UDC 519.8
MSC 90B85

Вестник СПбГУ. Прикладная математика. Информатика... 2019. Т. 15. Вып. 4

The problem of a maximal weighted area of axis-parallel rectangle that covers polygons

L. V. Shchegoleva[1], *R. V. Voronov*[1], *L. Sedov*[2]

[1] Petrozavodsk State University, 33, Lenina pr., Petrozavodsk, 185910, Russian Federation
[2] Linköping University, Department of Science and Technology, Campus Norrköping, 33, Bredgatan, 60221, Norrköping, Sweden

For citation: Shchegoleva L. V., Voronov R. V., Sedov L. The problem of a maximal weighted area of axis-parallel rectangle that covers polygons. *Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes*, 2019, vol. 15, iss. 4, pp. 592–602. https://doi.org/10.21638/11702/spbu10.2019.414

The paper presents the problem of finding the optimal location of the rectangle with the maximum weighted area. The dimensions of the rectangle are set, the sides of the rectangle are parallel to the axes. On the plane, there are non-self-intersecting polygons of arbitrary shape with a given density. The weighted area of a rectangle is calculated as a sum of the area of the parts of polygons covered by the rectangle multiplied by their densities. The algorithm for solving the problem is described. This problem arises when determining the places of forest felling when the planned cutting area can be modelled by a rectangle, and the polygons describe the areas with same forest taxation, for each of which is known forest stock per hectare.

Keywords: maximizing range sum (MaxRS), maximizing area-range sum, maximizing weighted area-range sum, polygons.

Introduction. The computational geometry have now received a new impetus in connection with the development of location and navigation technologies. One of classic problems is the problem of detecting a location where a fixed-size rectangle should be placed, so that it covers the maximum number of points.

In [1] the continuation of this problem for the case where the objective is to maximize the area covered by a rectangle on a set of convex non-intercepted polygons is presented.

This paper proposes a further development of this problem for the case, where polygons are not necessarily convex, can together cover the entire space under consideration (i. e. have no "holes"). And the goal is to maximize the weighted area covered by the rectangle.

This problem arises during the process of determining the places of forest felling, when the planned cutting area can be modelled by a rectangle, and the polygons describe the spots with the same forest taxations, tightly adjacent to each other. For each spot, a stock of wood per hectare is known, which can be considered equally distributed over the entire area of spot. A logger is interested in maximizing the volume of harvested wood. Therefore, the cutting area should be located so that the total volume of harvested wood on it is the maximum [2]. This corresponds to the maximum weighted area covered by the rectangle.

Now we describe the exact formulation of the problem.

A finite set of polygons of arbitrary shape and size (with the exception of self-intersecting polygons) $P = \{p_i\}_1^n$ is defined on the plane (Figure 1).

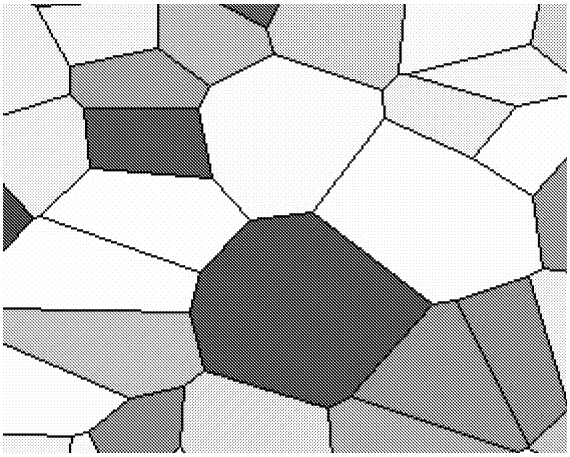

Figure 1. A plane covered with polygons with a specified density values indicated by a color saturation

Polygon p_i is defined by a set of points corresponding to the vertices of polygon $\{v_{ij} = (x_{ij}, y_{ij})\}, j = 1, ..., n_i$, where n_i is the number of vertices of polygon p_i. The sequence of points are ordered, the neighboring points define the edges of a polygon $a_{ij} = ((x_{ij}, y_{ij}), (x_{ij+1}, y_{ij+1})), j = 1, ..., n_i$, where $n_i + 1 = 1$, that is, the last point joins the first point to form the closing edge of the polygon.

Two polygons can share edges. The inner areas of the polygons do not intersect. Thus, the space covered by polygons may contain "holes".

Each point of the plane:

- either belongs to the inner area of only one polygon;
- or does not belong to any polygon;
- or lies on the edge of one or two polygons;
- or coincides with the vertex of one or more polygons.

