アルゴリズム 26－4
（1992. 3. 23）

多角形内部の格子点列挙と2変数整数計画問題への応用

金丸直義* 西関隆夫** 浅野哲夫***

*東北大学工学部通信工学科
**東北大学工学部情報工学科
***大阪電気通信大学

本報告では，与えられた凸 m 角形内部のすべての整数格子点を列挙する $O(K+m+\log n)$ 時間のアルゴリズムを与える．ここで K は列挙された格子点の個数であり，n は凸 m 角形の大きさ，すなわち m 角形を包含する軸平行な長方形の垂直，水平な辺のうち短い方の長さである．さらに，m 個の制約式を持つ2変数整数計画問題を解く $O(m\log m+\log n)$ 時間のアルゴリズムを与える．ここで n は計画問題の許容解空間である凸多角形の大きさを表す．このアルゴリズムは従来知られているアルゴリズムより単純であり，時間計算量も改善している．

Efficient Enumeration of Grid Points in a Polygon and its Application to Integer Programming

Naoyoshi Kanamaru* Takao Nishizeki** Tetsuo Asano***

*Department of Electorical Communications
Faculty of Engineering, Tohoku University
Sendai 980, Japan
**Department of Information Engineering
Faculty of Engineering, Tohoku University
Sendai 980, Japan
***Osaka Electro-Communication University
Neyagawa 572, Japan

This paper presents an algorithm for enumerating all the integer-grid points in a given convex m-gon in $O(K+m+\log n)$ time where K is the number of such grid points and n is the dimension of the m-gon, i.e., the shorter length of the horizontal and vertical sides of an axis-parallel rectangle enclosing the m-gon. Furthermore the paper gives a simple algorithm which solves a two-variable integer programming problem with m constraints in $O(m\log m+\log n)$ time where n is the dimension of a convex polygon corresponding to the feasible solution space. This improves the best known algorithm in complexity and simplicity.

1. Introduction

In a raster graphics a figure is expressed as a set of grid points, that is, points of integral coordinates. Given a polygonal contour of a figure whose vertices are specified by coordinates of floating-point numbers, we wish to find grid points contained in the polygon.[5] A naive algorithm for the problem is to examine for each grid line the intersection with the polygon. When the dimension of a polygon is n, this algorithm requires $\Omega(K+n)$ time where K is the output size, that is, the number of grid points in the polygon.

In this paper we first present an algorithm for enumerating all the grid points contained in an arbitrary triangle in $O(K + \log n)$ time. If we are interested only in whether there exists a grid point in a triangle, it is easy to modify the algorithm so that it checks the existence in $O(\log n)$ time. The algorithm together with an efficient triangulation algorithm leads to an efficient algorithm for enumerating all grid points contained in a polygon. We then extend the algorithm for triangles, in particular, to the case of convex polygons. The algorithm enumerates all grid points in a convex m-gon in $O(K + m + \log n)$ time.

An application to the integer programming problem is also considered. In fact we give an algorithm which solves a two-variable integer programming problem with m constraints in $O(m \log m + \log n)$ time. Our algorithm improves the existing algorithms both in complexity and simplicity.[2,4] It is known that the integer programming problem with a fixed number of variables is polynomially solvable.[7] In the case of two-variable, the best known one runs in $O(m \log m + m \log n)$ time.[2]

A key idea is a grid-point preserving transformation which is similar to the well-known Euclidean algorithm for finding the greatest common divisor of two integers.

2. Grid Points on a Line

Throughout the paper we classify lines into two types depending on the slope. The slope $m(L)$ of a line segment L, (x_1, y_1)-(x_2, y_2), is defined as follows: $m(L) = (y_2 - y_1)/(x_2 - x_1)$. A line L is called a type V line if $|m(L)| \geq 1$; otherwise, called a type H line. For a line segment L, its *horizontal* and *vertical dimensions* $w_h(L)$ and $w_v(L)$ are defined as follows:

$$w_h(L) = |x_2 - x_1| \text{ and } w_v(L) = |y_2 - y_1|.$$

The smaller one of the horizontal and vertical dimensions is called the *dimension* of a line segment.

Before proceeding to the problem of enumerating all grid points contained in a polygon, we consider the following simpler problem.

