SOME IMPLICATIONS OF DYNAMIC STRUCTURAL ANALYSIS

J. A. Hoskins and W. D. Hoskins

Department of Computer Science

University of Manitoba
Winnipeg, Manitoba

1. Introduction.

The interlacement structure exhibited by a woven textile has traditionally, and conveniently, been represented by a binary array. Graphically, this has taken the form of a cartesian grid with cells coloured either black or white [Figure 1]. This binary structural array can, in fact be considered as the product of three matrix factors. These factors are normally also binary matrices and, in most cases, their product arises as a result of conventional matrix multiplication [1]. The factorization process is of great practical interest, since these matrix factors correspond to parameters for the production of the corresponding structure [2], as well as being of considerable theoretical interest. The development of fast efficient factoring algorithms has been considered and has resulted in the processes described in [3,4].

In analysing an interlacement structure in this way, the data structure is first created using interactive graphical input to colour the cells of the corresponding grid [5,6]. When the data structure is complete, the factorization algorithm is invoked and

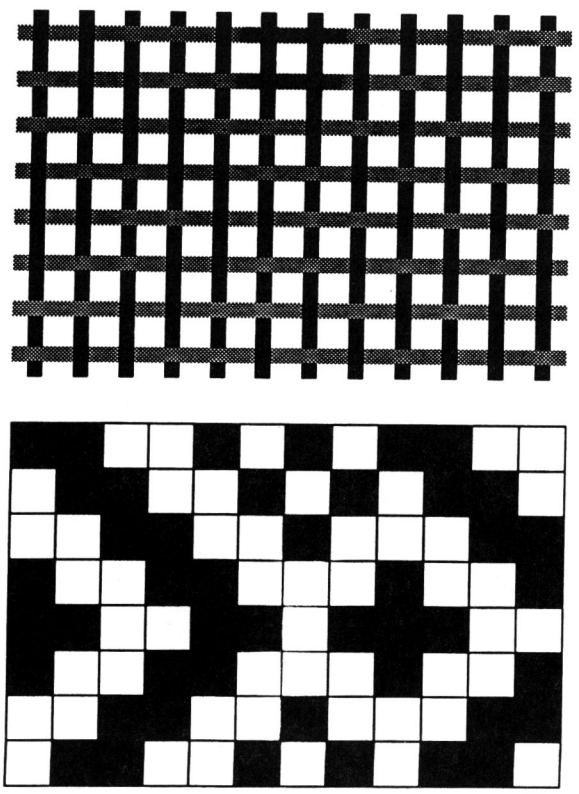

FIGURE 1

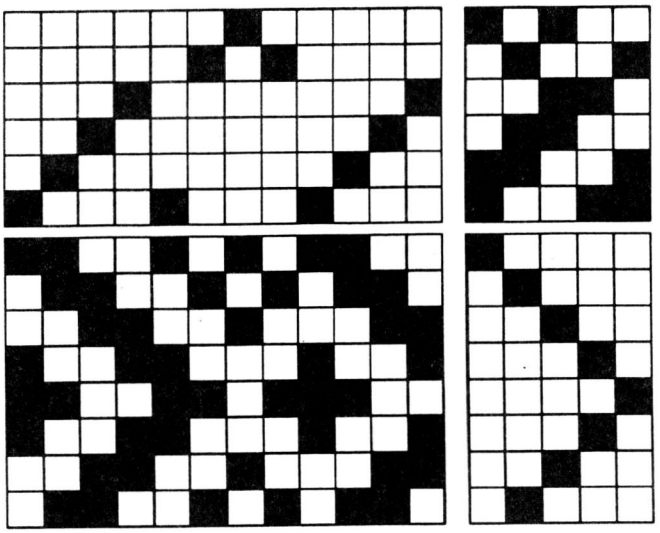

FIGURE 2

the *a posteriori* matrix factors computed. The graphical display is then updated to include their graphical display [Figure 2]. The main advantage to this approach is that data design updates are rapid, since no analysis-processing takes place at the time of design creation. This was of particular importance in previous implementations on small low speed microprocessors, where an emphasis was placed on the design environment [7]. However, a serious disadvantage is that, as the structural array is modified, there is no continuous feedback as to the structural implications of design modifications on the corresponding matrix factors. In this paper, algorithms for performing continuous dynamic factorization in response to incremental data modifications, are considered, along with the ensuing implications for the corresponding graphical display.