It should be noted that a vertex of one polygon cannot lie on the edge of another polygon, it is either the vertex of another polygon or belongs to only one polygon.

Each polygon is characterized by a density (weight, expressed by some value per unit area) G_i. At the edges of the polygon we consider this value to be zero. At points that do not belong to any polygon this value considered to be equal to zero. In Figure 1 the density of a polygon is indicated by the color saturation of the polygon interior in grayscale. The darker the color, the higher the density G_i of the polygon p_i.

It is necessary to find the location of the rectangle R, namely the coordinates of the lower left corner of the rectangle (x_0, y_0), with sides parallel to the coordinate axes (Figure 2), and given sizes: w is length along the axis OX, h is length along the axis OY, so that the weighted area $S(R)$ covered by rectangle is maximal.

The weighted area is calculated as the sum of the parts of the polygon areas that are covered by the rectangle multiplied by the weights of the corresponding polygons:

$$S(R) = \sum_{p_i \cap R \neq \varnothing} G_i \cdot S(p_i \cap R),$$

where $S(\cdot)$ is a function of area.

In Figure 1 the rectangle should be placed so that it covers as dark areas as possible.

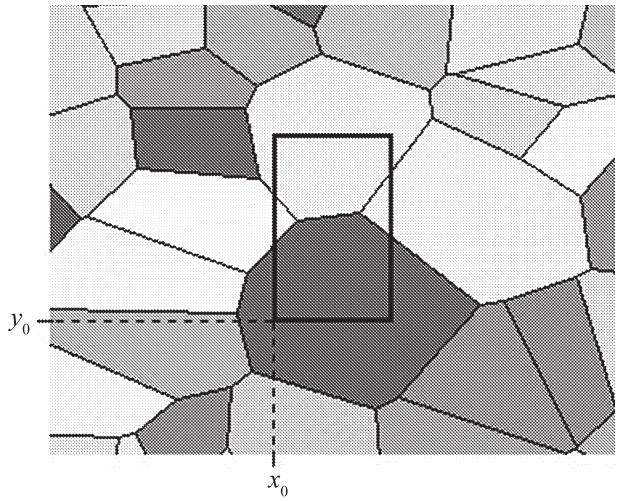

Figure 2. The location of the rectangle R

Related works. The described problem comes from the maximum subarray problem, which is formulated as follows. Given an $M \times N$ array of real numbers, find the maximum sum contained in any rectangular subarray. This problem was first posed by Grenander of Brown University in the two-dimensional space for pattern detection in images and then he simplified it to one dimension. In 1977 Kadane designed a linear time algorithm for this problem in one dimension. Obviously, there is a simple algorithm in $O((MN)^2)$ time for this problem in two dimensions. The history of this problem is described by Bentley in [3]. The maximum subarray problem has applications in pattern recognition [4] and data mining [5].

Tamaki and Tokuyama described a subcubic algorithm, by reducing this problem to "funny matrix multiplication" and also they developed an ε-approximation algorithm [6]. The scalar product and addition in matrix multiplication were replaced by addition and max operation in these algorithms.

Takaoka designed an algorithm, which solves the maximum subarray problem in $O(M^2N(\log\log M/\log M)^{1/2})$ time. Takaoka also gave a practical algorithm, whose expected time is better than the worst case time [7].

Daliels, Milenkovic and Roth considered the geometric optimization problem of finding the largest area axis-parallel rectangle of a fixed size in an n-vertex general polygon [8]. They considered different cases based on the types of contacts between the rectangle and the polygon. They obtained the following algorithm time results: $\Theta(n)$ for xy-monotone polygons, $O(n\alpha(n))$ for orthogonally convex polygons ($\alpha(n)$ is the inverse of Ackermann's function), $O(n\alpha(n)\log n)$ for horizontally (vertically) convex polygons, $O(n(\log n))$ for a special type of horizontally convex polygon, $O(n(\log^2 n))$ for general polygons, in which there may be holes. A lower bound of time $\Omega(n(\log n))$ is established for finding the largest area rectangle of a fixed size in both self-intersecting polygons and general polygons with holes. Also, a lower bound of $\Omega(n(\log n))$ and an upper bound of $O(n(\log^2 n))$ for general polygons were established.

Nandy and Bhattacharya proposed an unified algorithm for locating the position of the plate which encloses the maximum (or the minimum) number of objects [9]. Also they

extended their algorithm for identifying a cuboid, i. e., a rectangular parallelepiped that encloses the maximum number of polyhedral objects in $\mathbb{R}^3$.

