Analysis of Weave Structures, Part 1: Introduction

Fabric analysis [1] can determine the structure of a fabric — its interlacement pattern. Further analysis of the structure can provide more information, including detection of possible errors in the analysis or in the weave itself.

Representing Interlacement Patterns

In what follows, we'll use binary arrays [2, 3] to represent interlacement patterns, with a 1 where the warp is on top and a 0 where the weft is on top. Figure 1 shows a drawdown from Oelsner [4] and Figure 2 shows the corresponding binary array.

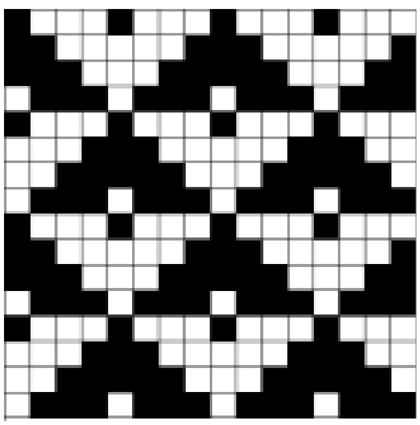

Figure 1. Drawdown

```
1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1
1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1
0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0
0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 0
0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1
1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1
0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0
0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 0
0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
```

Figure 2. Binary Array

Finding the Unit Motif

The first step in the analysis of an interlacement pattern is to determine its *unit motif*, the smallest subpattern from which the entire pattern can be constructed by repetition. All subsequent analyses will be done on unit motifs — and depend for their usefulness on that.

In many cases, finding the unit motif can be done by inspection, although it is easy to make a mistake, particularly by identifying a subpattern from which the entire structure can be constructed by repetition but which is not the smallest one.

The unit motif can be found by determining separately the shortest repeat lengths for the rows and columns of the pattern.

Here we need to explain the nature of the analysis methods we use. When done by hand, tasks like finding repeats involve seeing patterns and using various devices, like marking positions, when that's necessary. In this article, we'll describe how such tasks might be done by a simple, straightforward computer program. This approach often involves doing things that a human being would not need to do and which may seem silly and simple-minded. Indeed, they are simple-minded — but they also can be done very quickly by a computer and without error. So, in this article we'll often think like a programmer, writing down in words what would make up a program.

Finding the repeat for rows and columns is straightforward, if a bit tedious. Both procedures are the same; we'll refer to rows and columns as lines.

1. Start with the first line of the binary array for the pattern.
2. Form an initial segment by taking the first binary digit from the beginning of the line.
3. Repeat the initial segment to the length of the line, discarding any run-over.
4. Compare the repeated segment to the line.
5. If the two match, the length of the initial segment is the repeat length for the line

December 16, 2001; last revised August 1, 2004

and the process for the line is complete; go to the next line and continue with Step 2.

6. Otherwise, if the length of the initial segment is less than the length of the line, increase the length of the initial segment by 1 and continue with Step 3.
7. If the length of the initial segment is the length of the line, the width of the unit motif is the length of the line and no other lines need to be processed.
8. Otherwise, go to the next line if there is one and continue with Step 2.
9. If there are no more lines, the process is complete.

When done by hand, there are obvious shortcuts to shorten the process. In the first place, it should not be necessary to start with an initial segment of length 1, since a repeat of length one would be obvious and, of course, should never occur in an interlacement pattern. This procedure above could, in fact, start with an initial segment of length 2. But if a program is doing all the processing, it is wise to take into account the possibility of lines without interlacement. (Adequate program testing would cover this case.) And don't forget the possibility of extending the procedure to profile block patterns.

Similarly, a repeat of length 2 would be obvious and other lengths often can be skipped because it is clear they will not work. But in taking such obvious shortcuts, you are using very sophisticated mechanisms — the human mind and visual system, as well as experience in using them. A computer program plods instead.

Figure 3 shows the periods for the rows and columns of the binary array of Figure 2.

																	rows
	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	*4*
	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	1	*8*
	1	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	*8*
	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	*4*
	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	*4*
	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0	0	*8*
	0	0	1	1	1	1	1	0	0	0	1	1	1	1	1	0	*8*
	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	*8*
	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	*4*
	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	1	*8*
	1	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	*8*
	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	*4*
	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	*4*
	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0	0	*8*
	0	0	1	1	1	1	1	0	0	0	1	1	1	1	1	0	*8*
	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	*4*
columns	*8*	*8*	*8*	*8*	*8*	*4*	*8*	*4*	*8*	*4*	*8*	*8*	*8*	*8*	*8*	*8*	

Figure 3. Repeat Periods

When the repeat lengths of all rows have been determined and if all are shorter than the width of the pattern itself, the width of the unit motif is the *least common multiple* (*lcm*) of the lengths of all row repeats and similarly for the column repeats.

