Drawing Borders Efficiently

Iwama, Kazuo
School of Informatics, Kyoto University

Miyano, Eiji
Department of Systems Design and Informatics, Kyushu Institute of Technology

Ono, Hirotaka
Department of Computer Science and Communication Engineering, Kyushu University

出版情報：Theory of Computing Systems. 44 (2), pp.230-244, 2009-02. Springer
バージョン：
権利関係：

Drawing Borders Efficiently

Kazuo Iwama[1], Eiji Miyano[2], and Hirotaka Ono[3]

[1] School of Informatics, Kyoto University, Kyoto 606-8501, Japan.
iwama@kuis.kyoto-u.ac.jp
[2] Department of Systems Design and Informatics, Kyushu Institute of Technology, Fukuoka 820-8502, Japan. miyano@ces.kyutech.ac.jp
[3] Department of Computer Science and Communication Engineering, Kyushu University, Fukuoka 819-0395, Japan. ono@csce.kyushu-u.ac.jp

Abstract. A spreadsheet, especially MS Excel, is probably one of the most popular software applications for personal-computer users and gives us convenient and user-friendly tools for drawing tables. Using spreadsheets, we often wish to draw several vertical and horizontal black lines on selective gridlines to enhance the readability of our spreadsheet. Such situations we frequently encounter are formulated as the Border Drawing Problem (BDP). Given a layout of black line segments, we study how to draw it efficiently from an algorithmic view point, by using a set of border styles and investigate its complexity. (i) We first define a formal model based on MS Excel, under which the drawability and the efficiency of border styles are discussed, and then (ii) show that unfortunately the problem is $\mathcal{NP}$-hard for the set of the Excel border styles and for any reasonable subset of the styles. Moreover, in order to provide potentially more efficient drawing, (iii) we propose a new compact set of border styles and show a necessary and sufficient condition of its drawability.

1 Introduction

MS Excel is probably one of the most popular software applications for personal-computer users. Among other nice features, it gives us a convenient and user-friendly tool for drawing tables. Suppose, for example, we wish to draw a table as shown in Figure 1. Other than characters, we have to draw several black lines called *borders*. To do so, we click the "Border Style" button and then there appears the drop-down menu as shown in Figure 2. This includes 12 different styles, style (1) through style (12) in the order of top-left, top-second, through

Tokyo				
9, am	12, noon	3, pm	6, pm	9, pm
S	S	PC	PC	PC
Paris				
9, am	12, noon	3, pm	6, pm	9, pm
PC	C	C	R	R
London				
9, am	12, noon	3, pm	6, pm	9, pm
C	C	PC	PC	PC

Fig. 1. Borders

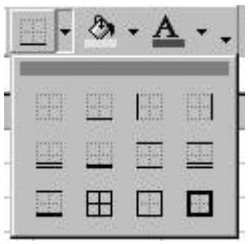


Fig. 2. Excel border styles

bottom-right. To draw the top horizontal border of the table, for example, we select the five horizontal cells just above this border and click style (2). Since the table includes 21 line segments, it is easy to draw it in 21 steps by using only style (2) and style (3). However, it turns out that, by using other styles, the same table can be drawn in as few as four steps.

Thus, there can be a big difference in the efficiency between naive users and highly trained users. It should be noted that the above described mechanism, namely applying ready-made templates sequentially to do something, is an important paradigm in many different systems, including theoretical models. One of the best known examples is the *PQ-tree* [2], which was introduced for checking the consecutive-one property of a Boolean matrix and has also been studied recently in the field of bioinformatics (e.g., [6, 10]). Also, in many data structures, a clever use of basic operations plays a key role for the development of efficient

programs. However, such a rigorous research from an algorithmic point of view has not been extended to more practical systems like MS Excel, Tgif [12] and Xfig [14] (see the previous work paragraph).

Our Contributions In this paper, we concentrate ourselves on MS Excel and investigate the complexity of the Border Drawing Problem (BDP), which is basically the same as the problem of drawing a table described above. Our model has been carefully designed, in order to maintain the basic nature of Excel and at the same time to be used for more general discussion such as the completeness of the style set. By using our model, we can discuss the efficiency of drawing. Furthermore, we show that the efficiency of drawing heavily depends on the used style set; we give several examples (border patterns) for which the drawing requires many steps to draw in a style set but only few steps in another style set. We discuss the relationship between the possible efficiency and the used styles.

As for the complexity of BDP, our results are somewhat negative. Namely, the problem is $\mathcal{NP}$-hard for the style set of Excel and is also $\mathcal{NP}$-hard for any reasonable subset of styles. We also make some observations on which styles are important for several kinds of instances. Furthermore, we consider the possibility of designing a style set which is better than the Excel set. More concretely, we give an interesting set of styles which is natural, compact, and more efficient than Excel by up to a factor of n for some instances, but unfortunately is not complete. It is apparently important to give approximation algorithms and/or heuristic algorithms, but in this paper, we only give a few basic observations.

Previous Work The most related problem is probably the rectilinear polygon covering problem [11] (also known as the rectilinear picture compression one), which is, given a Boolean matrix, to cover (or to draw) all the 1's with as few rectangles as possible. The problem has a number of important practical

applications, such as in data mining [4], and in the VLSI fabrication process [8]. Thus, it has received a considerable amount of attention and there are a lot of its variants [1, 3, 6, 7]. In [13] (page 433), the time complexities of various polygon covering problems are listed; almost all variations are $\mathcal{NP}$-hard. The difference is that our problem allows us to draw (and also to delete) lines by using several different border styles, which provide numerous possibilities for drawing a picture; this certainly makes the problem harder but more attractive.