Backurs, Dikkala and Tzamos studied max-weight rectangle in d dimensions. This problem asks to find an axis parallel rectangle that maximizes the total weight of the points it contains [10]. Additionally, they provide an overview of the algorithms for solving the maximum weight rectangles problem.

In their most recent paper [1], Hussian and Trajcevski investigated the problem of placing the rectangle so that the total area coverage on the input set of polygons is maximized. The proposed algorithm has a limitation: in an optimal solution at least one side of the rectangle must contain a vertex of some polygon. For the problem of a weighted area of rectangle, this limitation may lead to a non-optimal solution. In addition, when an area under consideration is completely filled with polygons (without "holes"), the problem of detecting a location of the rectangle with the maximum area degenerates, whereas the problem of the detecting a location of the rectangle with the maximum weighted area remains as a problem.

The main results. The general algorithm for solving the maximum weighted area problem includes the following steps:

1) cover the polygon area with a regular grid; calculate the upper bound of weighted area of each grid cell;

2) arrange the cells in descending order of the upper bound; choose the first cell from the ordered list;

3) place the rectangle R so that the upper left corner of the rectangle coincides the upper left corner of the current cell;

4) shift the rectangle to a local maximum of its weighted area $S(R)$ by the coordinate-wise descent method;

5) if the current solution is better than the previously obtained solutions, then remember it;

6) choose the next cell from the ordered list;

7) if the remembered value of the weighted area exceeds the upper bound of the current cell, then finish the algorithm, otherwise go to step 3.

Consider the first step of the algorithm.

We assume that the lengths of the sides w and h of the rectangle R are proportional to the given values d_x and d_y, i. e.

$$w = M \cdot d_x, \qquad h = N \cdot d_y,$$

here M and N are natural numbers. Let the region under consideration be divided by a regular grid into rectangular cells Υ_{ij} $(i = 1, \ldots, P,\ j = 1, \ldots, Q)$ with sides d_x and d_y.

Let A_{ij} denote the weighted area of cell Υ_{ij}:

$$A_{ij} = \int\limits_{\Upsilon_{ij}} G(z)dz,$$

where $G(z)$ is a density in the point z. We will assume that $A_{ij} = 0$ if $i < 1$, or $i > P$, or $j < 1$, or $j > Q$.

Let S_{ij} be the weighted area of the rectangle R_{ij} whose upper left corner coincides with the upper left corner of the cell Υ_{ij}:

$$S_{ij} = \sum_{p=i}^{i+M-1} \sum_{q=j}^{j+N-1} A_{pq}, \quad i = 1, \ldots, P - M + 1, \quad j = 1, \ldots, Q - N + 1.$$

Let Φ_{ij} $(i = 1, \ldots, P - M + 1,\ j = 1, \ldots, Q - N + 1)$ denote the region that includes rectangle R_{ij} and all the grid's cells around it:

$$\Phi_{ij} = \bigcup_{\substack{p=i-1,\ldots,i+M \\ q=j-1,\ldots,j+N}} \Upsilon_{pq}.$$

Let $G_{ij}^{\max}$ denote the maximum density of cells from $\Phi_{ij} \setminus R_{ij}$. They are cells with row and column numbers:

$$(i-1, q) \text{ and } (i+M, q), \qquad q = j-1, \ldots, j+N,$$

$$(p, j-1) \text{ and } (p, j+N), \qquad p = i, \ldots, i+M-1,$$

and $G_{ij}^{\min}$ denote the minimum density of cells with row and column numbers:

$$(i, q) \text{ and } (i+M-1, q), \qquad q = j, \ldots, j+N-1,$$

$$(p, j) \text{ and } (p, j+N-1), \qquad p = i+1, \ldots, i+M-2.$$

Then the upper bound B_{ij} for weighted area of any rectangle R located in region Υ_{ij} is calculated as

$$B_{ij} = S_{ij} + (G_{ij}^{\max} - G_{ij}^{\min}) \cdot (M + N - 1) \cdot d_x \cdot d_y. \tag{1}$$

Consider in more detail the 4th step of the algorithm.

The idea of the method of coordinate-wise descent is as follows. Step by step, the rectangle moves in one of four directions parallel to the coordinate axes (left, right, upward, downward). The value of the shift is determined based on the fact that the set of polygons intersecting the rectangle has to remain unchanged. The shift occurs at the maximum possible distance in the direction in which the increment of the weighted area is the maximum and positive. The algorithm stops, when the weighted area increment in all directions becomes negative or equal to zero.