[Problem 1] Given a line segment $L : (x_1, y_1)$-(x_2, y_2) in a grid plane, enumerate all the grid points on L.

This problem can be solved easily if its horizontal or vertical dimension is sufficiently small. If $w_h(L)$ is small, then the number of vertical grid lines intersecting the line segment L is also small and hence we can check every such grid line to enumerate all the grid points. Thus we hereafter assume that $w_h(L)$ and $w_v(L)$ are both large.

A basic idea here is to transform a line segment into another without losing any grid point. As seen later, a type H line segment is transformed into a type V line segment and vice versa. After $O(\log n)$ iterations we have a line segment whose dimension is small enough, where n is the dimension of a given line segment.

One may assume without loss of generality that the slope of a given line segment L is positive, that is, $m(L) > 0$; otherwise, reverse the y-axis and consider the line segment $(x_1, -y_1)$-$(x_2, -y_2)$ of positive slope. Then the above-mentioned transformation can be represented as follows:

$$\begin{array}{ll} T_v : (x, y) \Longrightarrow (x, y - kx) & \text{for type V, and} \\ T_h : (x, y) \Longrightarrow (x - ky, y) & \text{for type H,} \end{array}$$

where an integral parameter k is chosen so that

$$\begin{array}{ll} k = \lfloor m(L) \rfloor & \text{for type V, and} \\ k = \lfloor 1/m(L) \rfloor & \text{for type H.} \end{array}$$

Thus when a line segment L of positive slope is transformed into a new line segment L', the slope of L' is non-negative and more precisely

$$\begin{array}{ll} 0 \leq m(L') = m(L) - k < 1 & \text{if } L \text{ is of type V, and} \\ 1 < m(L') = 1/(1/m(L) - k) & \text{if } L \text{ is of type H.} \end{array}$$

Therefore an application of the transformation T_v to a line segment of type V results in a line segment of type H and vice versa. Figure 1 illustrates the transformations $T_v : L \Longrightarrow L'$ and $T_h : L' \Longrightarrow L''$. While the transformation is repeated until the dimension of the resulting line segment is sufficiently small, the type of a line segment alternates between types V and H.

The transformation can be expressed in terms of a matrix. Define a matrix A_t for each type t as follows.

$$A_v = \begin{pmatrix} 1 & 0 \\ -k & 1 \end{pmatrix} \quad \text{for type V, and}$$

$$A_h = \begin{pmatrix} 1 & -k \\ 0 & 1 \end{pmatrix} \quad \text{for type H.}$$

Then the mapping from a point (x, y) to a point (x', y') is expressed as

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = A_t \begin{pmatrix} x \\ y \end{pmatrix}.$$

The transformation matrix A_t is unimodular, and has the following integral inverse matrix A_t^{-1}.

$$A_v^{-1} = \begin{pmatrix} 1 & 0 \\ k & 1 \end{pmatrix} \quad \text{for type V, and}$$

$$A_h^{-1} = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix} \quad \text{for type H.}$$

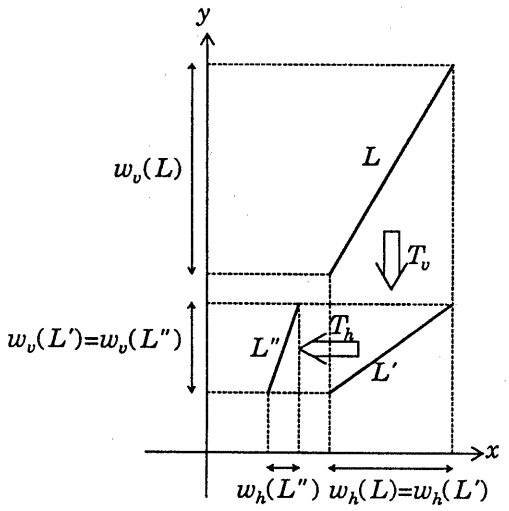

Fig. 1. Transformations T_v and T_h.

The following lemma is almost obvious from the definition of the transformations.