2 Models

We first give a formal definition of the terminology (basically we follow that of Excel). A *spreadsheet* (or *worksheet*) is delineated by $n+1$ horizontal and $n+1$ vertical *gridlines* of length n, which are illustrated by dotted lines in this paper. Note that the gridlines are always viewable on the screen; however, any gridline will not be actually drawn or printed on a spreadsheet. A single addressable unit surrounded by two consecutive horizontal gridlines and two consecutive vertical lines is called a *cell*. Let $c(i, j)$ be a cell on the intersection of the ith row from the top and the jth column from the left for $1 \leq i, j \leq n$. For example, reading left-to-right across the spreadsheet on the top row, we encounter $c(1, 1)$ through $c(1, n)$. The intersection of the kth horizontal and the ℓth vertical gridlines forms a *vertex*, (k, ℓ), for $0 \leq k, \ell \leq n$. That is, there are $(n+1)^2$ vertices, $(0, 0)$ through (n, n). Throughout the paper, we assume that n is not too small, for example $n \geq 4$, in order to avoid trivial cases.

A rectangle surrounded by two (not necessarily consecutive) horizontal gridlines and two vertical gridlines is called an *extended cell* or an *e-cell* in short (see Figure 3). An e-cell is specified by an ordered pair of its upper-left cell and lower-right one separated by a colon: For example, $c(1, 2) : c(3, 3)$ defines the e-cell, whose four corners are $(0, 1)$, $(0, 3)$, $(3, 1)$, and $(3, 3)$. Also, as a special case, $c(i, j) : c(i, j)$ denotes a single cell $c(i, j)$. A portion of a (single) gridline is called

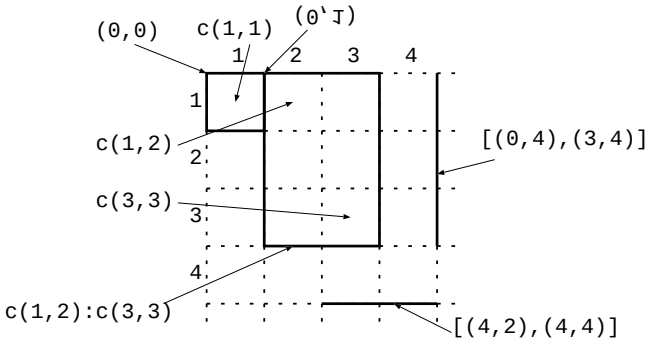


Fig. 3. Extended cell (e-cell)

a *line segment*, which is denoted by its two endpoints, $[(x, y), (x + u, y)]$, if it is vertical and by $[(x, y), (x, y + v)]$ if it is horizontal. Two horizontal line segments that are adjacent, namely $[(x, y_1), (x, y_2)]$ and $[(x, y_2), (x, y_3)]$, are equivalent to the single line segment $[(x, y_1), (x, y_3)]$ (similarly for vertical line segments).

In many situations, we may wish to draw several vertical and horizontal black lines on selective gridlines to enhance the readability of our spreadsheet, or to enclose a selected range of cells with four black lines to highlight the data contained in the range. The Border Drawing Problem (BDP) is formulated in order to model situations that we frequently encounter in spreadsheet applications. An instance of BDP, called a *pattern*, is given as a set of N black line segments, each of which is called a *border*. Given a pattern as an input, we study how to draw it by using a set of border styles defined as follows.

According to the Excel border styles shown in Figure 2, a *border style* (or *style*) is defined as a mapping from $\{1, 2, 3, a, b, c\}$ into $\{B, W, T\}$. It is convenient to use an illustration as in Figure 4 to represent a style, where three horizontal lines correspond to 1, 2 and 3 from top to bottom and three vertical lines to a, b and c from left to right. B, W, and T stand for black, white and transparency, respectively. In the figure, the left-side vertical line is given as a thick straight

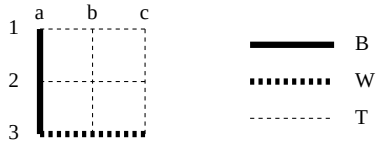


Fig. 4. Border style

line, which means that a is mapped to B in this style. Similarly, 1, 2, b and c are thin dotted lines, which means that they are mapped to T. Finally, 3 is a thick dotted line, meaning that it is mapped to W.

A pattern is drawn by a sequence of operations. A single operation is given by a pair of an e-cell and a style. For example, see Figures 5-(1) and (2). Here we selected the e-cell whose four corners are $(2,1)$, $(2,5)$, $(5,1)$ and $(5,5)$. Thus this e-cell includes four horizontal line segments and five vertical ones, each of which is represented by a symbol in $\{1,2,3,a,b,c\}$; in particular, 1 shows the uppermost horizontal line segment, 3 the bottom horizontal one, 2 the remaining (intermediate) horizontal ones, a the left most vertical one, b the intermediate vertical ones and c the right most vertical one. Now suppose that our style is the one illustrated in Figure 5-(2): Hence, the "colors" of the nine line segments of this e-cell will change as shown in Figure 5-(3) if the original colors of them are all white. Note that B (W, respectively,) requires that the corresponding line segments become black (white, respectively,) regardless of their original colors and T does not change the original colors. We assume that all the gridlines are white at the beginning and that the drawing is completed if the colors of all the borders have become black and all the others remain white.