The algorithm more strictly is described as follows:

1. Two regions Θ_h — a horizontal strip with a width of h, and Θ_v — a vertical strip with a width of w — are defined. Within these strips the rectangle can be shifted (Figure 3). Regions are formed by lines l_{right}, l_{right}, l_{top}, l_{bottom}, which are continuations of the sides of the rectangle R. On the line l_{left} lies the left side of the rectangle, on l_{right} lies the right side, on l_{top} lies the top side, on l_{bottom} lies the bottom side.

2. For horizontal, and vertical shifts, the sets of stopping points V_h, and V_v are defined (Figure 3). The stopping points are the points at which the sides of the rectangle should stop moving further in that direction. All stopping points are either at the vertices of polygons or lie on their edges.

3. For each of the four shift directions, the maximum shift value is determined based on the coordinates of the stopping points: ΔX_{left}, ΔX_{right}, ΔY_{up}, ΔY_{down} (Figure 4).

4. For each of the 4 shift directions, a weighted area's increment is calculated based on the maximum shift value: ΔS_{left}, ΔS_{right}, ΔS_{up}, ΔS_{down}.

5. The maximum value is selected from four increments $\Delta S_{\max} = \max\{\Delta S_{\text{left}}, \Delta S_{\text{right}}, \Delta S_{\text{up}}, \Delta S_{\text{down}}\}$.

6. If the maximum increment is greater than zero ($\Delta S_{\max} > 0$), the rectangle is shifted in the appropriate direction (Figure 5), this changes the coordinates of the lower

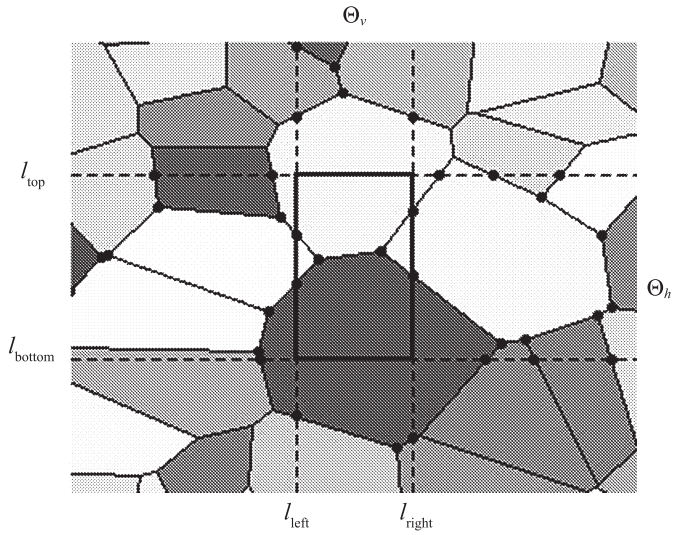

Figure 3. Stopping points for the current location of the rectangle R in regions Θ_h, and Θ_v

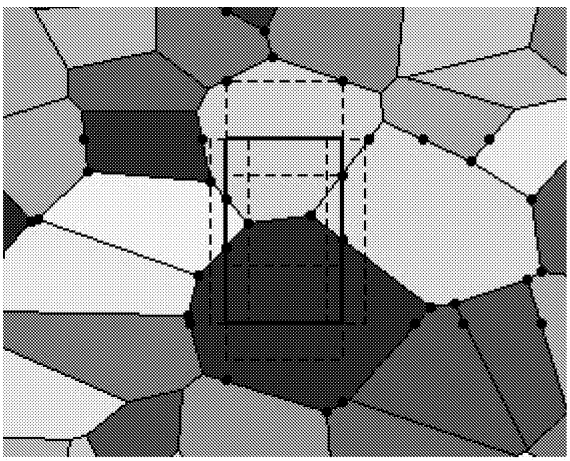

Figure 4. Possible shifts of the rectangle to the left, right, upward, and downward at the current step of the coordinate-wise descent algorithm (dashed lines)

left corner of the rectangle (x_0, y_0). Transition to the first step of the algorithm. Otherwise algorithm stops.

Now we write formal expressions for each step of the algorithm.

Regions: $\Theta_h = \{(x, y) \mid (y <= y_0 + h) \& (y >= y_0)\}$, and $\Theta_v = \{(x, y) \mid (x >= x_0) \& (x <= x_0 + w)\}$.

The set of stopping points are either the vertices of the polygons or possible intersections of the sides of the rectangle R with the edges of the polygons, when the rectangle is shifted.