Lemma 1 *If a line segment L is transformed into a line segment L', then there is one-to-one correspondence between the grid points on L and those on L'. Furthermore,*

$$w_v(L') \le w_v(L)/2 \quad \text{if } L \text{ is of type V, and}$$
$$w_h(L') \le w_h(L)/2 \quad \text{if } L \text{ is of type H.}$$

Proof. Since A_t is unimodular, there is one-to-one correspondence between the grid points on L and those on L'. We prove the remaining claim only for the case when L is of type V (See Figure 1); the proof for the other case is similar. Since L' is of type H,

$$w_v(L') < w_h(L') = w_h(L).$$

Since $m(L') = m(L) - k$ and $k \ge 1$,

$$w_v(L') = w_v(L) - kw_h(L) \le w_v(L) - w_h(L).$$

Therefore

$$w_v(L') \le \min\{w_h(L), w_v(L) - w_h(L)\} \le \frac{1}{2}w_v(L).$$

□

We are now ready to present an algorithm for enumerating all grid points on a line segment L.

Line(L)

/* $L : (x_1, y_1)$-(x_2, y_2) is a line segment, where the coordinates are real numbers */

{

let c be a small constant, say 1;

$M :=$ a 2×2 identity matrix with 1's in its diagonal;

while ($\min(w_h(L), w_v(L)) > c$) {

$t :=$ type of a line segment L;

calculate the parameter k characterizing the transformation T_t;

$M := M \cdot A_t^{-1}$;

apply the transformation T_t to L;

let L be the resulting line segment;

} /* end of while */

for (each grid point (x', y') on the final line segment L)

report the coordinate $(x, y)^T = M \cdot (x', y')^T$;

} /* end of Line() */

We have the following theorem.

Theorem 1 *Given a line segment L of dimension n, the algorithm* **Line** *enumerates all the grid points on L in $O(K + \log n)$ time, where K is the number of those grid points.*

Proof. The correctness of the algorithm follows from the discussions above. Lemma 1 implies that the dimension of L becomes a small constant after $O(\log n)$ iterations. The inverse transformation matrix which transforms the coordinate of each point on the final line segment into that on the original line segment is obtained by multiplying the inverse transformation matrix A_t^{-1} of each transformation from the right at each iteration. Once we get a line segment which is almost horizontal or vertical, we can report the coordinates of all the grid points on the final line segment after applying the inverse transformation above to those points. Thus the time required is $O(K + \log n)$, where K is the number of grid points to be reported. □

Our algorithm above is quite similar to Euclidean method. Notice that if $(x_1, y_1) = (0, 0)$ and both x_2 and y_2 are integers then our problem can be solved immediately by finding the greatest common divisor of x_2 and y_2.

3. Grid Points in a Triangle

We next consider the following problem.

[Problem 2] Given a triangle in the grid plane, enumerate all the grid points in the triangle.

If a triangle is so skinny that only a few grid lines have grid points in the triangle, a naive algorithm requires time proportional to the dimension n of the triangle, which is sometimes too expensive. The algorithm required here should enumerate all the grid points in $O(K + \log n)$ time

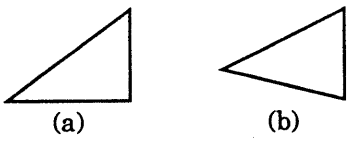

Fig. 2. Easy triangles.

even in the worst case, that is, with $O(\log n)$ time overhead.

One may assume that a given triangle has a horizontal or vertical side; otherwise, decompose the triangle into two triangles by drawing a horizontal or vertical chord from a vertex, and consider the two triangles separately. We choose an axis-parallel side of a triangle arbitrarily, and call it the *parallel side* of the triangle and the other two sides the *long sides*.

As shown in the following lemma, one can easily solve the problem if a given triangle of an axis-parallel side satisfies the following (a) or (b):

(a) the triangle is rectangular; or

(b) the two long sides have slopes of different signs.

Such triangles are called *easy triangles* (See Figure 2).