MS Excel basically allows us to use nine different styles which are given in Figure 6. Styles (1) through (9) are referred to by ℓ, r, t, b, tb, ℓr, o, θ, ϕ, respectively. A set of styles is said to be *complete* if we can draw any pattern by using only styles in the set. It is easy to see that $\{\ell, r, t, b\}$, denoted by S_4, is

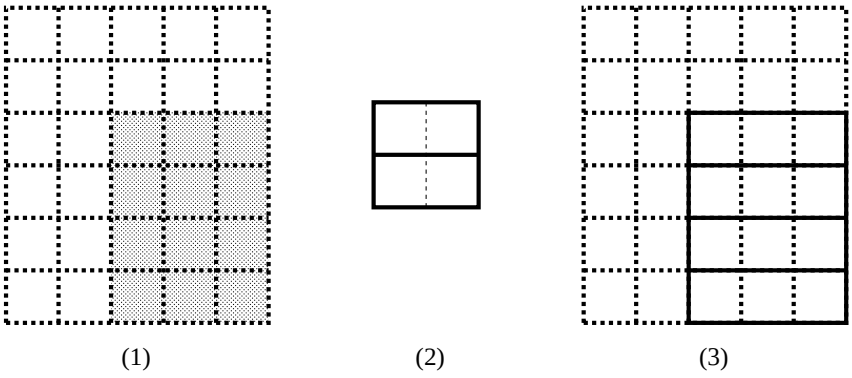


Fig. 5. (1) Original e-cell (2) Style (3) New e-cell

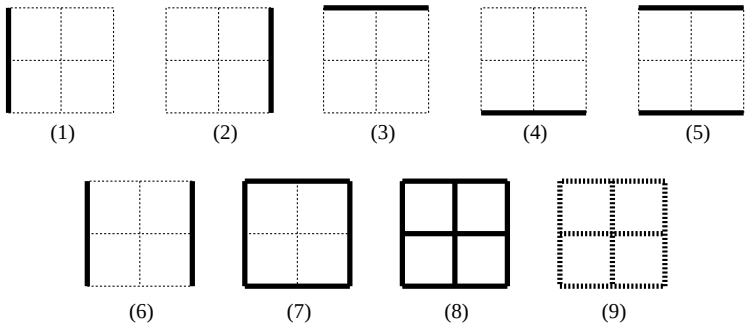
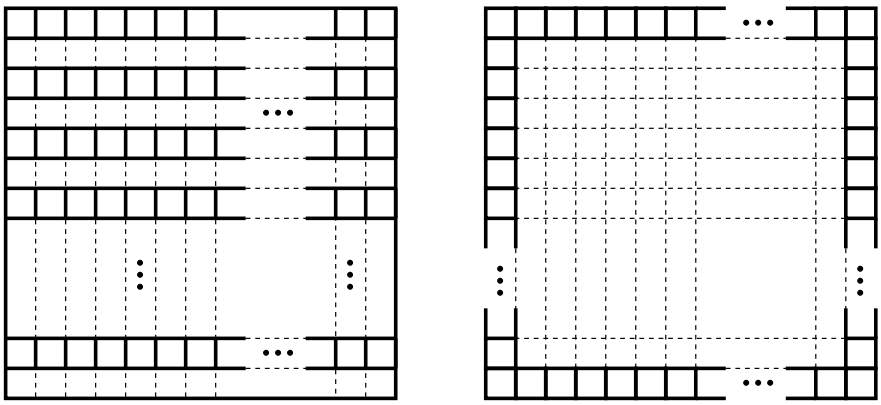


Fig. 6. Excel border styles: (1) ℓ-style (2) r-style (3) t-style (4) b-style (5) tb-style (6) ℓr-style (7) o-style (8) θ-style (9) ϕ-style

complete (and therefore any set including S_4 is also complete). The proof of the following result is straightforward and thus is omitted.

Theorem 1. *S_4, $\{\ell r, t, b, \phi\}$, $\{\ell, r, tb, \phi\}$, $\{\ell r, tb, \phi\}$ are only the minimal complete style sets.* □

Thus, to draw every pattern, for example, we need only four styles $\{\ell, r, t, b\}$. However, some other styles are important when considering the efficiency of the drawing. For example, consider the set $\{\ell, r, t, b, \phi\}$. This set, S_4 plus the style which makes all line segments of the e-cell white, is probably the most convenient



(1) (2)

Fig. 7. (1) Proof of Theorem 2-(i) (2) Proof of Theorem 2-(ii)

for beginners. Note that ϕ is mainly used to correct mistakes, but it is also important for the efficiency. We now show that there is a pattern for which S_4 needs $\Omega(n^2)$ steps, but $O(n)$ steps are enough for $S_4 \cup \{\phi\}$. In particular, consider the pattern illustrated in Figure 7-(1). For simplicity of exposition, we assume that n is divided by 2. The pattern has $\frac{n}{2}$ ladder-shaped tables. Since there exist $\frac{n}{2} \times (n-1)$ disjoint vertical segments, S_4 obviously requires $\Omega(n^2)$ steps. For $S_4 \cup \{\phi\}$, one can see that the following $O(n)$ sequence of operations draws the pattern: (i) Using the first $n-1$ steps, we place $n-1$ vertical line segments, $[(0,1),(n,1)]$ through $[(0,n-1),(n,n-1)]$. (ii) In the next $\frac{n}{2}$ steps, all borders of $\frac{n}{2}-1$ e-cells, $c(2,1):c(2,n)$, $c(4,1):c(4,n)$, through $c(n-2,1):c(n-2,n)$ are deleted by the sequence of the ϕ styles. Here, each ϕ style can disconnect $(n-1)$ segments at a time. (iii) $n+1$ horizontal line segments of length n are added. (iv) Finally, the leftmost and the rightmost vertical line segments of length n are placed.

As observed above, the deletion operation by using the ϕ-style gives us efficient drawing sequences. Also, the θ-style sometimes helps: See Figure 7-(2).