The shift of the rectangle from local position to the intersection with the stopping point guarantees a linear change of the increment of the weighted area relative to the value of the shift, since the set of polygons covered by the rectangle — P_0 does not change. When

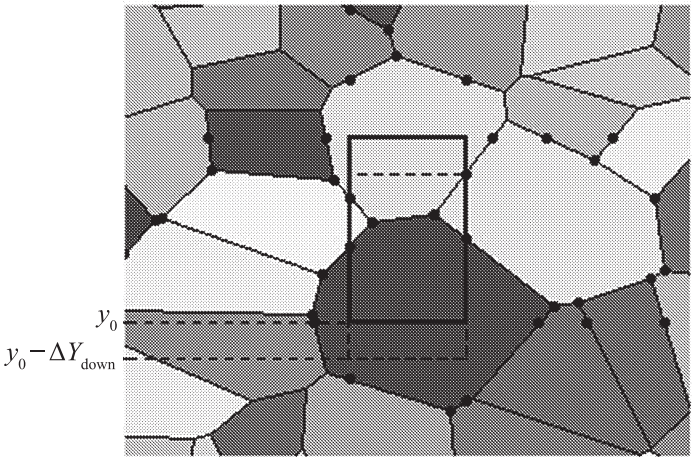

Figure 5. The best shift of the rectangle at the current step of the coordinate-wise descent algorithm is the downward shift (dashed lines)

the stopping point is reached by the side of the rectangle, during the movement the set P_0 could change.

The set of stopping points to shift the rectangle in horizontal directions (left or right) is $V_h = \{(x,y) \in \Theta_h \mid \exists p_j \in P : \ (\exists v_{jk} : (x,y) = v_{jk})$ *or* $(\exists a_{jk} : \ (a_{jk} \cap l_{\text{top}} = (x,y))$ *or* $(a_{jk} \cap l_{\text{bottom}} = (x,y)))\}$.

The set of stopping points to shift the rectangle in vertical directions (up or down) is $V_v = \{(x,y) \in \Theta_v \mid \exists p_j \in P : \ (\exists v_{jk} : (x,y) = v_{jk})$ *or* $(\exists a_{jk} : \ (a_{jk} \cap l_{\text{left}} = (x,y))$ *or* $(a_{jk} \cap l_{\text{right}} = (x,y)))\}$.

The maximum shift to the left ΔX_{left} is determined as a minimum of two numbers: the distance from the left side of the rectangle to the nearest stopping point outside the rectangle to the left of the left side of the rectangle, and the distance from the right side of the rectangle to the nearest stopping point inside the rectangle. If there are no stopping points inside the rectangle, the distance from the right side is not taken into account. If there are no stopping points outside the rectangle, the distance from the left side is not taken into account. The situation, when the set of stopping points is empty ($V_h = \varnothing$ or $V_v = \varnothing$), is impossible, because the first step of the general algorithm guarantees the intersection of the rectangle with at least one polygon ($P_0 \neq \varnothing$):

$$\Delta X_{\text{left}} = \min \left\{ \begin{array}{l} x_0 - \max\{x : (x, t_y) \in V_h \ \&\ x < x_0\}, \\ x_0 + w - \max\{x : (x,y) \in V_h \ \&\ x > x_0\} \end{array} \right\}.$$

The maximum shift to the right ΔX_{right} is determined as a minimum of two numbers: the distance from the right side of the rectangle to the nearest stopping point outside the rectangle to the right of the right side of the rectangle, and the distance from the left side of the rectangle to the nearest stopping point inside the rectangle:

$$\Delta X_{\text{right}} = \min \left\{ \begin{array}{l} \min\{x : (x,y) \in V_h \ \&\ x > x_0 + w\} - x_0 - w, \\ \min\{x : (x,y) \in V_h \ \&\ x < x_0 + w\} - x_0 \end{array} \right\}.$$

The maximum upward shift ΔY_{up} is determined as a minimum of two numbers: the distance from the top side of the rectangle to the nearest stopping point outside the

rectangle above the top side of the rectangle, and the distance from the bottom side of the rectangle to the nearest stopping point inside the rectangle:

$$\Delta Y_{\text{up}} = \min \left\{ \begin{array}{l} \min\{y : (x,y) \in V_v \ \&\ y > y_0 + h\} - y_0 - h, \\ \min\{y : (x,y) \in V_v \ \&\ y < y_0 + h\} - y_0 \end{array} \right\}.$$