The *lcm* of a list of integers is the smallest integer that all divide evenly. For example, the *lcm* of 2, 4, and 8 is 8, while the *lcm* 3 and 5 is 15.

For Figure 3, the row repeat lengths are 4 and 8, whose *lcm* is 8 and similarly for the columns. Consequently the unit motif is 8 × 8. Any 8 × 8 subpattern will do; we'll use the one starting at the upper left. See Figures 4 and 5.

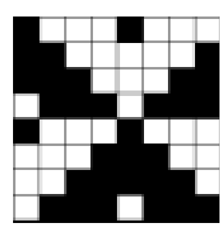

Figure 4. Unit Motif Drawdown

```
1 0 0 0 1 0 0 0
1 1 0 0 0 0 0 1
1 1 1 0 0 0 1 1
0 1 1 1 0 1 1 1
1 0 0 0 1 0 0 0
0 0 0 1 1 1 0 0
0 0 1 1 1 1 1 0
0 1 1 1 0 1 1 1
```

Figure 5. Unit Motif Array

Comment: There is a somewhat subtle issue in determining repeats — whether or not the pattern is composed of a full number of repeats or if there are partial repeats at the edges. The procedure above allows for the latter possibility and is disguised in the remark "discarding any run-over" in Step 3.

If the pattern is known to be an integral number of repeats of a unit motif, only segment lengths that evenly divide the lengths of lines need to be considered. For example, the interlacement pattern in our example is 16 × 16. Therefore, only lengths of 1, 2, 4, and 8 would need be tried.

This situation considerably reduces the number of cases that have to be tried and can make a measurable difference in how long it takes, especially for large patterns.

Indeed, it often is the case that the dimensions of a pattern are integral numbers of the dimensions of a unit motif. On the other hand, assuming there may be partial repeats at the edges is risky. For example, if a 12 × 12 pattern has an apparent 8 × 8 unit motif, there is no way to know for certain that the four rows and columns beyond the presumed unit motif accidentally match and the intention is to repeat a 12 × 12 unit motif. Here, the decision should be made on the basis of external information and experience.

Line Patterns

The rest of this article is concerned with line patterns — patterns of 0s and 1s that occur in the binary arrays for weave structures.

Lines can be viewed in two ways: as bit patterns and as blocks of consecutive 0s and 1s.

The first row of the unit motif for our example has the bit pattern 10001000. The corresponding block-length pattern is $\underline{1}$ 3 $\underline{1}$ 3 — one 1, three 0s, one 1, three 0s. The underscores distinguish 1-blocks from 0-blocks. This line occurs twice among the rows.

The second row has the bit pattern 11000001 and the block-length pattern $\underline{2}$ 5 $\underline{1}$. This line does not occur elsewhere among the rows, but a *circular permutation* of it occurs in row 6.

A circular permutation moves bits off one end of a line and puts them on the other end. For example, the circular permutation of 110000$\underline{01}$ by 2 to the right produces $\underline{01}$110000, where the underscores show the moved bits. Permutations may be to the right (positive), to the left (negative), or zero (no change). For every negative permutation, there is a corresponding positive permutation. For the example above, permutation by –6 is the same as permutation by 2: $\underline{110000}$01 becomes 01$\underline{110000}$. We will use only nonnegative permutation amounts.

Circular permutations form the heart of line analysis. Lines that are circular permutations of each other are considered structurally equivalent, allowing many interlacement structures to be characterized by a few structurally equivalent lines.

In order to deal with structurally equivalent lines, we define a *canonical form* for lines — a particular circular permutation that has special properties and from which all other structurally equivalent lines can be obtained.

The canonical form basically is the circular permutation that puts the most 1s at the left. It can be determined by forming all the circular permutations of a line (including by 0) and sorting the results. The largest is the canonical form.

For example, Figure 6 shows the circular permutations of row 4 of our unit motif example, and Figure 7 shows the results of sorting, with the canonical form indicated by an arrow.

		amount
0 1 1 1 0 1 1 1		*0*
1 1 1 0 1 1 1 0		*1*
1 1 0 1 1 1 0 1		*2*
1 0 1 1 1 0 1 1		*3*
0 1 1 1 0 1 1 1		*4*
1 1 1 0 1 1 1 0		*5*
1 1 0 1 1 1 0 1		*6*
1 0 1 1 1 0 1 1		*7*

Figure 6. Circular Permutations of Row 4

1 1 1 0 1 1 1 0 ←
1 1 1 0 1 1 1 0
1 1 0 1 1 1 0 1
1 1 0 1 1 1 0 1
1 0 1 1 1 0 1 1
1 0 1 1 1 0 1 1
0 1 1 1 0 1 1 1
0 1 1 1 0 1 1 1

Figure 7. Sorted Permutations

Since the row consists of two repeats of a length-4 pattern, each permuted row pattern appears twice.

