le m \le N} F(k,m)$. Thus, it suffices to compute $F(k,m)$ for all k and m.

For each $m = 1, 2, \ldots, N$ and for any interval $I \subset [1,N]$, we define $f_m(I) = \sum_{i \in I} w_{im}$. Then, we have the following recursion formula of $F(k,m)$.

$$F(k,m) = \max_{1 \le t \le \max\{N,k\}} \{F(k-t, m-1) + f_m([1,t])\}$$

This formula naturally leads us to a simple dynamic programming algorithm. Since $m \le N$, $t \le N$, and $k \le n$, the time complexity of the dynamic programming is $O(N^2 n)$, which is $O(n^2)$. Thus, we have the following theorem.

Theorem 3.1: Given an $N \times N$ matrix of n reals, an optimal partition with one x-monotone chain can be computed in $O(n^2)$ time.

The dynamic programming developed here can be extended to the case where S_0 is bounded by two x-monotone chains. However, the running time becomes

$O(N^7) = O(n^{3.5})$ time as shown in [4].

Since this running time is enormous, we take a different approach based on parametric optimization as explained in Sect. 2.2. In fact, Theorems 2.1 and 2.3 can be extended to two-dimensional case in a straightforward manner. Letting

$$\begin{aligned} w_\theta(S_0) &= w(S_0) - \theta|S_0| \\ &= \sum_{(i,j)\in S_0} (w_{ij} - \theta), \end{aligned}$$

we consider the following parametric problem.

$Q(\theta)$: Compute an admissible region S_0 which maximizes $w_\theta(S_0)$.

The following is a key lemma:

Lemma 3.1: $Q(\theta)$ can be solved in $O(n)$ time using $O(N) = O(\sqrt{n})$ working space.

Proof. We define $\tilde{w}_{ij} = w_{ij} - \theta$. We consider the m-th column M_m of the matrix M. For a closed interval $I = [u, v]$ of row indices, we define $\tilde{f}_m(I) = \sum_{s\in I} \tilde{w}_{sm}$.

We compute and store both $\tilde{f}_m([1, v])$ and $\tilde{f}_m([u, N])$ for all $1 \le v$, $u \le N$ in $O(N)$ time using $O(N)$ space. Since $\tilde{f}_m([u, v]) = \tilde{f}_m([1, N]) - \tilde{f}_m([1, u-1]) - \tilde{f}_m([v+1, N])$ for $1 < u \le v < N$, we can query $\tilde{f}_m(I)$ in $O(1)$ time for each interval $I = [u, v]$.

Let $s(i) = \max_{u\le i} \tilde{f}_m([u, i])$ and let $t(j) = \max_{v\ge j} \tilde{f}_m([j, v])$. Computing $s(i)$ (resp., $t(i)$) is equivalent to computing the positions of the maximum entry in the i-th column (resp., i-th row) of the upper triangle matrix $\mathcal{F}_m = (\tilde{f}_m([u, v]))_{1\le u\le v\le N}$.

The matrix $\mathcal{F}_m$ is a *Monge* matrix, that is, for $i < i' < j' < j$, $\tilde{f}_m([i, j]) + \tilde{f}_m([i', j']) \ge \tilde{f}_m([i, j']) + \tilde{f}_m([i', j])$. (Moreover, equality holds, and hence the matrix $\mathcal{F}_m$ is a *strong Monge* matrix.) We compute and store $s(i)$ and $t(i)$ for all $i = 1, 2, \ldots, N$ in $O(N)$ time and space by using a fast Matrix Searching algorithm on a Monge matrix [1].

For each pair (i, j) of row indices, we define $cover_m(i, j) = \max_{J\ni i,j} \tilde{f}_m(J)$. Then, it is easy to see that $cover_m(i, j) = \tilde{f}_m([s(i), t(j)])$ if $i \le j$. Thus, we can query $cover_m(i, j)$ for each pair (i, j) in $O(1)$ time.

We define $w_\theta(m, i)$ as the maximum value of $w_\theta(S')$ among all admissible regions $S' \subseteq G_{\le m}$ that contain the pixel (i, m). We set $w_\theta(0, i) = 0$ for all i. Obviously, $\max_{S_0} w_\theta(S_0) = \max_{0\le m\le N}\{\max_{1\le i\le N} w_\theta(m, i)\}$.