The maximum downward shift ΔY_{down} is determined as a minimum of two numbers: the distance from the bottom side of the rectangle to the nearest stopping point outside the rectangle below the bottom side of the rectangle, and the distance from the top side of the rectangle to the nearest stopping point inside the rectangle:

$$\Delta Y_{\text{down}} = \min \left\{ \begin{array}{l} y_0 - \max\{y : (x,y) \in V_v \ \&\ y < y_0\}, \\ y_0 + h - \max\{y : (x,y) \in V_v \ \&\ y > y_0\} \end{array} \right\}.$$

In Figure 4 dotted lines indicate the maximum shifts of the rectangle in each direction. You can see that the left shift is limited to the point to the left of the rectangle, which is the vertex of the polygon. The shift to the right is limited to the point inside the rectangle, which is also the vertex of the polygon. The upward shift is limited to the point that is the intersection point of the polygon edge and the line l_{left} — the continuation of the left side of the rectangle. The downward shift is limited to the point on the right side of the rectangle, which is the intersection of the side of the rectangle and the edge of the polygon.

When rectangle moves the rectangle area is changed by an amount equal to the sum of the weighted areas of the parts of the polygons. Each part of the polygon is a trapezoid (or triangle) with bases parallel to the corresponding side of the rectangle, and the same heights. Since the sum of the base lengths does not change, the change in area is proportional to the height of the trapezoids, and its heights are equal to the shift value.

The weighted area increment when shifted to the left is calculated as

$$\Delta S_{\text{left}} = \sum_{p_i \cap R_{\text{left}-\text{left}} \neq \varnothing} G_i \cdot S(p_i \cap R_{\text{left}-\text{left}}) - \sum_{p_i \cap R_{\text{left}-\text{right}} \neq \varnothing} G_i \cdot S(p_i \cap R_{\text{left}-\text{right}}). \quad (2)$$

In (2) rectangle $R_{\text{left}-\text{left}}$ has lower left corner in point $(x_0 - \Delta X_{\text{left}}, y_0)$, and upper right corner in point $(x_0, y_0 + h)$ (in Figure 4 it is a rectangle has left adjacent to the left side of the rectangle R indicated by a solid line), rectangle $R_{\text{left}-\text{right}}$ has lower left corner in point $(x_0 + w - \Delta X_{\text{left}}, y_0)$, and upper right corner in point $(x_0 + w, y_0 + h)$ (in Figure 4 it is a rectangle has left adjacent to the right side of the rectangle R indicated by a solid line).

The weighted area increment when shifted to the right is calculated as

$$\Delta S_{\text{right}} = \sum_{p_i \cap R_{\text{right}-\text{right}} \neq \varnothing} G_i \cdot S(p_i \cap R_{\text{right}-\text{right}}) - \sum_{p_i \cap R_{\text{right}-\text{left}} \neq \varnothing} G_i \cdot S(p_i \cap R_{\text{right}-\text{left}}). \quad (3)$$

In formula (3) rectangle $R_{\text{right}-\text{right}}$ has lower left corner in point $(x_0 + w, y_0)$, and upper right corner in point $(x_0 + w + \Delta X_{\text{right}}, y_0 + h)$, rectangle $R_{\text{right}-\text{left}}$ has lower left corner in point (x_0, y_0), and upper right corner in point $(x_0 + \Delta X_{\text{right}}, y_0 + h)$.

The weighted area increment when shifted upward is calculated as

$$\Delta S_{\text{up}} = \sum_{p_i \cap R_{\text{up}-\text{top}} \neq \varnothing} G_i \cdot S(p_i \cap R_{\text{up}-\text{top}}) - \sum_{p_i \cap R_{\text{up}-\text{bottom}} \neq \varnothing} G_i \cdot S(p_i \cap R_{\text{up}-\text{bottom}}). \quad (4)$$

In (4) rectangle $R_{\text{up}-\text{top}}$ has lower left corner in point $(x_0, y_0 + h)$, and upper right corner in point $(x_0+w, y_0+h+\Delta Y_{\text{up}})$, rectangle $R_{\text{up}-\text{bottom}}$ has lower left corner in point (x_0, y_0), and upper right corner in point $(x_0 + w, y_0 + \Delta Y_{\text{up}})$.

The weighted area increment when shifted downward is calculated as

$$\Delta S_{\text{down}} = \sum_{p_i \cap R_{\text{down}-\text{bottom}} \neq \varnothing} G_i \cdot S(p_i \cap R_{\text{down}-\text{bottom}}) - \sum_{p_i \cap R_{\text{down}-\text{top}} \neq \varnothing} G_i \cdot S(p_i \cap R_{\text{down}-\text{top}}). \tag{5}$$

In (5) rectangle $R_{\text{down}-\text{bottom}}$ has lower left corner in point $(x_0, y_0 - \Delta Y_{\text{down}})$, and upper right corner in point (x_0+w, y_0), rectangle $R_{\text{down}-\text{top}}$ has lower left corner in point $(x_0, y_0+ h - \Delta Y_{\text{down}})$, and upper right corner in point $(x_0 + w, y_0 + h)$.