2. Dynamic Analysis Model.

The analysis process can be considered to consist of three distinct phases, corresponding to each of the three matrix factors. The first phase requires that all of the columns of the structure array be sorted into distinctness classes [3] . Each distinct column is allocated a separate row in the threading matrix (top left in Figure 2), and all identical columns of the structure array have the single non-zero element in the corresponding columns of the threading array in the same row. The second phase of the algorithm requires that the rows of the structure array be sorted in the same manner, and that each distinct row be allocated a separate column of the shed sequence matrix (bottom right in Figure 2). The third phase involves determining a mapping between the distinct columns and rows of the structure array. This information is recorded in the tie-up matrix (top right in Figure 2), and can normally be generated simply as the intersection of one representative of each distinct column with one representative of each distinct row, without further processing.

The expensive part of this algorithm is the bit-wise comparison of all the columns of the structure array. Although the rows also require a bit-wise comparison, having determined the distinct columns of the array, only one representative of each distinct column set need be considered and the number of elements in each row is greatly reduced. The sorting process can be represented graphically, very effectively, using a *digital trie* [7] for the columns and for the rows.

In constructing the array in Figure 1, initially all of the cells are coloured white, all of the columns are in a single node of the *threading trie*, and all of the rows are in a single node of the *shed sequence trie*. The single node, a leaf in this case, of the *threading trie* contains all of the column indices along with the binary sequence {00000000}, while the single node of the *shed sequence trie* contains all of the row indices along with the binary sequence {0}. As each change is made to the data structure, new branches are added, between the appropriate existing leaf and the new point of disagreement between two previously identical columns. The leaf column indices and binary sequences are also updated. After updating the *threading trie*, the *shed sequence trie* is modified. At each stage, the binary sequences contained in the leaves of the *shed sequence trie* correspond to the columns of the tie-up matrix. The digital tries corresponding to the structure array of Figure 1 are given in Figures 3 and 4.

It can be noted that the traditional binary tree graphical representation consumes a great deal of space, particularly as sequences become more complex. An alternative method of display is given by a *combed trie* representation, where all 'zero branches' are vertical and where 'one branches' appear to the right of any 'zero branches' at the same depth. Figures 5 and 6 show a *combed trie* representation of the structure array of Figure 1.

This data structure provides a means of rapidly updating the matrix factors corresponding to a given structure array in direct response to modifications to that array. It also supports

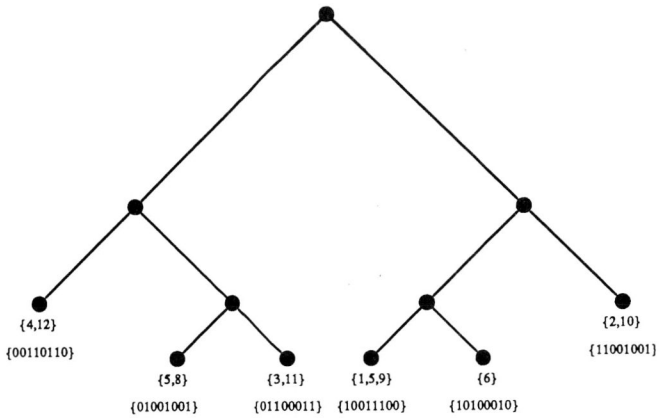

FIGURE 3

an excellent graphical representation of the analysis process which can, in fact, be monitored to determine where changes to the algorithm structure or implementation will result in increased efficiency [8,9].

3. Some Implications for the User Interface.

By using the digital trie data structure, the binary structure array and its corresponding three matrix factors can always be self-consistent. In addition, the rank of the factors can always be minimal and thus, in some sense, optimal. This pre-supposes, however, that the user will only update the structure array and will never wish to modify any of the three factors. This is definitely not the case. One of the tremendous utilities of such an interactive graphical system is the ability to modify any one of the four components and receive immediate visual feedback as to the implications of this action. This requires that a hierarchy of data elements be maintained by the software, with user-defined cells taking precedence over those set as a consequence of the analysis algorithm. This means that user-defined elements are not moved if, at some stage in the factorization process, their corresponding structure array