Lemma 2 *All grid points in an easy triangle can be enumerated in $O(K+1)$ time, where K is the number of the enumerated grid points.*

Proof. We prove the lemma only for the case when the parallel side is vertical; the proof for the other case is similar. We sweep vertical grid lines from the parallel side to the vertex common to the two long sides and enumerate the grid points on the intersection of the grid line with the triangle. This operation is repeated until the current vertical intersection contains no grid point. Since only the last scan line may contain no grid point in the triangle, the algorithm runs in $O(K+1)$ time. □

We thus consider a non-easy triangle having a parallel side, i.e., a triangle such that the slopes of the two long sides have the same sign, say plus (See Figure 3). We apply to the triangle a transformation which is very similar to one in the previous section.

[Transformation of a Triangle]

(1) Let v_1, v_2 and v_3 be the three vertices of a triangle.

(2) Choose one of the two long sides arbitrarily.

(3) Calculate the transformation matrix A_t for T_t concerning the chosen long side.

(4) Transform the three vertices into $A_t \cdot v_1, A_t \cdot v_2$ and $A_t \cdot v_3$.

One can observe the following lemma.

Lemma 3 *There is one-to-one correspondence between the grid points in a triangle and the grid points in the triangle obtained by the transformation above.*

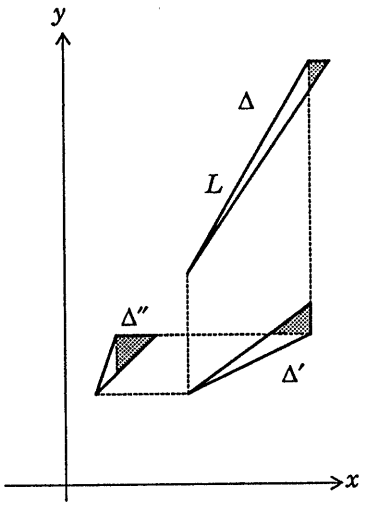

Fig. 3. Transformations of triangles where trimmed parts are shaded.

If the transformed triangle is easy, then we have done. Otherwise, we repeat the transformation. However, before applying the succeeding transformation, we trim off an easy part from a triangle: decompose the current triangle into two triangles by drawing an axis-parallel chord from a vertex so that one of the resulting triangles is rectangular, as depicted in Figure 3. All the grid points in this rectangular triangle can be easily enumerated. We hence apply the transformation above to the other resulting triangle. We repeat this operation until the triangle becomes easy or the dimension is sufficiently small. Clearly the iteration is done at most $O(\log n)$ times. Therefore all the grid points can be enumerated in $O(K + \log n)$ time.

Since there are two long sides, two different transformations are possible. However there is no much difference whichever side is chosen to determine the parameter k characterizing the transformation. It should be noted that if one of the two long sides is type V and the other type H then the new triangle obtained by the transformation is necessarily easy.

We are now ready to present an algorithm for enumerating all grid points in a triangle Δ.

```
Triangle(Δ);
/* One may assume that Δ is a triangle having a parallel
   side */
{
    let c be a small constant;
    M := a 2×2 identity matrix with 1's in its diagonal;
```

```
while ( the dimension of Δ is greater than c ){
    if ( Δ is easy ) {
        enumerate all grid points (after multiplying M) in Δ;
        return;
    } /* end if */
    choose one long side L of Δ arbitrarily;
    t := type of the line segment L;
    if ( t = V (resp. H) and the parallel side of
       Δ is horizontal (resp. vertical) ) {
        trim off Δ vertically (resp. horizontally);
        enumerate all the grid points (after multiplying M) in the trimmed rectangular
            triangle;
        let Δ be the resulting triangle having a
            vertical (resp. horizontal) side;
    } /* end if */
    calculate the parameter k for L;
    M := M · A_t^{-1};
    apply the transformation defined by k to the
        current triangle;
    let Δ be the resulting triangle;
} /* end of while */
using a brute-force algorithm, enumerate all grid
    points (after multiplying M) in the final triangle Δ of small dimension;
} /* end of Triangle() */
```

We have the following theorem and corollary.

Theorem 2 *Given a triangle of dimension n, all the grid points in the triangle can be enumerated in $O(K + \log n)$ time, where K is the total number of those grid points.*

Corollary 1 *Given a triangle of dimension n, it can be determined in $O(\log n)$ time whether there is a grid point in the triangle.*

A naive extension of the argument above is as follows. Given a polygonal region, we first partition it into triangles. If it is a simple m-gon, the triangulation can be done in $O(m)$ time.[1] If it has holes, the triangulation can be done in $O(m \log m)$ time.[3] This results in $O(m)$ triangles. So, if we apply the algorithm above for each triangle, we can enumerate all the grid points in the polygonal region. The time required is $O(K + m \log n)$ plus the time for partition, where K is the total number of those grid points.

