Information Visualization (2005) 4, 32–48

www.palgrave-journals.com/ivs

Constructing and reconstructing the reorderable matrix

Harri Siirtola[1]
Erkki Mäkinen[1]

[1]*Department of Computer Sciences, Information Visualization Research Group at TAUCHI,*[†] *University of Tampere, Finland.*

Correspondence: Harri Siirtola, TAUCHI, Department of Computer Sciences, FIN-33014 University of Tampere, Finland.
Tel: +358 3 215 6066;
Fax: +358 3 215 6070;
E-mail: harri.siirtola@cs.uta.fi
[†]**http://www.cs.uta.fi/hci/ivrg/**

Received: 10 February 2004
Revised: 13 August 2004
Accepted: 28 September 2004
Online publication date: 3 February 2005

Abstract
We consider the backgrounds, applications, implementations, and user interfaces of the reorderable matrix originally introduced by Jacques Bertin. As a new tool for handling the matrix, we propose a new kind of interface for interactive cluster analysis. As the main tool to order the rows and columns, we use the well-known barycenter heuristic. Two user tests are performed to verify the usefulness of the automatic tools.
Information Visualization (2005) **4,** 32–48. doi:10.1057/palgrave.ivs.9500086

Keywords: Multi-dimensional data visualization; the reorderable matrix; cluster analysis; barycenter heuristic; user interface

Introduction

Perhaps the most common method to represent a data collection is to construct a table out of it. In everyday life, there are many examples of this: transportation schedules, television and radio programs, or even results from a beer tasting. In scientific writing, the tables are even more common – could you imagine a description of an experiment without a single table?

Generally, a *table* is defined to be *a set of data arranged in rows and columns*. Within this definition, we have to decide the actual layout: what is the interpretation for columns and rows. Usually, we tend to have objects in the columns and their characteristics in the rows, or *vice versa*. This kind of layout is also referred as to *cases by variables*.

Webster's Revised Unabridged Dictionary gives several definitions for the word 'table'. One of them is:

> Table: A condensed statement which may be comprehended by the eye in a single view; a methodical or systematic synopsis; the presentation of many items or particulars in one group; a scheme; a schedule.

Often a table fails to be condensed enough to be 'comprehended by the eye in a single view'. The human eye is very capable in detecting and processing graphical patterns – why do not we transcribe the table to a graphic? This is what the French cartographer Jacques Bertin proposed.

Bertin's reorderable matrix[1–3] is a simple visualization method for exploring tabular data. The basic idea is to transform a multidimensional data set into a two-dimensional (2D) interactive graphic. The graphical presentation of a data set closely resembles the underlying data table in that it contains rows and columns. These rows and columns can be permuted, allowing different views of the data set. The actual data values are replaced with symbols, say circles or rectangles, which have a size relative to the actual data value. The smallest value is represented as a 0-sized symbol and the largest value as a symbol filling the whole area available. While interacting with the visual presentation, the user has a

chance to detect patterns in the presentation and to gain insight into the data. This kind of pattern recognition is something that human vision is known to do remarkably well.

The initial transformation from a data table into a reorderable matrix is called 'construction' by Bertin, while the seeking of interesting patterns by row and column permutations is termed 'reconstructing' the matrix; hence the name of this article.

We advocate the use of the reorderable matrix in application areas that call for human expertise to guide an automatic process. These areas include architecture, medical diagnosis, and various other disciplines where the knowledge is difficult to transform into a form 'understandable' for computers. When using the reorderable matrix, the manual and automatic phases for reordering the matrix take turns and support each other. Therefore, the human knowledge guides the process, and yet the dull parts are left to computer.

Figure 1 contains a small reorderable matrix (Bertin, p. 32)[1] that has binary values only. In this simple example, we have 16 townships *A*,*B*,*C*,…,*P*, for which we know the presence or absence of nine characteristics. The question is whether the same planning decisions should be applied to all of these townships?

Processing the reorderable matrix in Figure 1 involves bringing together similar rows and columns. This in turn involves dragging rows and columns, one by one, and can take a while. We leave the intermediate steps out and represent the result in Figure 2.

From the arrangement in Figure 2, it is quite easy to see that there are obvious groups in townships. Townships *H*,*K* have similar characteristics and could be labeled column-wise as CITIES and row-wise as URBAN. Similarly, townships *A*,*I*,*E*,*M*,*F*,*P*,*J*, and *N* could be classified column-wise as VILLAGES and row-wise as RURAL. The remaining set could be called TOWNS, as an intermediate level between CITIES and VILLAGES.

The reorderable matrix can assist in a knowledge acquisition task in a number of ways. The general principle is to sort the matrix either horizontally or vertically, according to a column or a row, respectively. The chosen column or row should be one that seems to portray a phenomenon that has general influence. After threading, the matrix should be arranged so that similar rows appear together, forming black areas in the matrix. This kind of arrangement is ready to be analyzed.

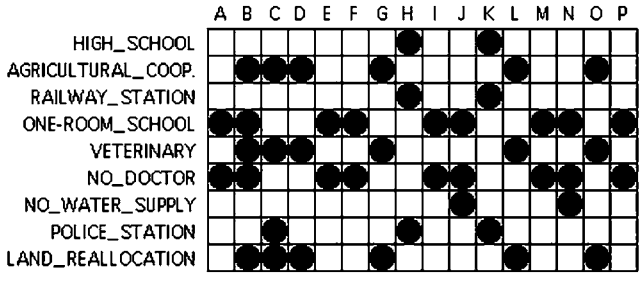

Figure 1 The townships planning decisions example.