Below, we compute $w_\theta(m, i)$ for all $0 \le m \le N$ and $1 \le i \le N$ in $O(N^2)$ time. From the definition of admissible regions, the following recursion formula holds:

$$\begin{aligned} &w_\theta(m, i) \\ &\quad = \max_{1\le j\le N} \{\max\{0, w_\theta(m-1, j)\} + cover_m(i, j)\}. \end{aligned}$$

We compute $w_\theta(m, i)$ for all i by using the above formula. It is not difficult to see that the matrix $(cover_m(i, j))_{1\le i\le j\le N}$ is a strong Monge matrix. Accordingly, the matrix $\mathcal{D}_m$ defined by $\mathcal{D}_m(i, j) = w_\theta(m-1, j) + cover_m(i, j)$ is also a strong Monge matrix. (To say precisely, its upper triangular part is a strong Monge matrix, and also its lower triangular part is the transposition of a strong Monge matrix.) The computation of the above formula for a fixed m is equivalent to computing all row maxima of these two matrices, which can be done with $O(N)$ queries of $cover_m(i, j)$ by using a fast Matrix Searching algorithm [1].

Therefore, the computation of $\max_{S_0} w_\theta(S_0)$ altogether needs $O(N^2) = O(n)$ queries each of which takes $O(1)$ time, and hence the total time complexity is $O(n)$. The space complexity is obviously $O(N) = O(\sqrt{n})$. □

Theorem 3.2: The image segmentation problem can be solved in $O(n^2)$ time and $O(n)$ working space.

Proof. Since there are $O(n)$ different sizes of S_0, the convex hull of the point set of $\{(|S_0|, w(S_0)) \mid S_0$ is an admissible region$\}$ has $O(n)$ vertices. Therefore, from Lemma 3.1 and Theorem 2.3, the theorem follows. □

An $O(n^2)$ time of Theorem 3.1 for the case where S_0 and S_1 are separated by a single x-monotone chain is further improved to $O(n^{1.5}) = O(N^3)$ time by elaborating the algorithm explained above. In fact, by spending $O(n)$ time for preprocessing using the data structure of convex hull and fractional cascading of [7], it was shown that the optimal interval in each column can be obtained in $O(N)$ time for any given θ (see [14]).

Figure 3 shows an experimental result of the above algorithm. The upper figure is an original image and the lower one is the result of the segmentation. Although the original image is an color image, we consider it as an monochrome image whose gray scale level is obtained by adding the values of R, G and B.

3.2 Rectangular Region

In this subsection, we consider the problem of finding a rectangular subarray in a given $N \times N$ two-dimensional array A. We first consider the objective function that maximizes the sum of weights. The existence of $o(N^3)$ algorithm has been open for a long time, but recently Tamaki and Tokuyama [17] have settled this issue. Their algorithm is based on *funny matrix multiplication*, where a scalar product and addition in usual matrix multiplication are replaced by addition and max operations. Its running time is $O(N^3(\frac{\log\log N}{\log N})^{1/2}))$. They also proposed a faster ϵ-approximation algorithm. The details are omitted here.

Let us consider the objective function that maximizes the interclass variance. We briefly explain

an $O(N^3)$ time algorithm developed in [5]. The idea is divide-and-conquer and the reduction to one-dimensional problem. The algorithm consists of the following three steps:

Step 1: Decompose the original $N \times N$ matrix into two $N/2 \times N$ submatrices. We find a rectangular submatrix maximizing the interclass variance such that it overlaps both the upper and lower submatrices (see Fig. 4 (a)). For this, let us focus on rectangular submatrices such that their smallest and largest column indices are j_1 and j_2, respectively. Finding the one maximizing the interclass variance among these submatrices can be reduced to the case of the one-dimensional array whose i-th element has weight

$$W_i = \sum_{i=j_1}^{j_2} w_{ij}.$$

This can be solved in $O(N \log N)$ time. Since we have to examine all possible pairs of j_1 and j_2, Step 1 requires $O(N^3 \log N)$ time in total.

Step 2: We further decompose each $N/2 \times N$ submatrix into two $N/2 \times N/2$ square submatrices. In each $N/2 \times N$ submatrix, we consider submatrices that overlap the dividing perpendicular line, and among these submatrices we compute the one maximizing the interclass variance (see Fig. 4 (b)). This computation can be done in the same fashion as in Step 1.

Step 3: For each of $N/2 \times N/2$ square submatrices, we recursively apply the above two steps (see Fig. 4 (c)).