The canonical patterns for row 4 therefore are

bits	*blocks*
11101110	$\underline{3}$ 1 $\underline{3}$ 1

There are altogether four different canonical row patterns:

A	10001000	$\underline{1}$ 3 $\underline{1}$ 3
B	11100000	$\underline{3}$ 5
C	11111000	$\underline{5}$ 3
D	11101110	$\underline{3}$ 1 $\underline{3}$ 1

There are two important properties of canonical block-length patterns:

1. They always start with a 1-block and it is the longest 1-block.
2. There are an even number of block-length patterns, alternating between 1-blocks and 0-blocks.

Exceptions to these rules occur for lines that consist entirely of 0s or 1s. These do not occur in real interlacement patterns, although they may occur in block patterns. Such lines are *improper*. They will be considered later.

Figure 8 shows the blocks for the rows of the unit motif together with the amounts of permutation.

row		block	*amount*
1	1 0 0 0 1 0 0 0	A	*0*
2	1 1 0 0 0 0 0 1	B	*7*
3	1 1 1 0 0 0 1 1	C	*6*
4	0 1 1 1 0 1 1 1	D	*1*
5	1 0 0 0 1 0 0 0	A	*0*
6	0 0 0 1 1 1 0 0	B	*3*
7	0 0 1 1 1 1 1 0	C	*2*
8	0 1 1 1 0 1 1 1	D	*1*

Figure 8. Row Patterns

The row block-pattern sequence for the rows is ABCDABCD and the permutation-amount sequence is *07610321.*

This characterization of the rows tells us something about the interlacement. The first four rows have the same canonical patterns as the last four — that is, the block sequence is a repeat of ABCD. In addition, blocks A and D appear with the same rotation amounts, while blocks B and C have different rotation amounts in their two appearances.

There are three different canonical column patterns:

E	11101000	$\underline{3}$ 1 $\underline{1}$ 3
F	11100010	$\underline{3}$ 3 $\underline{1}$ 1
G	11001100	$\underline{2}$ 2 $\underline{2}$ 2

Note that these are all different from the canonical row patterns.

Figure 9 shows the blocks for the rows of the unit motif with the amounts of permutation.

column	*1 2 3 4 5 6 7 8*
	1 0 0 0 1 0 0 0
	1 1 0 0 0 0 0 1
	1 1 1 0 0 0 1 1
	0 1 1 1 0 1 1 1
	1 0 0 0 1 0 0 0
	0 0 0 1 1 1 0 0
	0 0 1 1 1 1 1 0
	0 1 1 1 0 1 1 1
block	E F G F E F G F
amount	*0 1 2 5 4 5 2 1*

Figure 9. Column Patterns

Here again, the block sequence is a repeat. Note that the permutation amounts, except for the first, form a palindromic sequence — one that reads the same way forwards and backwards.

Figure 10 shows the complete line analysis.

row		*block*	*amount*
1	1 0 0 0 1 0 0 0	A	*0*
2	1 1 0 0 0 0 0 1	B	*7*
3	1 1 1 0 0 0 1 1	C	*6*
4	0 1 1 1 0 1 1 1	D	*1*
5	1 0 0 0 1 0 0 0	A	*0*
6	0 0 0 1 1 1 0 0	B	*3*
7	0 0 1 1 1 1 1 0	C	*2*
8	0 1 1 1 0 1 1 1	D	*1*
block	E F G F E F G F		
amount	*0 1 2 5 4 5 2 1*		

Figure 10. Complete Line Analysis

Comments

This article introduces the basic concepts for a kind of structure analysis that gives insights and useful information for many kinds of interlacements.

It works best for relatively simple interlacement patterns that are geometrical in nature, as many are.

Topics to come include the characteristics of line analysis for certain kinds of weaves, such as plain weave, twills, satins, basket weaves, and so on. This will provide insight into to the common properties of such weaves.

Another topic is detecting errors in interlacement patterns. Do you see how that might be done?

In a more formal vein, line analysis will be used to detect interlacement patterns that are equivalent in various ways.

References

1. Barker, Aldred F. and Eber Midgley, *Analysis of Woven Fabrics*, Scott, Greenwood & Son, 1914.

2. Griswold, Ralph E., “From Drawdown to Draft — A Programmer’s View”, 2004: http://www.cs.arizona.edu/patterns/weaving/webdocs/gre_fa.pdf

3. Griswold, Ralph E., “Designing Weave Structures Using Boolean Operations, Part 1”, 2002: http://www.cs.arizona.edu/patterns/weaving/webdocs/gre_bol1.pdf

4. Oelsner, G. H., *A Handbook of Weaves*, Dover Reprint, 1952. (Figure 628)

Ralph E. Griswold

Department of Computer Science
The University of Arizona
Tucson, Arizona