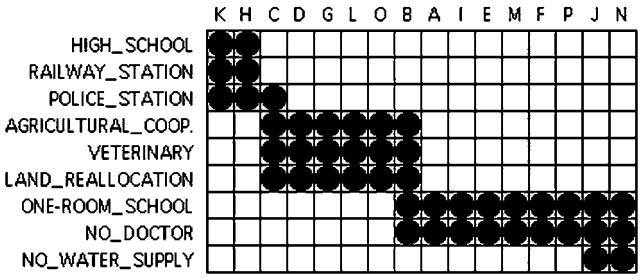

Figure 2 An arrangement that reveals the correct decisions.

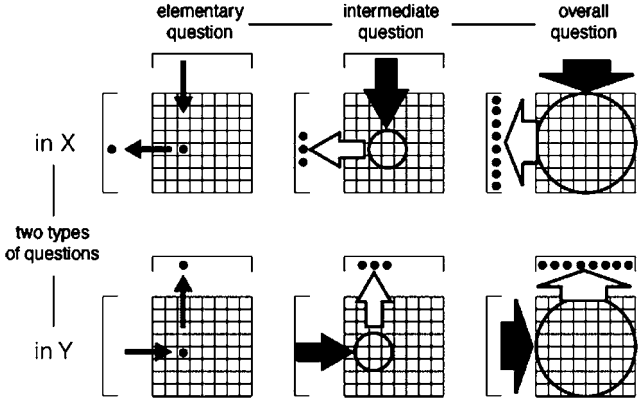

Figure 3 Three levels of information: elementary, intermediate, and overall (Bertin, p. 13).[1]

There are three kinds of questions that can be answered with an appropriately arranged matrix: questions about certain objects or characteristics, questions about subsets of objects or characteristics, and questions about overall object or characteristic sets. Figure 3 illustrates these three levels of information.

A typical elementary-level question from the townships example would be 'Does the township A have a police station?' This kind of elementary question can, of course, be answered whether or not the matrix is arranged. An example of an intermediate-level question would be 'What are the townships that have one-room schools and no doctor?' And finally, an overall question could be 'How many townships have their own veterinary and doctor?'

The reorderable matrix is an early interactive information visualization tool, proposed a long time before information visualization was recognized as a field of its own. The reorderable matrix takes advantage of one of the fundamental techniques in information visualization: the rearrangement of data (see Spence pp. 14–32)[4] for discussion). Today, it is difficult to find a visualization tool that does not capitalize this technique in one form or another.

The reorderable matrix is a general tool independent of any application area. Somewhat similar tools are developed in certain specific fields, such as ROP in architecture,[5,6] and The design structure matrix (DSM) in system analysis and project management.[7] Alike the reorderable matrix, also ROP and DSM are based on the re-arrangements of data given in a form of matrices.

It is typical for all applications that they call for human expertise in the ordering of the rows and columns. An interactive tool best helps the designer by allowing manual and automatic phases of the ordering process to take turns and support each other.

The purpose of this paper is to survey the known material concerning the reorderable matrix, and to establish links between the reorderable matrix and some other closely related formalisms for handling tabular information. Moreover, we discuss techniques for efficiently automating the process of ordering the rows and columns of the matrix, and propose a method to produce interesting arrangements automatically. The usefulness of automatic ordering is verified by two user tests.

Here, we will briefly describe the organization of this paper. We introduce some new applications and discuss the relationship between the reorderable matrix and some closely related formalisms. In the section 'Ordering the rows and columns', we discuss different implementations of the automatic reordering of rows and columns in the reorderable matrix. We introduce the column similarity view as a new tool for handling the reorderable matrix. It was found that the view indeed helps the user to order and understand the information presented in the matrix. Also of our interest is the barycenter heuristic used so far mainly in graph drawing algorithms. The section 'The user interface' is devoted to studies of various alternatives for the user interface of a computer implementation of the reorderable matrix. In the section 'Empirical studies', we report two user tests where we verify the usefulness of the proposed new interaction methods. Algorithmic problems related to the reorderable matrix are studied elsewhere by the authors.[8]

Ordering the rows and columns

Consider again the townships example given in the Introduction. Even for this simple example, there is a large number of *arrangements* or possible row and column permutations to be explored. For the townships example, the number of possible arrangements is

$$(\#\ of\ rows)!\times(\#\ of\ columns)!>7.5\times10^{18}.$$

Many of the arrangements are isomorphic as patterns – for example, mirroring the matrix vertically or horizontally does not change the patterns, although the arrangements are not the same. Still, even the number of different patterns is high. In this example, the size of the matrix was small, 16×9, although Bertin suggested that the reorderable matrix should be usable with sizes up to $x\times y=10{,}000$. The size of the largest experiment he wrote about was $415\times76=31{,}850$. It is obvious that finding the interesting patterns from such matrices calls for automation.