Letting $T(N)$ denote the running time of the above algorithm, we have the following recurrence equation:

$$T(N) = 4T(N/2) + O(N^3 \log N).$$

Solving this, we get $T(N) = O(N^3 \log N)$. However, closely examining the time required for Steps 1 and 2, we can carry out these steps in $O(N^3)$ time. The reason is as follows. Recall that $\log N$ term of $O(N \log N)$ running time for one-dimensional problem comes from $O(\log N)$ recursion levels. At each level, the required time is only $O(N)$. Thus, in this problem, the work required for lower recursion levels can be taken care of at the lower recursion levels of the two-dimensional

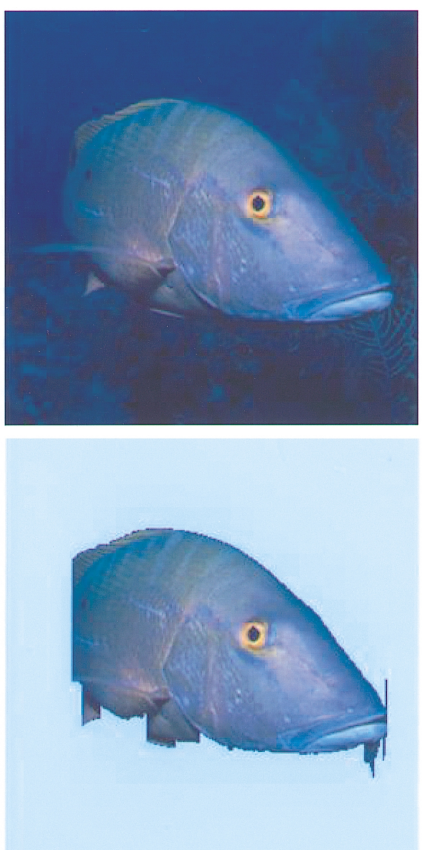

Fig. 3 Experimental result: a fish.

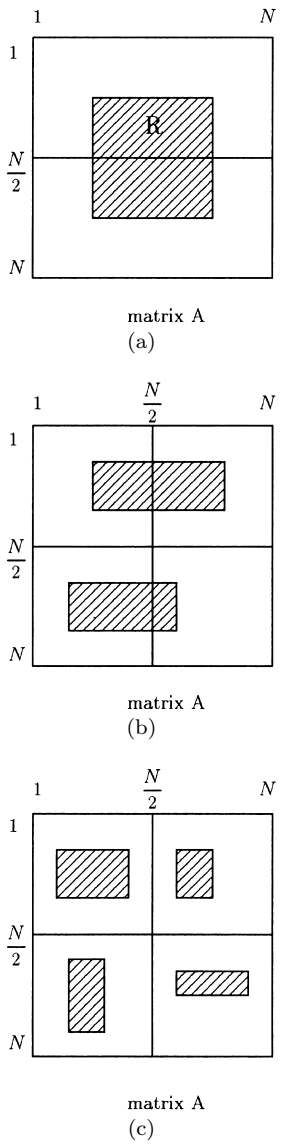

Fig. 4 Illustration of the algorithm in Sect. 3.2.

divide and conquer carried out in this algorithm. Thus, it follows that Steps 1 and 2 can be done in $O(N^3)$ time. Therefore, we have the following improved recurrence equation:

$$T(N) = 4T(N/2) + O(N^3).$$

Solving this, we get $T(N) = O(N^3)$.

Theorem 3.3: A rectangular subarray in an $N \times N$ two-dimensional array that maximizes the interclass variance can be found in $O(N^3)$ time.

3.3 Data Mining Applications

In data mining, binary associations between two attributes explained in Sect. 1 are not enough to describe the characteristics of a data set, and thus we are often asked to find a rule for more than two attributes. For this, two-dimensional association rules have been considered [10], [11], [14], [18], which represent the dependence between a pair of conditional numeric attributes and a target Boolean attribute. Let A and B be two numeric attributes and C be the target Boolean attribute. For the sake of simplicity, suppose that each domain of A and B is $\{1, 2, \ldots, N\}$. So, the entire domain of A and B can be viewed as two-dimensional array, or $N \times N$ grid in the plane. For each tuple t in a database, t is mapped to a component in the array. We want to find a rule of the form $((A, B) \in P) \Rightarrow C$. The shape of a region P is important for finding a useful association rule. If P consists of many connected components, it is hard to characterize the rule associated with the region. Therefore, P should have a simple form that has nice geometric properties.

In view of this, the following three classes of regions have been considered in the literature; rectangular, rectilinear and x-monotone regions. A connected region is called *rectilinear* if it is both x-monotone and y-monotone. The following results for the case of rectangular and x-monotone regions are obtained by generalizing those given in Sects. 2.3, 3.1 and 3.2. We assume here that there are M tuples in a database.