4. Grid Points in a Convex Polygon

In this section we present an algorithm for enumerating all the grid points in a convex polygon. The algorithm is very similar to that for a triangle. As described later, we will define an "easy" convex m'-gon for which all the grid points can be enumerated in $O(K + m')$ time. We first check whether a given convex polygon is easy. If so, we have done. Otherwise, we trim off easy parts and apply to the remaining part the transformation similar to the one in the previous section. This operation is repeated until the remaining part is sufficiently small. The iteration is done $O(\log n)$ times. This is a rough sketch of the algorithm.

We shall describe the algorithm in detail. An "easy" convex polygon is defined as follows: a convex polygon P with a horizontal side S_h is *easy* if there is an axis-parallel rectangle which encloses P and has S_h as a horizontal side. (The two shaded regions in Figure 4 are easy polygons.) Then we have the following lemma.

Lemma 4 *All the grid points in a given easy convex m'-gon P can be enumerated in $O(K + m')$ time where K is the total number of those grid points.*

Proof. We prove the lemma only for the case where the horizontal side S_h is the upper side of the axis-parallel rectangle enclosing P; the proof for the case of the lower side is similar. If we sweep horizontal grid lines from up to down and enumerate all the grid points on the intersection of the sweep line with P until the intersection contains no grid point, we can enumerate all the grid points in P. Intersections of the sweep line with P are found by walking on the boundary of P, which takes $O(m')$ time in total. Thus we have the lemma. □

As mentioned above, our algorithm trims off easy parts and then apply to the remaining part the transformation described in the previous sections. Suppose that the transformation matrix is M at some iteration. Denote by $M \cdot P$ the polygon obtained by applying the transformation M to P. Then $M \cdot P$ is also a convex polygon. Suppose that we are now going to enumerate all grid points in the region of $M \cdot P$ restricted to a horizontal interval from $x = x_L$ to $x = x_R$; the region is denoted by $M \cdot P[x_L, x_R]$.

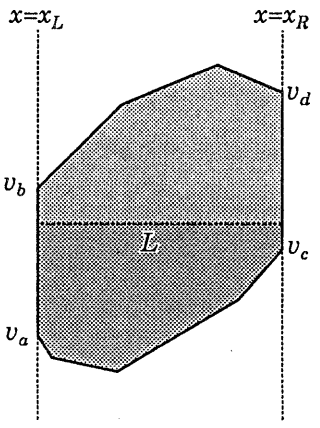

Fig. 4. Partition into two easy convex polygons.

Given such a convex polygon $M \cdot P$ and a horizontal interval $[x_L, x_R]$, we want to remove easy parts; this operation is called the *horizontal trimming*. Let $\overline{v_a v_b}$ and $\overline{v_c v_d}$ be the intersections of $M \cdot P$ with the vertical lines $x = x_L$ and $x = x_R$, respectively and assume that $y(v_a) \leq y(v_b)$ and $y(v_c) \leq y(v_d)$. If the two intervals $[y(v_a), y(v_b)]$ and $[y(v_c), y(v_d)]$ have a nonempty intersection, the whole $M \cdot P[x_L, x_R]$ can be partitioned into two easy convex polygons by an arbitrary horizontal line which acrosses the intersection (See Figure 4). Thus we now assume that the two intervals have an empty intersection. Then one may assume that $y(v_a) \leq y(v_b) < y(v_c) \leq y(v_d)$; the other case is similar (See Figure 5). The *horizontal trimming* of $M \cdot P[x_L, x_R]$ is to partition it into three parts: the lower part below $y = y(v_b)$, the upper part above $y = y(v_c)$ and the middle part. Both the lower and upper parts are easy convex polygons. We transform the middle part into a new convex polygon P' by a horizontal transformation T_h. P' is a region of a convex polygon restricted to a vertical interval $[y(v_b), y(v_c)]$.