There are some known implementations for automatic reordering of the reorderable matrix, most notably those published by researchers of the Eindhoven University of Technology (see Daru and Adams,[9] Snijder[10] and Veenendaal[6]) as well as those used in the connection with engineering data management. Experimental implementations of the reorderable matrix are reported by Lohninger,[11] Rao and Card,[12] and Schmid and Hinterberger.[13] All these are too complicated to be useful in an interactive tool. Thus, we need much simpler and faster methods to reorder the rows and columns.

We present a simple and efficient heuristic for reordering the reorderable matrix. The algorithm has been implemented and tested with a number of matrices, and we have experimented to find an efficient user interface for it.

The problem of drawing bipartite graphs with as few edge crossings as possible is a much-studied subproblem of graph drawing. It is assumed that the vertices are drawn in horizontal lines such that separate vertex sets are placed in different horizontal lines and edges are drawn as straight lines between the sets. The drawing problem reduces to the problem of ordering the vertices in the lines such that the number of edge crossings is minimized.

There are actually two separate problems: we can fix the order in one of the horizontal lines and ask for the optimal order of vertices in the other line, or we can order the vertices in both vertex sets of the bipartite graph in question, that is, order the vertices in both of the horizontal lines. Even the former problem is known to be NP-complete.[14] Next, we will discuss the latter, somewhat harder problem. To avoid the possibility of confusion, we call it simply the *drawing problem*.

In the barycenter heuristic, we order the vertices according to the averages of their adjacent vertices in the opposite vertex set. By repeating this ordering process in turns in the two vertex sets, we (hopefully) reach orderings of vertices that minimize the number of edge crossings. For further information concerning the drawing problem, Di Battista *et al.*[15]

Figure 4 shows how the adjacency matrix of a bipartite graph is changed when applying the barycenter heuristic. It is evident that when applying the heuristic, the corresponding adjacency matrices have the tendency to be reordered so that there are 'black areas' in the top left and bottom right corners. This is just what we wanted to establish when ordering the rows and columns of a reorderable matrix. Hence, the barycenter heuristic seems to suit the task of ordering the reorderable matrix very well. Consider, for instance, the case where we thread the matrix with respect to a certain row. We can drag the chosen row to be the topmost row, and then permute the columns so that all the non-null entries in the topmost row are in the beginning of the row. Now the pattern with black areas in the upper left and lower right corners

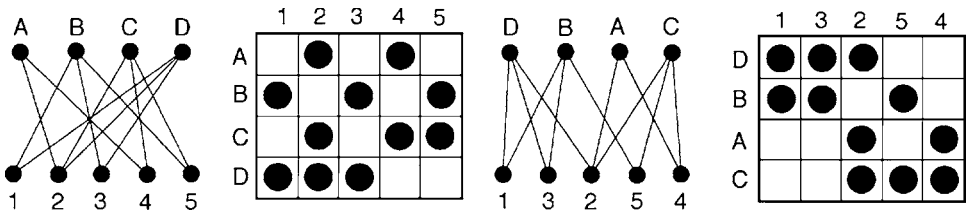

Figure 4 Applying the barycenter heuristic to a simple bipartite graph.

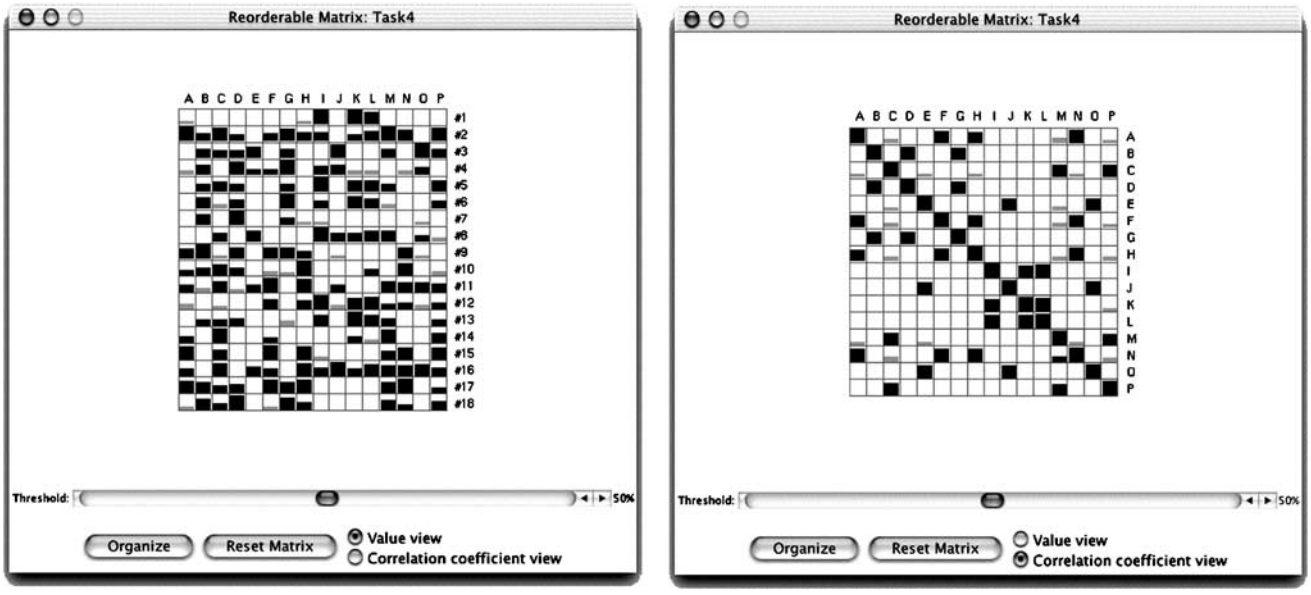

Figure 5 The user interface: slider for adjusting the threshold and the views for actual values and column correlation coefficients.

is an obvious goal that is reached using the barycenter heuristic.