Table 1. The acceleration factors $\alpha_{(A,B)}(n)$ of A (row) for B (column)

	S_4	$S_4 \cup \{\phi\}$	$S_4 \cup \{\theta\}$	$S_4 \cup \{\phi, \theta\}$	$\{\ell r, tb, \phi\}$	$\{\ell r, tb, \phi\} \cup \{\theta\}$
S_4	–	subset	subset	subset	$\Omega(1)$	$\Omega(1)$
$S_4 \cup \{\phi\}$	$\Omega(n)$	–	$\Omega(n)$	subset	$\Omega(1)$	$\Omega(1)$
$S_4 \cup \{\theta\}$	$\Omega(n)$	$\Omega(n)$	–	subset	$\Omega(n)$	$\Omega(1)$
$S_4 \cup \{\phi, \theta\}$	$\Omega(n)$	$\Omega(n)$	$\Omega(n)$	–	$\Omega(n)$	$\Omega(1)$
$\{\ell r, tb, \phi\}$	$\Omega(n)$	$\Omega(1)$	$\Omega(n)$	$\Omega(1)$	–	subset
$\{\ell r, tb, \phi\} \cup \{\theta\}$	$\Omega(n)$	$\Omega(n)$	$\Omega(n)$	$\Omega(1)$	$\Omega(n)$	–

Since there are $(n-3) \times 2 + 4$ vertical and $(n-3) \times 2 + 4$ horizontal borders, $S_4 \cup \{\phi\}$ obviously needs $\Omega(n)$ steps, but only three steps suffice for $S_4 \cup \{o, \theta, \phi\}$ to draw these borders: $(c(1,1) : c(n,n), \theta)$, $(c(2,2) : c(n-1, n-1), \phi)$, $(c(2,2) : c(n-1, n-1), o)$ in this order.

Theorem 2. *(i) There is a pattern for which S_4 requires $\Omega(n^2)$ steps, but $S_4 \cup \{\phi\}$ takes $O(n)$ steps. (ii) There is a pattern for which $S_4 \cup \{\phi\}$ requires $\Omega(n)$ steps, but $S_4 \cup \{o, \theta, \phi\}$ takes $O(1)$ steps.*

To quantify the efficiency of drawing a pattern P by using style sets A or B, we introduce an *acceleration factor* of A for B to draw P, as $\alpha_{(A,B)}(P) = step_B(P)/step_A(P)$,where $step_A(P)$ and $step_B(P)$ are the *minimum* numbers of steps necessary to draw P by using style sets A and B, respectively. For any $n \geq 1$, we define the acceleration factor of A for B as

$$\alpha_{(A,B)}(n) = \max\{\alpha_{(A,B)}(P) \mid P \in \mathcal{P}_n\},$$

where $\mathcal{P}_n$ is the set of all possible patterns in the spreadsheet with size n. Table 1 summarizes the acceleration factors between representative styles that we found.

For some patterns, $S_4 \cup \{\phi, \theta\}$ (and, hence, the full set of the Excel border styles, denoted by S_{Excel}) can be more efficient than S_4 by up to a factor of n. One might ask whether there is a pattern for which this factor is significantly greater than n; the following result shows that the answer is negative.

Theorem 3. *S_{Excel} can be simulated by S_4 with an overhead factor of $O(n)$. That is, $\alpha_{(S_{\text{Excel}},S_4)}(n) = \Theta(n)$.*

Proof. We show that $S_{\text{Excel}} \setminus S_4$ can be simulated by S_4 in $O(n)$ steps. (1) A single use of the tb-style (resp. ℓr-style) in S_{Excel} is achieved by using only a pair of the t- and b-styles (resp. ℓ- and r-styles) in S_4. (2) The o-style in S_{Excel} is equal to a sequence of four styles in S_4. (3) The θ-style can be simulated in $O(n)$ steps because it includes at most n horizontal and at most n vertical line segments. (4) The remaining is the ϕ-style. Here we show that $S_4 \cup \{\phi\}$ can be simulated by S_4 in $O(n)$ steps instead. After several uses of styles in S_4, suppose that the ϕ style is now used. Then, it divides one horizontal (resp. vertical) line segment into at most two pieces, which means that a single operation of the ϕ-style can be simulated by at most two operations of the t-style (resp. ℓ-style) per horizontal (resp. vertical) line segment. Since the ϕ-style cuts at most $2n$ line segments at a time, it can be simulated by S_4 in $O(n)$ steps. □

3 Complexity of Border Drawing Problem

The *border drawing problem with a style set S*, BDP(S), consists in finding a drawing sequence of minimum size for a given pattern where every style is in S. Restating this optimization problem as a decision problem, BDP(S, k), we wish to determine whether a pattern has a drawing sequence with size k. As mentioned in the previous section, this problem is obviously in P for the set S_4. In this section we show that the problem becomes intractable if we use the set $S_5 = S_4 \cup \{\phi\}$, the most interesting subset as mentioned in the previous section.

Theorem 4. *BDP(S_5, k) is $\mathcal{NP}$-complete.*

Proof. It is easy to show that BDP(S_5, k) is in $\mathcal{NP}$. Its $\mathcal{NP}$-hardness is proved by reducing the $\mathcal{NP}$-complete *rectilinear picture compression problem* (RPC in short) [11] to BDP(S_5, k). The RPC problem asks whether given an $m \times m$

matrix M of 0's and 1's and a positive integer q, there exists a collection of q or fewer rectangles that cover precisely those entries in M that are 1's. That is, we have to show that for a given $m \times m$ matrix M we can construct a pattern P such that P can be drawn by a drawing sequence of length k or shorter if and only if there exists a collection of q or fewer rectangles that cover precisely those entries in M that are 1's.