Theorem 3.4: An optimal rectangular region that maximizes the interclass variance or the entropy gain can be found in $O(nM)$ time, where $n = N^2$.

Theorem 3.5 [4], [11]**:** An optimal x-monotone region that maximizes the interclass variance or the entropy gain can be found in $O(nM)$ time.

Theorem 3.6 [18]**:** An optimal rectilinear region that maximizes the interclass variance or the entropy gain can be found in $O(n^{1.5}M)$ time.

4. Conclusion

We have reviewed several algorithmic issues that arise from problems for finding optimal regions in one- and two-dimensional arrays. We would like to further investigate the case of higher dimensions. The problems discussed in this paper have a close connection with image segmentation in computer vision and with association rules and decision trees in data mining. In this sense, developing efficient algorithms for these problems will directly have a significant impact on such application fields. On the other hand, those fields have many important and interesting research problems for researchers working in the fundamental area of algorithms. The interplay between these two different fields will be helpful to further development of *algorithm engineering*.

Acknowledgement

The author gratefully acknowledges several helpful discussions with Takeshi Tokuyama.

References

[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] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," Proc. ACM SIGMOD Conference on Management of Data, pp.207–216, 1995.
[3] T. Asano, "Image processing as discrete system," Discrete Structure and Algorithms, vol.5, ed. S. Fujishige, pp.101–146, Kindai-Kagakusha, 1998.
[4] T. Asano, D. Chen, N. Katoh, and T. Tokuyama, "Polynomial time solution to image segmentation," Proc. 7th ACM-SIAM Symposium on Discrete Algorithms, pp.104–113, 1996.
[5] T. Asano, N. Katoh, and T. Tokuyama, "An algorithm for finding an interval with maximum interclass variance and its extension to higher dimensions," Research Report, no.98-AL-65, SIGAL IPSJ, pp.1–8, 1998.
[6] J. Bentley, "Programming pearls – Algorithm design techniques," CACM, vol.27, no.9, pp.865–871, 1984.
[7] B. Chazelle and L. Guibas, "Fractional cascading: A data structure technique," Algorithmica, vol.1, pp.133–162, 1986.
[8] D. Dobkin, H. Edelsbrunner, and C.K. Yap, "Probing convex polytopes," Proc. 18th ACM STOC, pp.387–392, 1986.
[9] M.J. Eisner and D.G. Severence, "Mathematical techniques for efficient record segmentation," JACM, vol.23, pp.619–635, 1976.
[10] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Constructing efficient decision trees by using optimized association rules," Proc. 22nd VLDB Conference, pp.146–155, 1996.
[11] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization," Proc. ACM SIGMOD International Conference on Management of Data, pp.13–23, 1996.
[12] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Mining optimized association rules for numeric attributes," Proc. 15th ACM Principle of Database Systems, pp.182–191, 1996, Also J. Comput. Syst. Sci., vol.58, pp.1–12, 1999.

[13] N. Katoh and T. Ibaraki, "A parametric characterization and an ϵ-approximation scheme for the minimization of a quasiconcave program," Discrete Applied Mathematics, vol.17, pp.39–66, 1987.
[14] Y. Morimoto, T. Fukuda, S. Morishita, and T. Tokuyama, "Implementation and evaluation of decision trees with range and region splitting," Constraint, vol.2, nos.3/4, pp.163–189, 1997.
[15] J.R. Quinlan, "Induction of decision trees," Machine Learning, vol.1, pp.81–106, 1986.
[16] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.
[17] H. Tamaki and T. Tokuyama, "Algorithms for the maximum subarray problem based on matrix multiplication," Proc. 9th ACM-SIAM Symposium on Discrete Algorithms, pp.446–452, 1998.
[18] K. Yoda, T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Computing optimized rectilinear regions for association rules," Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining 1997 (KDD '97), pp.96–103, 1997.

Naoki Katoh received the B.Eng, M.Eng. and Dr.Eng. degrees in Applied Mathematics and Physics from Kyoto University in 1973, 1975 and 1981, respectively. He was an Assistant Professor during 1981–1982, an Associate Professor during 1982–1990, and a Professor during 1990-1997, respectively, at Department of Management Science of Kobe University of Commerce. In 1997 he joined Kyoto University where he is currently a Professor at Department of Architecture and Architectural Systems. His research interests include the design and analysis of combinatorial and geometric algorithms, data mining and architectural information systems. He is a member of IPSJ, OR Soc. Japan, Japan SIAM, and ACM.