In our previous study,[8] we proposed two methods for creating interesting arrangements automatically for a reorderable matrix, namely, the 2D Sort method and the barycenter heuristic. The 2D Sort method is simply a two-dimensional iterative sort that tries to build a black area into the upper left and lower right corner of the matrix. Earlier,[16] we have found the evidence that these two patterns are important goal states for users when they are 'interpreting' a reorderable matrix.

In our previous study,[8] we have conducted experiments with both artificially generated test matrices and real-life data sets. As a result, we now believe that both the 2D sort and the barycenter heuristic only perform well with a relatively small subset of matrices.

Our new method is based on the barycenter heuristic, but includes two important improvements. The former is related to fixing the parameters for the barycenter heuristic, and the latter is a new way to look at data.

In our previous implementation,[8] we used a fixed threshold value for what cell values are considered 'black'. In our new implementation, the threshold value can be changed with a slider provided in the user interface. Thus, the user has full control over which cells are considered black and as such, also over the candidates to be moved towards the diagonal.

The second improvement is to provide an alternative view to our matrix, and treat it also as a reorderable matrix. Both a column correlation view and a Euclidian column distance view have been experimented with. Changing the column order in one view is reflected in the other view as well. The user interface with a correlation coefficient view is displayed in Figure 5.

The reason for introducing another view to the data is two-fold. It is a common task with a reorderable matrix to look for *similarities* between cases or variables, or to perform an interactive cluster analysis. If we can visualize the similarity information, we can reduce the cognitive load and make the task of interpreting the data easier to perform. We have several possibilities for the similarity metric to be used, and we have considered the correlation coefficient and the Euclidian distance metric in this article. Another idea here is to represent the similarity information as a reorderable matrix. When we initially construct the reorderable matrix for a data table, it is trivial to construct a reorderable matrix of distance metrics at the same time.

The second reason is that the distance matrix is the kind of matrix that is easy to process with the barycenter heuristic – it is rectangular, has always 'black' or 'white' values on the diagonal, and when it contains obvious groups, the groups can be arranged as rectangular areas around the diagonal. The matrix in Figures 5 and 6 contain strong dependencies and is easy to arrange automatically. A data set with less obvious dependencies would require experimenting with slider settings in order to find the appropriate threshold value.

It is well known in information visualization that dynamic graphics generally enable us to see things that static graphics do not (Ware, pp. 223–225).[17] Our first slider implementation for the threshold value of the barycenter heuristic did not update the view while the slider was moved. It turned out to be useful to implement the continuous update because observing the data items switching between black and grey often reveals the most useful threshold, and makes it easier to observe the structure of the data. This idea can be taken even further, as we show in the second experiment.

An obvious concern in a matrix reordering process is its motivation. If the problem is just to bring the similar rows and columns together, why do not we just compute the solution and be done? Dynamic and rapid exploration of possible clustering makes it possible to make observations that would be difficult to see from a static representation. Even statisticians admit (Venables and Ripley, p. 316)[18] that the visualization methods are often far more effective than many of the proposed clustering methods that usually give conflicting answers and expose to the danger of over-interpretation.

The user interface

Bertin's reorderable matrix is superficially a straightforward interface to tabular data. However, many issues have to be accounted for and resolved when constructing a proper user interface for it. This section presents our analysis of the user interface and its implementation.

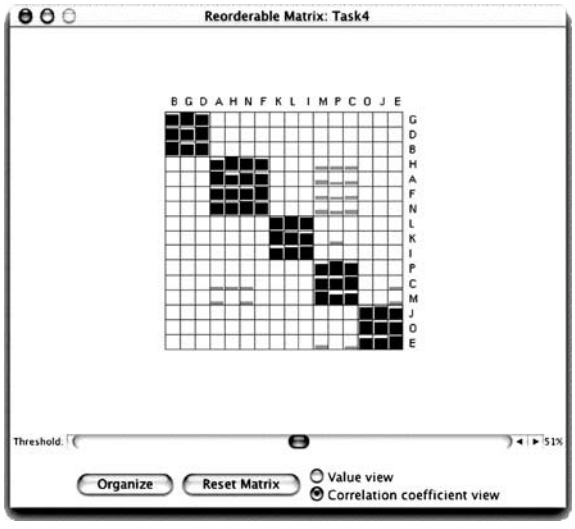

Figure 6 The column correlation coefficient matrix from Figure 5 arranged with the barycenter heuristic.

Operations

There are four classes of operations that a user can perform to permute a reorderable matrix: move, thread, block, and arrange operations.

Move operations. Moving a row or a column to a new position in an attempt to form a more interesting pattern, or to compare two distant rows or columns to each other.

Thread operations. Sorting the whole matrix according to a row or a column. This is a kind of structured operation for multiple move operations, or for arranging the matrix according to a selected characteristic.

Block operations. When an interesting pattern is detected the area can be 'locked down', preventing an accidental change. The locked-down area can be moved, but only as whole. When a matrix with a blocked area is threaded, the area behaves as a single row or column. The value for this area is calculated simply by averaging the values according to appropriate direction.