First of all, the $m \times m$ matrix M is modified to an $(m+2) \times (m+2)$ matrix M' by padding one row of $(m+2)$ 0's on the top row, one row of $(m+2)$ 0's under the bottom row, one column of $(m+2)$ 0's in the leftmost, and one column of $(m+2)$ 0's in the rightmost. Namely, the new matrix M' is obtained by surrounding the original matrix M with 0's. For example, if

$$M = \begin{pmatrix} 0\,1\,1\,1 \\ 0\,1\,0\,1 \\ 1\,1\,1\,1 \\ 1\,0\,0\,0 \end{pmatrix}, \qquad \text{then} \qquad M' = \begin{pmatrix} 0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,1\,1\,0 \\ 0\,0\,1\,0\,1\,0 \\ 0\,1\,1\,1\,1\,0 \\ 0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0 \end{pmatrix}.$$

We next prepare a two-dimensional grid of $(m+2) \times 3$ rows and $(m+2) \times 3$ columns, and place black borders on all gridlines except for its outline. Then, if the entry at the ith row and jth column of M' is 1, then we obtain borders by placing white lines (or deleting the black borders drawn above) on all the outside and inside black borders of nine cells, $c(3i-2, 3j-2)$, $c(3i-2, 3j-1)$, $c(3i-2, 3j)$, $c(3i-1, 3j-2)$, $c(3i-1, 3j-1)$, $c(3i-1, 3j)$, $c(3i, 3j-2)$, $c(3i, 3j-1)$, $c(3i, 3j)$ for every $1 \leq i, j \leq m+2$.

Finally, by surrounding the above grid with $(3m+7) \times 2$ horizontal and $(3m+7) \times 2$ vertical black borders of length one, called *scraps*, we obtain our reduced pattern P from the instance of RPC. Figure 8 illustrates P, that has $(3m+10) \times (3m+10)$ cells.

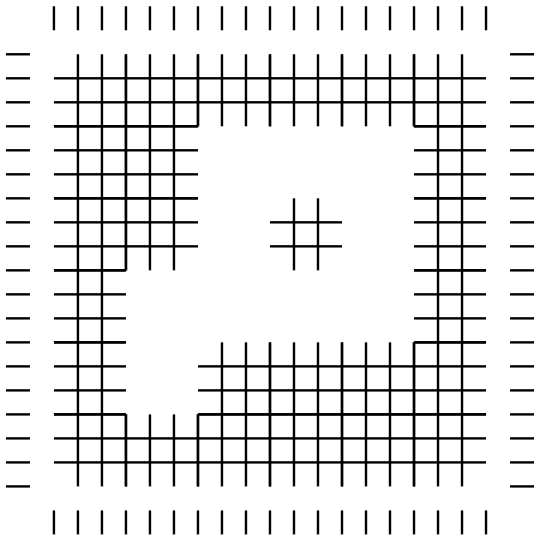


Fig. 8. Pattern P

As shown below, the reduced pattern P has a feasible drawing sequence of length $k = q + 6m + 18$ or shorter if and only if all 1's in M are covered by a collection of q or fewer rectangles.

($\Leftarrow$) We can actually give a drawing sequence of $q+6m+18$ steps as follows: (i) We place $3m+7$ horizontal black borders of length $3m+10$, i.e., all of them go across from the left-end to the right-end, by using the first $3m + 7$ steps. (ii) Also, $3m + 7$ vertical borders of length $3m + 10$ are placed in the next $3m + 7$ steps. (iii) By using the ϕ style, all black borders of the four e-cells $c(2, 2) : c(2, 3m + 9)$, $c(3m + 9, 2) : c(3m + 9, 3m + 9)$, $c(2, 2) : c(3m + 9, 2)$, and $c(3m+9, 2) : c(3m+9, 3m+9)$ are completely deleted. (iv) Finally, according to the rectangle covering of RPC, we delete the black borders again by using the ϕ style in at most q steps.

($\Rightarrow$) Suppose that the pattern P can be drawn in at most $k = q + 6m + 18$ steps. Our first claim is that out of this $k = q + 6m + 18$ steps we need $6m + 18$ steps only to draw the $(3m + 7) \times 4$ scraps and the borders corresponding to

the 0's in M' padded to the original matrix M in its surrounding area. (Since we have so many scraps and at most two scraps are drawn in a single step, one can see easily that the procedures (i) through (iii) as above is the only way to draw this portion of the pattern.) Moreover, after drawing these scraps and the padded ones in this number of steps, all the gridlines of the central part of the figure must be black. (This is obvious if we have no choice other than using the procedures (i) through (iii).)

Thus, we now have to complete the drawing with the remaining q steps. Obviously we have to use the ϕ style for all those steps to make the "holes" in the central part, but that can be simulated by the same number of rectangles which cover all the 1's of the matrix M. This completes the proof. □

Let S'_5 denote $\{\ell, r, t, b, \theta\}$, namely ϕ is replaced by θ in S_5. Then the proof of $\mathcal{NP}$-hardness for BDP(S'_5, k) is easier than above, since we can simulate the RPC problem almost directly. Also one can see easily that BDP is $\mathcal{NP}$-hard if its style set includes $\{\ell, r, t, b\}$ and θ or ϕ (actually we do not need all the four basic styles). Finally, by slightly modifying the proof of the previous theorem, we can also show the following result.

Theorem 5. *BDP(S_{Excel}, k) is $\mathcal{NP}$-complete.*

Now it is natural to consider approximation algorithms or heuristic algorithms for BDP. Among the several intractable cases, the first one to be considered is S'_5, because an approximation algorithm for BDP(S'_5) might be a prototype for other cases. (The reason will be mentioned later.)