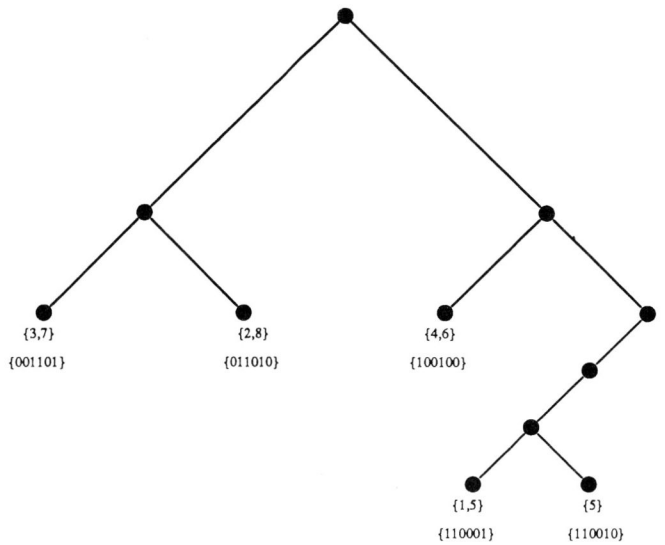

FIGURE 4

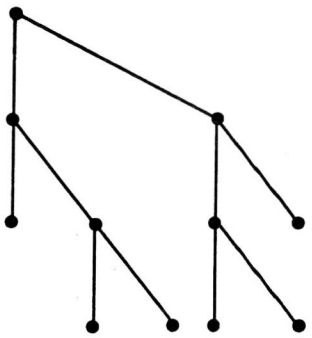

FIGURE 5

column or row becomes identical to another one. While this results in the factors not necessarily being of minimal rank, it does cause the user interface to conform to the design principle of predictability [10].

The consequences of this differential treatment of the user defined components of the partially determined factors, develops in complexity as the design process continues, and of course ultimately begins to limit the choices of the user. Contending with the accruing weight of these limitations places an increasing burden on the interactive response capabilities of the system and much further work needs to be done to resolve these problems.

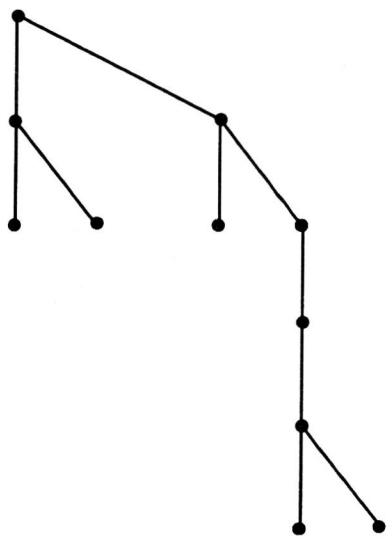

FIGURE 6

References.

1. J.A. Hoskins and W.D. Hoskins, The Solution of Certain Matrix Equations Arising from the Structural Analysis of Woven Fabrics, Ars Combinatoria 11 (1981), 51 - 59.
2. J.A. Hoskins and W.D. Hoskins, Using a Microcomputer for the Design and Analysis of Woven Textiles, 1983 ACM Conference on Personal and Small Computer, San Diego, California, 1983.
3. J.A. Hoskins and W.D. Hoskins, A Faster Algorithm for Factoring Binary Matrices, Ars Combinatoria 16B (1983), 341 - 350.
4. Janet Anne Hoskins, Isonemal Arrays and Textile Computer Graphics, Ph.D. thesis, 1985, University of Manitoba.
5. J.A. Hoskins and M.W. King, Interactive Design of Woven Textiles, Proceedings of the International Computer Color Graphics Conference, Tallahassee, Florida, 1983.
6. J.A. Hoskins and M.W. King, An Interactive Database for Woven Textile Design, Textile Institute Annual Conference "Computers in the World of Textiles", Hong Kong, 1984.
7. Alfred V. Aho, John E. Hopcroft and Jeffrey D. Ullman, Data Structures and Algorithms, (Reading, Mass.: Addison-Wesley, 1983)
8. Marc H. Brown, A System for Algorithm Animation, Proceedings of the SIGGRAPH '84 Conference, Minneapolis, 1984, 177-186.
9. Gretchen P. Brown, Christopher F. Herot and David A. Kramlich, Program Visualization: Graphical Support for Software Development, Computer 18 (8), 1985, 27-35.
10. J.D. Foley and A. VanDam, Fundamentals of Interactive Computer Graphics, (Philippines: Addison-Wesley Publishing Company, 1982).