Arrange operations. The fourth class of operations is the one where we try to seek potentially interesting arrangements by automation. Obviously, this class of operations would be very difficult to implement in a manual construction.

The third class is error-prone as a manual operation, and the fourth one is feasible in a computer program only. These four classes are not mutually exclusive. Both arrange operations and threading operations can be seen as automated collections of move operations.

Appearance

One important aspect in the user interface for the reorderable matrix is the visual appearance of the matrix. The matrix is of the type 'cases by variables', or *objects* in columns and *characteristics* in rows, as Bertin prefers to call them. Besides the layout, there are alternatives for representing the entries of the matrix.

A single data item can be represented in a number of ways. Bertin (pp. 41, 44)[1] turns data items into graphics by filling a rectangle with an appropriate amount of black color or by sizing a centered black shape appropriately, as can be seen in Figure 7. The shape can be a circle or a rectangle, or a combination, where maximum values are indicated with a rectangle and other values with a circle shape.

Bertin seems to use these two presentations interchangeably, and does not discuss their relative merits or

Figure 7 Two alternatives to present the row values of a reorderable matrix.

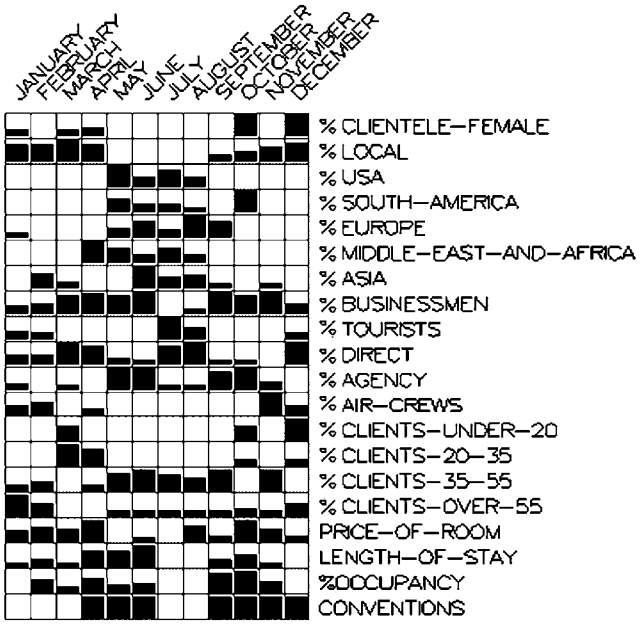

Figure 8 Placement of label elements.

disadvantages. The space-filling approach resembles histograms and could be more familiar to new users.

The visual appearance of a reorderable matrix can be *row-oriented*, *column-oriented*, or *neutral*. When the presentation is row-oriented, we use stronger visual grouping between the line elements than between the lines. With this approach, the matrix appears more like a set of stacked histograms. This representation has been selected in some of the more recent implementations (e.g. *Voyager*[19] and *Table Lens*[12]). The traditional presentation is neutral and does not emphasize cases or variables.

Another issue in the appearance is the treatment of row and column labels. With row labels, the natural thing to do is to place them next to rows, either on the left or on right side of the matrix. However, with the column labels, we have a problem – short labels can be placed above (or below) of the columns. Unfortunately, this kind of space does not really allow anything but one or two character labels. Bertin's solution is to place column labels into orthogonal angle. In Figure 8, we have placed the column labels in a 45° angle.

Besides the row and column labels, it is also common to add annotations to the diagram. In Figure 8, there may be sets of characteristics that have something in common and the column groupings may indicate classes of objects we have found.

Interaction

The reorderable matrix is one of those user interfaces where the direct manipulation approach is the natural one. The first devices that Bertin constructed to permute matrices were based on paper strips, photocopying machines, and objects resembling domino pieces (Bertin, p. 35).[1] With these devices, the direct manipulation is of course evident, and there is no reason to divert from this in a computer implementation. The user interface of our implementation can be seen in Figure 9.

The visibility of possibilities, or *affordances*, is an important feature in a user interface that is based on direct manipulation. In the user interface of the reorderable matrix, this principle means that the possibility of move, sort, and arrangement operations should be visible and obvious to the user.

Empirical studies

We conducted two experimental studies to see how the proposed method compares to the traditional permutation interface, and to the solving of tasks by using a static image of the matrix. The test task was to find clusters from a matrix. This section presents the details and results of the experiments.

The performance of the automated matrix reconstruction process can be split into three factors. The two most easily measured issues are the time it takes to perform a task and the correctness of the result. The third issue is the subjective satisfaction, which can be estimated with questionnaires and interviews.

Task for the experiment

The task in the experiment was to recognize groups of similar cases in a given reorderable matrix. Here, similarity means simply that the cases have roughly the same values in most of their characteristics, or more formally, that the Euclidian distance is small. This is the general basis for grouping objects together, or for classifying objects as belonging to the same group.

We constructed two sets of matrices where the objects were named by letters $A,B,C, \ldots$, and the rows were numbered from 1 to m. Cases varied from strong dependencies between groups of columns to no dependencies at all. The sets were constructed so that the corresponding matrices had the same dimensions and the same number of subgroups, although not the same ones.