Consider a pattern as an input for BDP(S'_5) illustrated in Figure 9-(1). A cell surrounded by black borders is called a *black-cell*; otherwise *gray-cell*. For example, $c(1, 2)$ and $c(2, 2)$ are black-cells, and $c(1, 1)$ and $c(1, 3)$ are gray-cells. A sequence of consecutive vertically aligned black-cells bounded by gray-cells on the top and the bottom constitutes a *strip*. For example, see Figure 9-(2); the

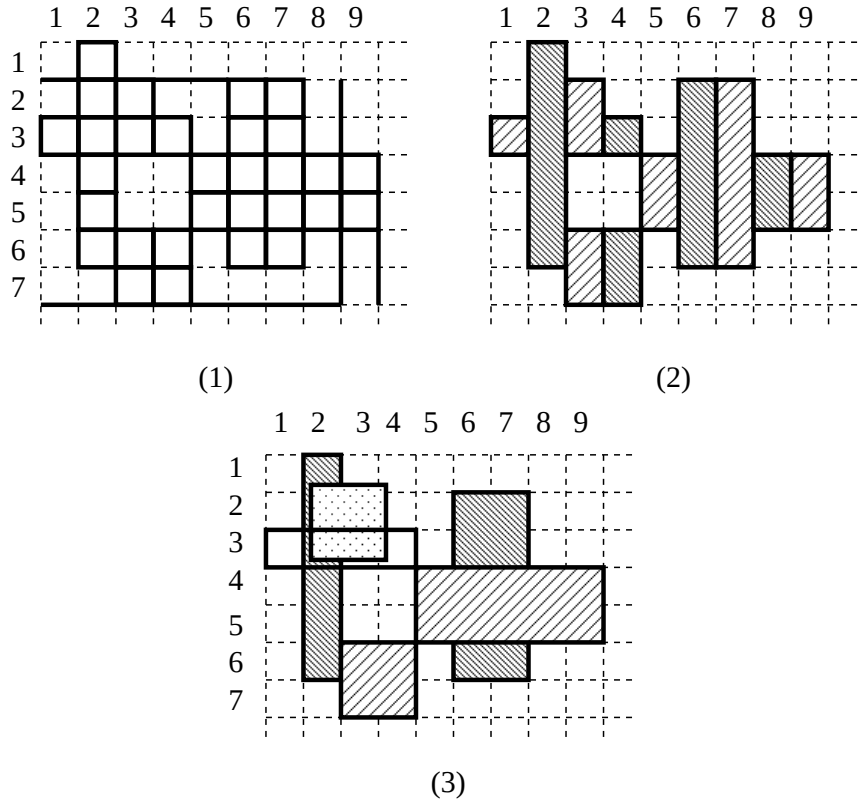


Fig. 9. (1) black-cells, gray-cells, (2) strips, and (3) associated rectangles

pattern has 11 strips, $c(3,1)$, $c(1,2) : c(6,2)$, $c(2,3) : c(3,3)$, and so on. Two strips $c(i_1,k_1) : c(j_1,k_1)$ and $c(i_2,k_2) : c(j_2,k_2)$ are said to be *independent* if $i_1 \neq i_2$ or $j_1 \neq j_2$ holds. For example, six stripes $c(1,2) : c(6,2)$, $c(2,3) : c(3,3)$, $c(6,3) : c(7,3)$, $c(3,4)$, $c(4,5) : c(5,5)$, $c(2,6) : c(6,6)$ are *mutually independent*. For each strip $c(i,k) : c(j,k)$, we define its *associated rectangle* to be the unique rectangle that covers this strip, and extends as far as possible to the left and to the right, still containing only black-cells. As shown in Figure 9-(3), there are six rectangles associated with the mutually independent six strips.

The basic idea of our approximation algorithm `ALG(`S'_5`)` for BDP(S'_5) is quite simple: First we draw by the ℓ- or r-style (resp., t- or b-style) every vertical (resp.,

horizontal) black line segment which separates some pair of consecutive horizontally (resp. vertically) aligned gray-cells. For example, a vertical line segment $[(1,8),(7,8)]$ separates two gray-cells $c(3,8)$ and $c(3,9)$ and thus it is drawn by the ℓ-style. Similarly, we draw a horizontal line segment $[(1,0),(1,7)]$ by the t-style since it separates two gray-cells, say, $c(1,5) : c(2,5)$. Notice that these draws are indispensable, because other styles cannot draw the line segments. Then only black-cells are left. To draw the black-cells, it is better to use the θ-style. Since drawing the black-cells by the θ-style is essentially the same as the RPC problem, we run a procedure similar to the one introduced in [9] as a subroutine; the approximation factor of this procedure is $O(\sqrt{\log n})$. Here is a description of `ALG(`S'_5`)`:

`Algorithm ALG(`S'_5`)`

Step 1. Draw every vertical (resp. horizontal) black line segment which separates some pair of consecutive horizontally (resp. vertically) aligned gray cells by using ℓ- or r-style (resp. t- or b-style) in column-first order (resp. row-first order).

Step 2. Find rectangles associated with mutually independent strips, and draw each of the associated rectangles by using the θ-style.

Theorem 6. *Algorithm* `ALG(`S'_5`)` *achieves an approximation ratio of* $O(\sqrt{\log n})$ *for BDP(*S'_5*).* □

If we can use ϕ as well, then what we should do first is to look for "holes" for which using ϕ helps. Then we once "fill" those holes and apply the above greedy algorithm. After that those holes are dug again by using ϕ. Unfortunately we do not know the approximation factor of this algorithm, whose analysis appears to be difficult.