In addition to a procedure to enumerate all the grid points in a convex polygon restricted to a horizontal interval, we thus need one more procedure to enumerate all the grid points in a convex polygon restricted to a vertical interval. These procedures will be named horizontal-trim() and vertical-trim(). We will describe only the procedure horizontal-trim() in detail since vertical-trim() is just symmetric to it.

We are now ready to present an algorithm for enumerating all the grid points in a convex polygon P.

Polygon(P)

/* P is a convex polygon */

{

find the leftmost and rightmost vertices v_l and v_r of P;

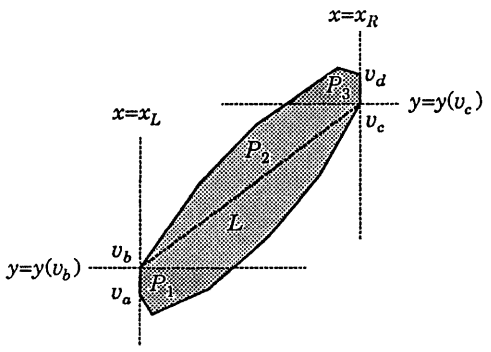

Fig. 5. The horizontal trimming of $M \cdot P[x_L, x_R]$.

if ($y(v_l) \leq y(v_r)$) $M := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$; **else** $M := \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$;

horizontal-trim($x(v_l), x(v_r), M, P$);

} /* end of Polygon() */

horizontal-trim(x_L, x_R, M, P)

{

if ($x_R - x_L$ is sufficiently small) {

enumerate all the grid points by a brute-force algorithm;

exit;

} /* end if */

let $\overline{v_a v_b}$ and $\overline{v_c v_d}$ be the intersections of $M \cdot P$ with $x = x_L$ and $x = x_R$;

let L be the shortest line segment connecting $\overline{v_a v_b}$ and $\overline{v_c v_d}$;

if (L is of type V) {

$M := A_v \cdot M$ where A_v is the matrix determined by L;

update $\overline{v_a v_b}$ and $\overline{v_c v_d}$ and L;

} /* end if */

if (L is horizontal (i.e. the intervals $[y(v_a), y(v_b)]$ and $[y(v_c), y(v_d)]$ intersect as shown in Figure 4)) {

partition $M \cdot P[x_L, x_R]$ into two easy convex polygons;

enumerate all the grid points in each polygon;

exit;

} /* end if */

by the horizontal trimming, partition it into three parts: the lower part P_1, the middle part P_2 and the upper part P_3 (See Figure 5);

enumerate all the grid points in the two easy convex polygons P_1 and P_3;

$M := A_h \cdot M$ where A_h is the matrix determined by L;

enumerate all the grid points in P_2 by calling **vertical-trim**($y(v_b), y(v_c), M, P$);

} /* end of horizontal-trim() */

We have the following theorem and corollary.

Theorem 3 *The algorithm above enumerates all the grid points in a given convex m-gon P in $O(K + m + \log n)$ time where K is the total number of those grid points and n is the dimension of a rectangle enclosing P.*

Proof. The correctness of the algorithm is obvious from the discussion above. We therefore concentrate ourselves on the analysis of the time complexity. By Lemma 1, we have

$w_h(L') \leq w_h(L)/2$ in procedure horizontal-trim(), and
$w_v(L') \leq w_v(L)/2$ in procedure vertical-trim()

where L' is the new line segment after the trimming and transformation. Therefore one can see that all the grid points are enumerated after performing the trimming and transformation $O(\log n)$ times. Since one execution of the transformation can be done in constant time, all the transformations can be done in $O(\log n)$ time in total. One can find v_a, v_b, v_c and v_d by scanning edges of the boundary of P. This operation can be done in $O(m + \log n)$ time in total since every edge is visited at most once throughout the execution of the algorithm. Clearly all trimmed easy polygons total up to $O(m + \log n)$ sides. Therefore the grid points in all easy convex polygons can be enumerated in $O(K + m + \log n)$ time in total by Lemma 4. Thus the algorithm runs in $O(K + m + \log n)$ time. □

Corollary 2 *Given a convex m-gon of dimension n, it can be determined in $O(m + \log m)$ time whether there is a grid point in the polygon.*

5. Application to Integer Programming

The technique above can be applied to a two-variable integer programming problem. Given m constraints with two variables, our algorithm finds a solution or determines whether there is no solution in $O(m \log m + \log n)$ time, where n is the dimension of a rectangle enclosing the convex polygon corresponding to the feasible solution space. This is an improvement over the best known algorithm of complexity $O(m \log m + m \log n)$ by S. D. Feit.[2] One important distinction from the previous algorithm is that in our model we do not assume that coefficients of constraints are integers.