In Figure 10, we have a simple test task where the correct answer would be $\{A,G,D,I\},\{C,B\}$, and $\{F,E,H\}$. The initial ordering is shown on the left and one of the answer-revealing orderings in the middle.

The correct grouping for each matrix was explored by creating a *heatmap* (a function in the package 'classical multivariate analysis', or *mva*, in the statistical system R[20]), which has row and column *dendrograms* displaying the similarities. Figure 10 shows the heatmap of the matrix on the right. A dendrogram is a representation where the dissimilarity between two cases can be read from the height of the path by which they are joined. In Figure 10, the dendrogram has been computed by using Euclidian distance metrics and by treating the dissimilarity between groups as the maximum of the dissimilarities between group members. The correct answer is shown by the column dendrograms.

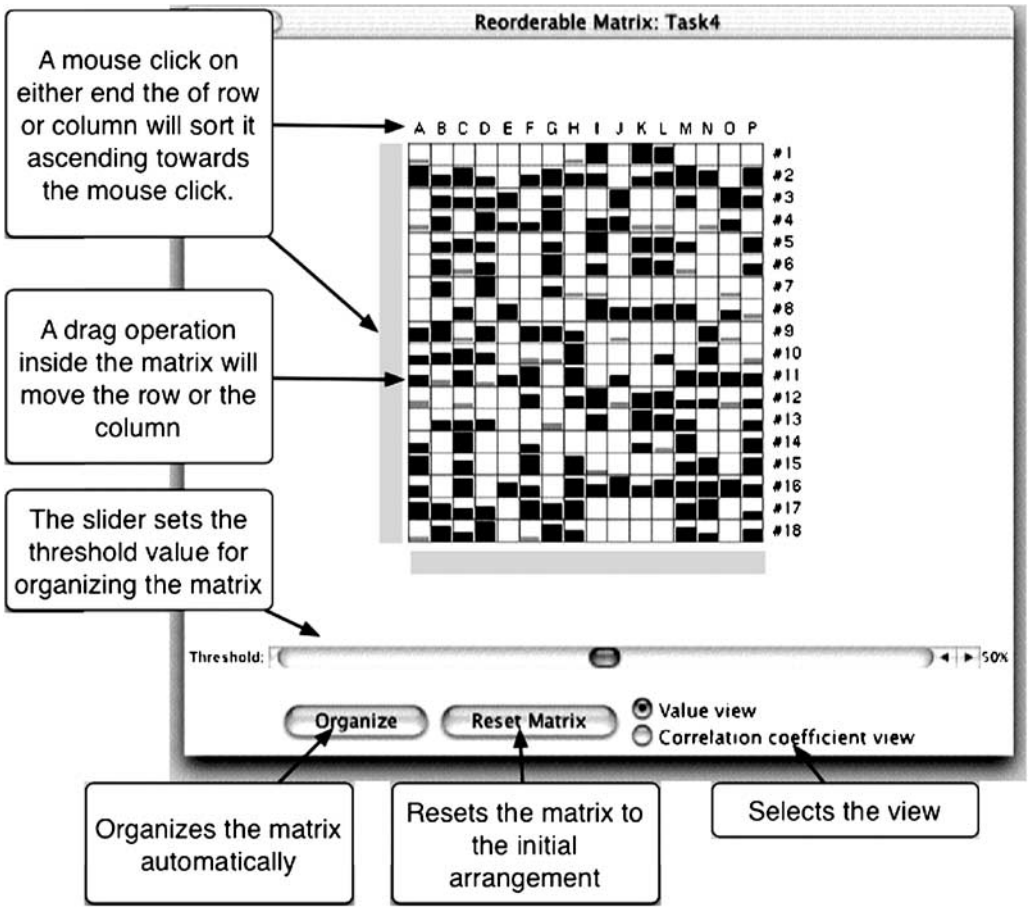

Figure 9 The user interface for reorderable matrix.

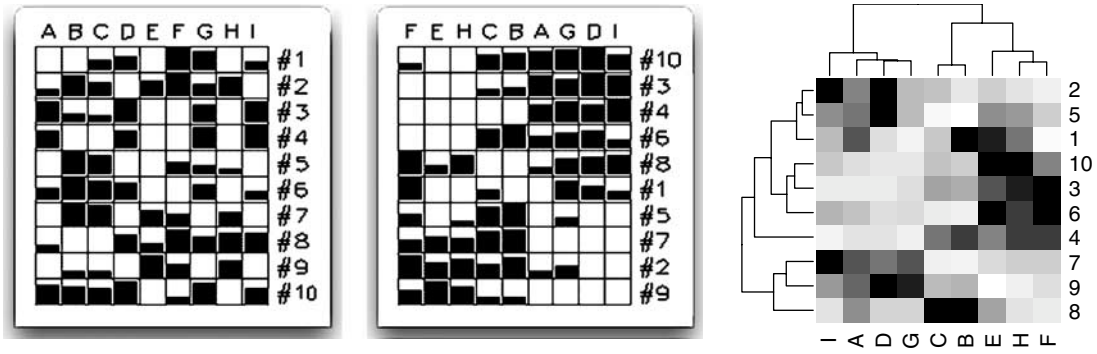

Figure 10 The initial setting on the left, one of the solution-revealing settings in the middle, and the heatmap of the matrix on the right.