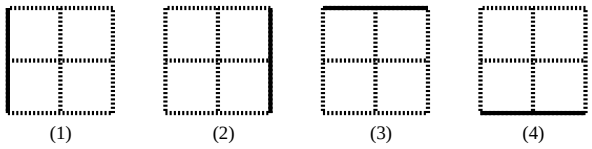


Fig. 10. Black-White styles: (1) ℓ^{bw}-style (2) r^{bw}-style (3) t^{bw}-style (4) b^{bw}-style

4 Border Styles with Black and White

Recall that all Excel styles, except for the ϕ-style, have no white segments and then do not turn black borders into white ones. Only the ϕ-style deletes black borders we have drawn previously or turns black lines back into white ones. As shown in the previous section, this deletion capability gives us efficient drawing sequences. In this section we consider styles which include both Black and White at the same time, as illustrated in Figure 10. In the case of the ℓ^{bw}-style in Figure 10-(1), all three horizontal line segments 1, 2, and 3 (from top to bottom) are mapped to W, and three vertical ones a, b, and c are mapped to B, W, and W, respectively. The r^{bw}-style maps 1, 2, 3, a, b, and c into W, W, W, W, W, and B, respectively. The t^{bw} and b^{bw}-styles are similar. Let S_4^{bw} be $\{\ell^{bw}, r^{bw}, t^{bw}, b^{bw}\}$.

In this section, we assume that the given pattern does not include the gridlines of the boundary of spreadsheets. The reason is as follows: For example, there are no cells above the top horizontal gridline of the sheet itself. Therefore, any border on this gridline cannot be drawn by the b^{bw} style. However, all other horizontal borders can be drawn by that style. One can see that the above assumption excludes such a trivial incompleteness of the style set.

As shown in a moment, S_4^{bw} is sometimes very efficient, which indicates some possibilities of improving the style set of Excel.

Proposition 1. *There is a pattern for which S_4 needs $\Omega(n^2)$ steps, but S_4^{bw} takes $O(n)$ steps.*

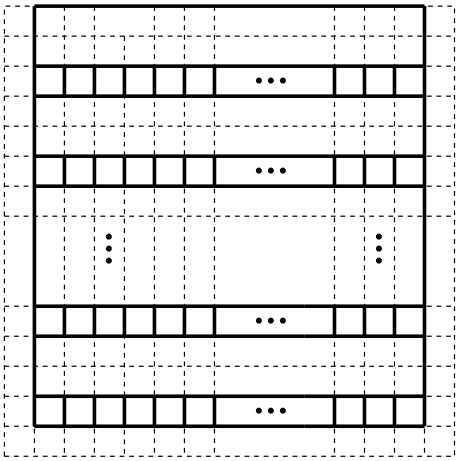


Fig. 11. Proof of Proposition 1

Proof. Figure 11 illustrates one of such patterns. S_4 needs $\Omega(n^2)$ since there are $\Omega(n^2)$ vertical segments. Here is a rough description of the drawing sequence for S_4^{bw}: We first place all vertical n-length borders in $O(n)$ steps. Then, in order to cut them, we place around $\frac{2n}{3}$ horizontal borders by using the t^{bw} and the b^{bw} styles in $O(n)$ steps. Finally, two outer vertical borders are added. □

For several patterns, S_4^{bw} is more efficient than S_4 but, unfortunately, there is a large class of patterns for which S_4^{bw} has no feasible drawing sequences (other than the trivial ones mentioned at the beginning of this section). Here are some definitions: As shown before, in the case of S_4^{bw}, the order of the drawing sequence is very critical and strongly affected by the pattern's layout. In order to discuss the drawing-order of borders, we associate a pattern with an undirected graph, defined below. In the following, we apply each border style in S_4^{bw} only on unit cell, which simplifies the explanation. Actually, this restriction may affect the number of drawing steps, but not the (in)completeness of S_4^{bw}.

For a given pattern, we say (distinct) unit cells are *neighbors* to each other if they share a border. For example, cell $c(x, y)$ and $c(x + 1, y)$ are neighbors if

the former has a bottom border, or equivalently the latter a top one. Given a pattern P of borders $\{b_1, b_2, \cdots, b_m\}$, the *neighborhood graph* $G(P)$ is a graph with node set $V(P)$ and edge set $E(P)$, where

$$V(P) = \{u_{i,j} \mid 1 \leq i, j \leq n\} \quad \text{and,}$$
$$E(P) = \{(u_{i,j}, u_{k,\ell}) \mid c(i,j) \text{ and } c(k,\ell) \text{ are neighbors}\}.$$

Note that each node $u_{i,j}$ corresponds to unit cell $c(i,j)$.

Theorem 7. *Pattern P has no feasible drawing sequences on S_4^{bw} if and only if its neighborhood graph $G(P)$ contains a cycle.*

Proof. ($\Rightarrow$) Suppose that $G(P)$ does not have a cycle, i.e., $G(P)$ is a tree. Take an arbitrary node as root r, then find paths from r to its leaves. According to the paths, we add direction information $(u_1 \rightarrow u_2)$ to each edge $e = (u_1, u_2)$, which means that u_1 is a tail node and u_2 a head one. If, for example, the edge between $u_{1,2}$ and $u_{2,2}$ has direction $(u_{1,2} \rightarrow u_{2,2})$, then the operation $(c(2,2) : c(2,2), t^{bw})$ is executed. Due to the orientation, the in-degree of each node is at most 1, which implies that a border of each node (or cell) once drawn will not be erased. Therefore, according to the tree-orientation, we can find at least one drawing sequence for P.