The problem addressed here is to
Find integers x and y maximizing the objective function

$$a_{01}x + a_{02}y$$

subject to the constraints

$$a_{11}x + a_{12}y \geq c_1,$$
$$a_{21}x + a_{22}y \geq c_2,$$
$$\vdots$$
$$a_{m1}x + a_{m2}y \geq c_m.$$

There are no restrictions on the coefficients $a_{11}, a_{12}, \cdots, a_{m1}, a_{m2}, c_1, \cdots, c_m$, that is, there are no sign restrictions or they may be even floating-point numbers. But a_{01} and a_{02} have to be integers. Notice that in (Ref. 4) all coefficients had to be non-negative integers and in (Ref. 2) they had to be integers since they are based on a procedure for calculating the greatest common divisor of two integers.

It is well known that the problem of the form above can be transformed into a standard form to find a grid point satisfying all constraints whose x-coordinate is minimal, as follows.[2,9] Let g be the (positive) greatest common divisor of the coefficients a_{01}, a_{02} of the objective function. Then there are integers r and s with

$$r\,a_{01} + s\,a_{02} = g, \ |r| < |a_{02}|, |s| < |a_{01}|.$$

Hence the change of coordinates

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -r & a_{02}/g \\ -s & -a_{01}/g \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}$$

transforms the objective function to $-gx'$. Thus the problem of the form above can be transformed into a standard form in $O(m + \log n')$ time where $n' = \max(a_{01}, a_{02})$.

It is also known that the constraints can be represented in the form of a convex polygon. Of course, it could correspond to an empty set (in this case there is no solution) or an open set (in this case x values may not be bounded). In this paper we assume that the constraints specify a closed convex polygon of at most m sides. The polygon can be found in $O(m \log m)$ time.[8] Thus a two-variable integer programming problem is reduced to a problem of finding a grid point of minimal x-coordinate in a given convex polygon.

There may exist two or more grid points of the same minimal x-coordinate in a convex polygon. The grid point of minimal y-coordinate among them is called the *xmin-ymin* grid point in the polygon. The *ymin-xmin* grid point is similarly defined. We show that one can find the xmin-ymin grid point in a given convex polygon by an algorithm similar to the algorithm **Polygon()** in the previous section.

We have the following lemma.

Lemma 5 *Given an easy convex m'-gon P, one can either find the xmin-ymin grid point in P or know that no grid point is contained in P in $O(m')$ time.*

Proof. We prove the lemma only for the case where the horizontal side S_h is the upper side of the axis-parallel rectangle enclosing P; the proof for the case of the lower side is similar. We first find the intersection of the horizontal grid line $y = \lfloor y(S_h) \rfloor$ with P in $O(m')$ time, where $y(S_h)$ is the y-coordinate of a horizontal side S_h. We can then find a grid point p^* of minimal x-coordinate on the horizontal intersection in constant time. We next find the intersection of the vertical grid line $x = x(p^*)$ with P in $O(m')$ time, and finally find the grid point of minimal y-coordinate on the vertical intersection in constant time. It is the one we wish to find. If the horizontal intersection above does not have any grid point, then no grid point is contained in the m'-gon at all. □

Suppose that we are now going to find the xmin-ymin grid point in a convex polygon restricted to a horizontal interval. We first partition the polygon into three parts, the lower, middle and upper parts, next find the xmin-ymin grid point in each of them, and finally return the xmin-ymin grid point among the three found in the three parts or conclude no grid point exists in the polygon. Since the lower and upper parts are easy convex polygons, one can easily find the xmin-ymin grid points in the two parts. Thus we shall show how to find the xmin-ymin grid point in the middle part.