In Figure 6 the results of the coordinate-wise movement of the rectangle from the initial location indicated by the solid line to the optimal location indicated by the dashed line are presented.

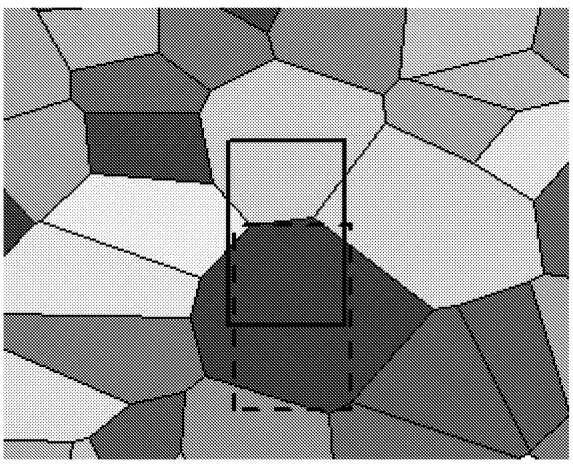

Figure 6. The initial (solid lines) and final (dashed lines) locations of the rectangle R in the coordinate-wise descent algorithm

The estimation of relative algorithm error δ can be calculated as follows.

Let S^{opt} is a weighted area of the optimal solution, that locates in region Φ_{pq}, S^{alg} is a weighted area of algorithm solution. Let we know some solution with weighted area $S^{\text{good}} > 0$. Define $G^{\text{good}} = \frac{S^{\text{good}}}{w \cdot h}$ — average density in R corresponding S^{good}. Calculate β as $\frac{G^{\max}}{G^{\text{good}}}$, where $G^{\max}$ is a maximum density among all polygons.

Relative algorithm error δ satisfies the following expression:

$$\delta = \frac{S^{\text{opt}} - S^{\text{alg}}}{S^{\text{opt}}} \leqslant \frac{S^{\text{opt}} - S^{\text{alg}}}{S^{\text{good}}}.$$

Taking into account the upper bound (1) for S^{opt}, the expression can be proposed:

$$\delta \leqslant \frac{S^{\text{opt}} - S^{\text{alg}}}{S^{\text{good}}} \leqslant \frac{S_{pq} + \left(G^{\max}_{pq} - G^{\min}_{pq}\right) \cdot (M + N - 1) \cdot d_x \cdot d_y - S^{\text{alg}}}{G^{\text{good}} \cdot M \cdot N \cdot d_x \cdot d_y} \leqslant$$

$$\leqslant \frac{S_{pq} + G^{\max} \cdot (M+N-1) \cdot d_x \cdot d_y - S^{\text{alg}}}{G^{\text{good}} \cdot M \cdot N \cdot d_x \cdot d_y} \leqslant$$

$$\leqslant \frac{G^{\max} \cdot (M+N-1) \cdot d_x \cdot d_y}{G^{\text{good}} \cdot M \cdot N \cdot d_x \cdot d_y} \leqslant \frac{(M+N-1) \cdot \beta}{M \cdot N}.$$

Without loss of generality, let $d_x = d_y$ and $w = \alpha \cdot h$, then $M = \alpha \cdot N$, and

$$\delta \leqslant \frac{(M+N-1) \cdot \beta}{M \cdot N} \leqslant \frac{(\alpha+1) \cdot \beta}{\alpha \cdot N}.$$

Let $\gamma > 0$ be an arbitrary constant. If the following relationship holds, then the relative algorithm error $\delta \leqslant \gamma$:

$$d_y \leqslant \frac{h \cdot \alpha \cdot \gamma}{(\alpha+1) \cdot \beta}.$$

For example, if the rectangle R has $h = 1000$ m and $w = 500$ m (i. e. $\alpha = 1/2$), and $\beta = 3$, then a cell size of no more than 5.5 provides a solution with relative error not exceeding $\gamma = 0.05$.

Conclusion. The article presents the formulation and algorithm for solving the problem of maximizing the weighted area of axis-parallel rectangle covering polygons of arbitrary shapes and given densities.

The algorithm can find a solution to the problem with a given accuracy.