The relation between a static dendrogram and our method is that we use a heuristic to approximate rapidly the groups that are achieved by cutting a dendrogram at certain height. This raises the question of why not to use a pre-computed dendrogram and compute the groups by slicing it? In case of Euclidian metrics that approach would be possible, but we anticipate the use of more general metrics and the possibility to add of more interaction parameters to the method.

It is obvious that the solution based on cluster analysis is far more fine-grained than what we perceive as natural grouping, meaning that we may over-interpret the found differences. The 'correct' solution for each task was determined by both looking at the analytical solution and by checking the consensus with the empirical results. This may lead to some errors, but blind trust to the analytical solution would probably cause even larger errors.

Measuring the accuracy of the result

The results of the arrangements made by the test persons are evaluated by using a simple distance measure. Consider a set of elements S. The partitions of S form a lattice structure, in which the partition containing all the elements of S is the universal upper bound, and the partition with singleton sets is the universal lower bound. The lattice operation is the shift of an element of S from a set to another. Figure 11 shows the lattice structures in the cases $S_2 = \{a,b\}$, $S_3 = \{a,b,c\}$, and $S_4 = \{a,b,c,d\}$.

Suppose we have the correct rearrangement c and an arrangement t produced by a test person with the partitions P_c and P_t, respectively. The accuracy of t is defined to be the shortest distance from P_c to P_t in the lattice. The bigger the distance, the worse the accuracy. Intuitively, the rearrangement t is evaluated by counting the number of shifts of single elements needed to obtain the correct partition from the one corresponding to t.

The number of partitions of an n element set is known as the Bell's number, b_n.[21] The number count the ways that n distinguishable objects can be grouped into sets if no set can be empty. The Bell's numbers grow exponentially and they satisfy the recurrence

$$b_{n+1} = \sum_{k=0}^{n} b_k \binom{n}{k},$$

where $b_0 = 1$. From Figure 11 we see that, for example, $b_3 = 5$ and $b_4 = 15$. The next Bell's numbers are 52, 203, 877, and 4140. Hence, in some of our test cases, the lattice is already of a remarkable size. The closed formula for b_n is

$$b_n = \sum_{k=1}^{n} \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n.$$

The most practical method to compute small Bell's numbers is to use the *Bell's Triangle*.[21]