($\Leftarrow$) We just give a sketch of the proof. We show, by contradiction, that if $G(P)$ contains a cycle, then P cannot be drawn by S_4^{bw}. Suppose that P can be drawn by S_4^{bw}. This implies that S_4^{bw} has a finite drawing sequence of styles for a pattern corresponding to a simple cycle C, because drawing sequences that are noncontiguous for the cycle always leave some borders undrawn. Note that if we apply one of the *bw*-type border styles, then one border is added but at the same time three other ones are deleted. Hence, the node corresponding to the cell where the last style of the drawing sequence is placed must be a leaf, which is a contradiction. □

From the above theorem, every subset of S_4^{bw} is also not complete. Since we have a simple characterization of the drawability and the incompleteness means the number of patterns which can be drawn is small, one might think that, for example, BDP(S_4^{bw}, k) becomes tractable. However, it still remains $\mathcal{NP}$-complete even for BDP($\{r^{bw}, t^{bw}\}, k$).

Theorem 8. *BDP(S, k) is $\mathcal{NP}$-complete for any of the following S: $\{r^{bw}, t^{bw}\}$, $\{r^{bw}, b^{bw}\}$, $\{\ell^{bw}, t^{bw}\}$, $\{\ell^{bw}, b^{bw}\}$, $S_4^{bw}\backslash\{\ell^{bw}\}$, $S_4^{bw}\backslash\{r^{bw}\}$, $S_4^{bw}\backslash\{t^{bw}\}$, $S_4^{bw}\backslash\{b^{bw}\}$ and S_4^{bw}.* □

5 Conclusion and Discussion

In this paper, we consider the problem of drawing border patterns by a typical spreadsheet software application, e.g., Excel. We give a formal model for the problem, under which we can discuss the drawability, the completeness, the efficiency of drawing, the complexity and algorithms. The hardness of our problem is related to the rectilinear picture compression problem (RPC), but it appears in two ways: The difficulty of finding an optimal drawing black patterns on a white canvas and that of finding an optimal drawing white patterns on a black canvas. Namely, our problem has a multiply layered structure of RPCs in a sense.

To consider this interesting feature of the problem, in Section 4, we introduce new styles that contain both black and white segments (called Black-White styles), and discuss the efficiency and the complexity for style sets containing Black-White styles. By using Black-White styles, we can draw black patterns on a white canvas and white patterns on a black canvas simultaneously; the multiple layers of RPCs can be solved at the same time.

To explore the feature of the problem, we can consider a drawing function. In the current model, if two (or more) styles are put on a cell, the result is determined by the overwriting manner. (Putting transparent T means that nothing

is put.) That is, if we represent the result by function f, we have $f(W, B) = B$, $f(B, W) = W$ and $f(B, T) = f(B, B) = B$ and $f(W, T) = f(W, W) = W$. It might be interesting to extend the rule to a more general logical operation, e.g., AND, OR and XOR. We have a few results about such style sets. For example, concerning a style set containing only XOR operation, transitivity of two patterns can be determined in linear time. Considering more general operations may be an interesting issue.

Acknowledgments

We would like to thank anonymous referees whose comments help to improve the presentation of the paper.

This work is in part supported by KEKENHI, No. 16092101, 16092215, 16092223, 16300002, 17700022, 18300004 and 18700014.

References

1. P. Berman and B. DasGupta, "Complexities of efficient solutions of rectilinear polygon cover problems," *Algorithmica*, Vol.17 (4), 331–356 (1997).
2. K.S. Booth and G.S. Lueker, "Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms," *J. of Computer and System Science*, Vol.13, 335–379 (1976).
3. J.C. Culberson and R.A. Reckhow, "Covering polygons is hard," *J. of Algorithms*, Vol.17, 2–44 (1994).
4. J. Edmonds, J. Gryz, D. Liang, and R.J. Miller, "Mining for empty rectangles in large data sets," *Theoretical Computer Science*, Vol.296, 435–452 (2003).
5. M.R. Garey and D.S. Johnson, *Computers and Intractability - A Guide to the Theory of $\mathcal{NP}$-Completeness*, W.H .Freeman & Co. (1979).
6. D.S. Greenberg and S.C. Istrail, "Physical mapping by STS hybridization: Algorithmic strategies and the challenge of software evaluation," *J. Computational Biology*, Vol.2, 219–274 (1995).
7. J. Gudmundsson and C. Levcopoulos, "Close approximations of minimum rectangular coverings," *J. of Combinatorial Optimization*, Vol.3 (4), 437–452 (1999).
8. A. Hegedüs, "Algorithms for covering polygons by rectangles," *Computer Aided Geometric Design*, Vol.14, 257–260 (1982).
9. V. S. Anil Kumar and H. Ramesh, "Covering Rectilinear Polygons with Axis-Parallel Rectangles", *SIAM Journal on Computing*, Vol.32 (6), 1509 – 1541 (2003).
10. P.M. Magwene, P. Lizardi and J. Kim, "Reconstructing the temporal ordering of biological samples using microarray data," *Bioinformatics*, Vol.19 (7), 842–850 (2003).

11. W.J. Masek, "Some $\mathcal{NP}$-complete set covering problems," MIT, Cambridge, MA, *unpublished manuscript* (1978). Referenced in [5].
12. D. Reuhman, "A short route to tgif pictures," `ftp://bourbon.usc.edu/pub/tgif/contrib/tgifintro/` (2002).
13. S. Suri, "Polygons," in *Handbook of Discrete and Computational Geometry*, J.E. Goodman and J. O'Rourke Eds, CRC Press (1997).
14. *Xfig User Manual (ver. 3.2.5)*, `http://www.xfig.org/userman/`.