The time complexity of the proposed algorithm is $O(n \cdot P \cdot Q)$. Since the main algorithm goes through all the cells $P \times Q$, and coordinate-wise descent algorithm take into account edges and vertices of all polygons.

The task finds its practical application in the area of forest cutting planning.

References

1. Hussain M. M., Trajcevski G. Maximizing area-range sum for spatial shapes (MAxRS3). *SSDBM18: 30th International Conference on Scientific and Statistical Database Management*, July 9–11, 2018. Bozen-Bolzano, Italy; New York, USA, ACM, 5 p. https://doi.org/10.1145/3221269.3221301

2. Shegelman I. R., Lukashevich V. M. *Podgotovitel'nye raboty v otechestvennoy sisteme lesopol'zovania* [*Preparatory work in the national forest management system*]. Monograph. Petrozavodsk, Publishing House Petrozavodsk State University, 2012, 84 p. (In Russian)

3. Bentley J. Programming Pearls — Algorithm design techniques. *Communications of the ACM*, 1984, vol. 27, iss. 9, pp. 865–873.

4. Fischer P., Höffgen K., Lefmann H., Luczak T. Approximations with axis-aligned rectangles. *Fundamentals of Computation Theory*, 1993, vol. 710, pp. 244–255.

5. Fukuda T., Morimoto Y., Morishita S., Tokuyama T. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. *ACM SIGMOD Record*, 1996, no. 25(2), pp. 13–23.

6. Tamaki H., Tokuyama T. Algorithms for the maximum subarray problem based on matrix multiplication. *Interdisciplinary Information Sciences*, 2000, vol. 6, no. 2, pp. 99–104.

7. Takaoka T. Efficient algorithms for the maximum subarray problem by distance matrix multiplication. *Electronic Notes in Theoretical Computer Science*, 2002, vol. 61, pp. 191–200.

8. Daniels K., Milenkovic V., Roth D. *Finding the largest rectangle in several classes of polygons.* Cambridge, Massachusetts, Harvard University Press, 1995, 40 p.

9. Nandy S. C., Bhattacharya B. B. A unified algorithm for finding maximum and minimum object enclosing rectangles and cuboids. *Computers Mathematical Applications (Computers and Mathematics with Applications)*, 1995, vol. 29, no. 8, pp. 45–61.

10. Backurs A., Dikkala N., Tzamos C. *Tight hardness results for maximum weight rectangles.* arXiv preprint arXiv:1602.05837. 2016. Available at: https://arxiv.org/abs/1602.05837 (accessed: May 15, 2019).

Received: April 29, 2019.
Accepted: November 07, 2019.

Author's information:

Liudmila V. Shchegoleva — Dr. Sci. in Technics, Associate Professor; schegoleva@petrsu.ru

Roman V. Voronov — Dr. Sci. in Technics, Associate Professor; rvoronov@petrsu.ru

Leonid Sedov — Postgraduent Student; leonid.sedov@liu.se

Задача покрытия полигонов прямоугольной областью с максимальным весом

Л. В. Щеголева[1], *Р. В. Воронов*[1], *Л. Седов*[2]

[1] Петрозаводский государственный университет, Российская Федерация, 185910, Петрозаводск, пр. Ленина, 33

[2] Линчепингский университет, Швеция, 60221, Норрчепинг, Бредгатан, 33

Для цитирования: *Shchegoleva L. V., Voronov R. V., Sedov L.* The problem of a maximal weighted area of axis-parallel rectangle that covers polygons // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. 2019. Т. 15. Вып. 4. С. 592–602. https://doi.org/10.21638/11702/spbu10.2019.414

Представлена задача поиска оптимального расположения прямоугольника с максимальной взвешенной площадью. Размеры прямоугольника заданы, стороны прямоугольника параллельны осям координат. На плоскости расположены несамопересекающиеся полигоны произвольной формы с заданной плотностью. Взвешенная площадь прямоугольника рассчитывается как суммарная площадь частей полигонов, накрытых прямоугольником, умноженных на плотность. Описан алгоритм решения задачи. Такая задача возникает при определении мест рубок леса, когда планируемая лесосека может быть смоделирована прямоугольником, а полигоны описывают таксационные выделы лесного фонда, для каждого из которых известен запас леса на 1 га.

Ключевые слова: задача максимизации взвешенной площади, полигоны.

Контактная информация:

Щеголева Людмила Владимировна — д-р техн. наук, доц.; schegoleva@petrsu.ru

Воронов Роман Владимирович — д-р техн. наук, доц.; rvoronov@petrsu.ru

Седов Леонид — аспирант; leonid.sedov@liu.se