The following lemma implies that the xmin-ymin grid point and the ymin-xmin grid point in the middle part above coincide.

Lemma 6 *Let P be a convex polygon, and let v be a vertex of P. If there is an axis-parallel rectangle which encloses P and whose lower left vertex is v, then the xmin-ymin grid point in P coincides with the ymin-xmin grid point in P.*

Proof. Let v_1 be the xmin-ymin grid point in P and v_2 be the ymin-xmin grid point in P. By the definition we have $x(v) \leq x(v_1) \leq x(v_2)$ and $y(v) \leq y(v_2) \leq y(v_1)$. Since v, v_1 and v_2 are all in the convex polygon P, the grid point $(x(v_1), y(v_2))$ also must be in P and hence v_1 and v_2 must coincide. □

We transform the middle part into a new convex polygon P' by a horizontal transformation T_h. Then the ymin-xmin grid point in P' corresponds to the ymin-xmin grid point in the middle part since T_h preserves y-coordinate. Thus the problem of finding the xmin-ymin grid point in the middle part reduces to that of finding the ymin-xmin grid point in P'. We vertically partition P' into three parts. One can easily find the ymin-xmin grid points in the right and left parts. By vertically transforming the middle part of P', the problem of finding the ymin-xmin grid point in the part reduces to that of finding the xmin-ymin grid point in the resulting region which is a convex polygon restricted to a horizontal interval.

Thus, by an algorithm similar to **Polygon()**, one can either find the xmin-ymin grid point in a convex m-gon P or decide that no grid point is contained in P in $O(m + \log n)$ time where n is the dimension of a rectangle enclosing P.

We can now conclude that a two-variable integer programming problem with m constraints can be solved in $O(m \log m + \log n + \log n')$ time where $n' = \max(a_{01}, a_{02})$.

6. Conclusions

In this paper we have presented an efficient algorithm for enumerating all the grid points contained in an arbitrary convex m-gon in $O(K + m + \log n)$ time where K is the output size and n is the dimension of the polygon. The basic idea is a transformation preserving grid points. Based on the transformation we have also presented an efficient algorithm for a two-variable integer programming problem. Its time complexity is $O(m \log m + \log n)$ time for m constraint problems, which is an improvement over the existing best algorithm of time complexity $O(m \log m + m \log n)$. We are now trying to extend the result above to higher dimensional cases.

Lee and Chung have recently given an approximation algorithm for the following problem which is related to the grid point enumeration problem: find a minimal set of parallel straight lines with equal spacing to hit all grid points in a given convex polygon.[6]

Acknowledgments

The authors would like to thank D. T. Lee for his fruitful comments on the first part of this paper. The third author would like to thank Kurt Mehlhorn for discussion on the integer programming part. This work was supported in part by the Grants in Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan.

References

1. B. Chazelle, "Triangulating a simple polygon in linear time," *Proc. 31st Symp. of Foundation of Computer Science* (1990) 220–230.
2. S. D. Feit, "A fast algorithm for the two-variable integer programming problem," *J. of ACM* **31-1** (1984) 99–113.
3. M. R. Garey, D. S. Johnson, F. P. Preparata and R. E. Tarjan, "Triangulating a simple polygon," *Inf. Process. Letters* **7** (1978) 175–180.
4. R. Kannan, "A polynomial algorithm for the two-variable integer programming problem," *J. of ACM* **27-1** (1980) 118–122.
5. D. T. Lee, "Shading of regions on vector display devices," *Computer Graphics* **15-3** (1981) 37–44.
6. H. S. Lee and R. C. Chung, "On hitting grid points in a convex polygon with straight lines," *Proc. Sec. Int. Symp. on Algorithms, Lect. Not. in Compt. Sci.* **557** (Springer-Verlag, Dec. 1991) 176–189.
7. H. W. Lenstra, Jr.,"Integer programming with a fixed number of variables," *Mathematics of Operation Research* **8** (1983) 538–548.
8. F. P. Preparata and M. I. Shamos, *Computational Geometry*, (Springer-Verlag, 1985).
9. H. E. Scarf,"Production sets with indivisibilities, Part II. The case of two activities," *Econometrica* **49-2** (1981) 395–423.