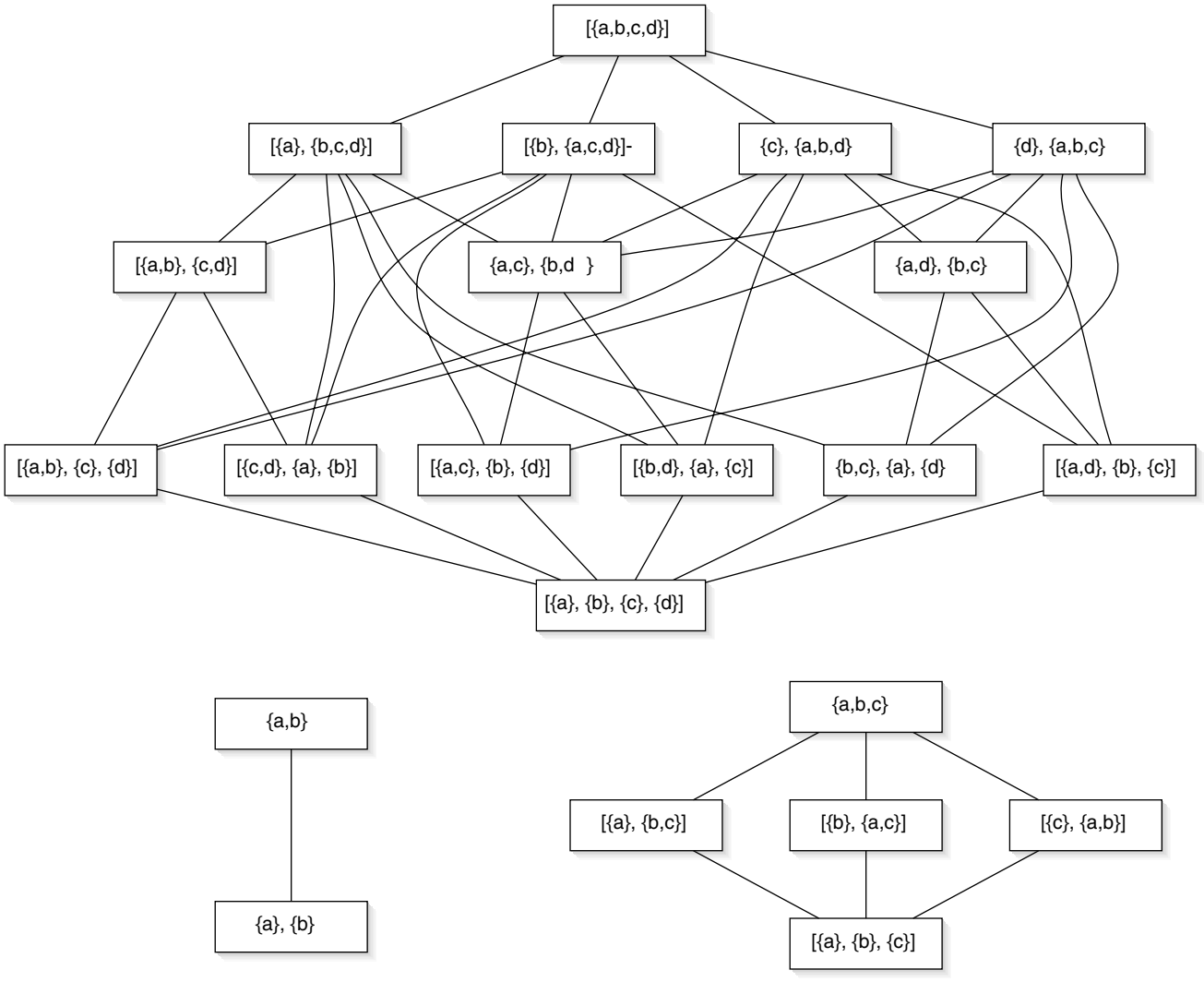

Figure 11 The lattice structure for the three cases $S_2 = \{a,b\}$, $S_3 = \{a,b,c\}$, and $S_4 = \{a,b,c,d\}$.

Guessing the result When estimating the overall accuracy of the results obtained by the new tool, it is instructive to compare them against those obtained by guessing.

As an example, consider the case of partition {*A*,*F*,*E*}, {*B*,*H*}, {*D*,*I*,*G*}, {*C*} , where we have only nine elements. Assuming that all partitions are equally probable, the possibility that this particular partition appears is 0.000047. On the other hand, the probability increases when the allowed distance from the correct result increases. The probability of guessing a partition by a distance of at most 1 from {*A*,*F*,*E*}, {*B*,*H*}, {*D*,*I*,*G*}, {*C*} is ≈ 0.0016.

As an additional example, consider the partition {*B*,*E*},{*D*}, {*M*,*I*,*A*,*F*}, {*L*,*C*}, {*J*,*H*,*G*}, {*N*,*K*}. Since each of the elements can be moved from its current set to any other set or a new set can be formed, there are 80 partitions at a distance of 1 from the correct answer. Hence, the probability to obtain a result at a distance at most 1 from the correct results (the average result of the user tests) is $81/b_{14} \approx 0.00000107$. The difficulty of guessing grows rapidly when the total number of elements increases. These probability calculations clearly stress the difficulty of the tasks.

First experiment

The first experiment was carried out with the user interface in Figure 5. The similarity view uses column correlation coefficient as a metric, and the black value indicates a high similarity. Only one of the views is visible at a time, and the interaction involves setting the threshold and then reordering by pressing the *Organize* button. Moving the threshold slider updates the black/grey condition continuously and provides a method for observing structures.

The first experiment was set to find the average differences of task execution time and solution correctness between the following three conditions: performing the tasks with pencil and paper (no reordering), performing the tasks with a traditional user interface for the reorderable matrix (manual reordering and sorting), and performing the tasks with a new interface (automated reordering). Besides offering a baseline comparison, the paper and pencil condition will show the effect provided by the reordering of rows and columns.

Participants A total of 12 subjects participated in the experiment for a chance to win a movie ticket. All subjects were either students or employees of our research unit, Finnish speaking, and had considerable experience in the use of computers, but no prior experience in specialized visualization tools. The medians of 'computer use' and 'daily computer use' were 15 and 7 years, respectively.

Seven of the subjects were men and five were women, and the age of the subjects varied from 22 to 35 years. The median age was 26 years.

Apparatus The experiment was performed with a 500 MHz Apple PowerBook laptop computer having a 1024×768 pixel display and a Microsoft Intellimouse Optical mouse. The interaction in the experiment required only the use of mouse as a pointer and the left mouse button as a selector.

The test program was implemented in Java by using the GL4Java library for OpenGL graphics. This design allowed faster screen updates than the standard Java graphics API.

Task materials Two isomorphic sets of 10 matrices were generated with a program. The generation process was parameterized so that the resulting matrices contained varying types of clusters. The median number of clusters in a matrix was 4.5 and the median size of a cluster was 3. The matrix size ranged from 49 to 575 cells and the median size was 177 cells. The matrices can be seen in Figure 12.

Procedure The cases by variables data structure and the idea of a reorderable matrix were explained with examples. The cluster finding task was described and it was stressed that the test matrices might have any number of clusters, ranging from one up to total number of cases.

The. instructions were to find the clusters from a matrix 'as quickly as possible while trying not to make mistakes'. If the participant asked the priority of these conflicting goals, the answer was to strive for correctness. Participants were instructed to give the answers by writing them down on a sheet of paper.

The procedure was repeated 10 times using paper and a pencil, and then 10 times using one of the computer applications. A demographics questionnaire was completed prior to starting the test tasks and a feedback questionnaire was completed afterwards. In the end, the participants were interviewed for comments and had a chance to ask questions about the experiment.

The experiment was conducted in an office room with one supervisor. The subjects were informed that the test contains 10 matrices. The matrices were shown one by one and there was no time limit for solving a task. The scheduled time to conduct a single experiment was 45 min.

Design We have three conditions:

(1) *Paper and pencil*. Subjects receive a paper with a picture of the reorderable matrix in a random arrangement. Subjects have the freedom to use any kind of technique they see appropriate as long as they use only paper and a pencil.
(2) *Computer program with manual operations*. Subjects use a computer program implementation of the reorderable matrix. The operations available are the manual ordering operations: move a row, move a column, sort according to a row, and sort according to a column.

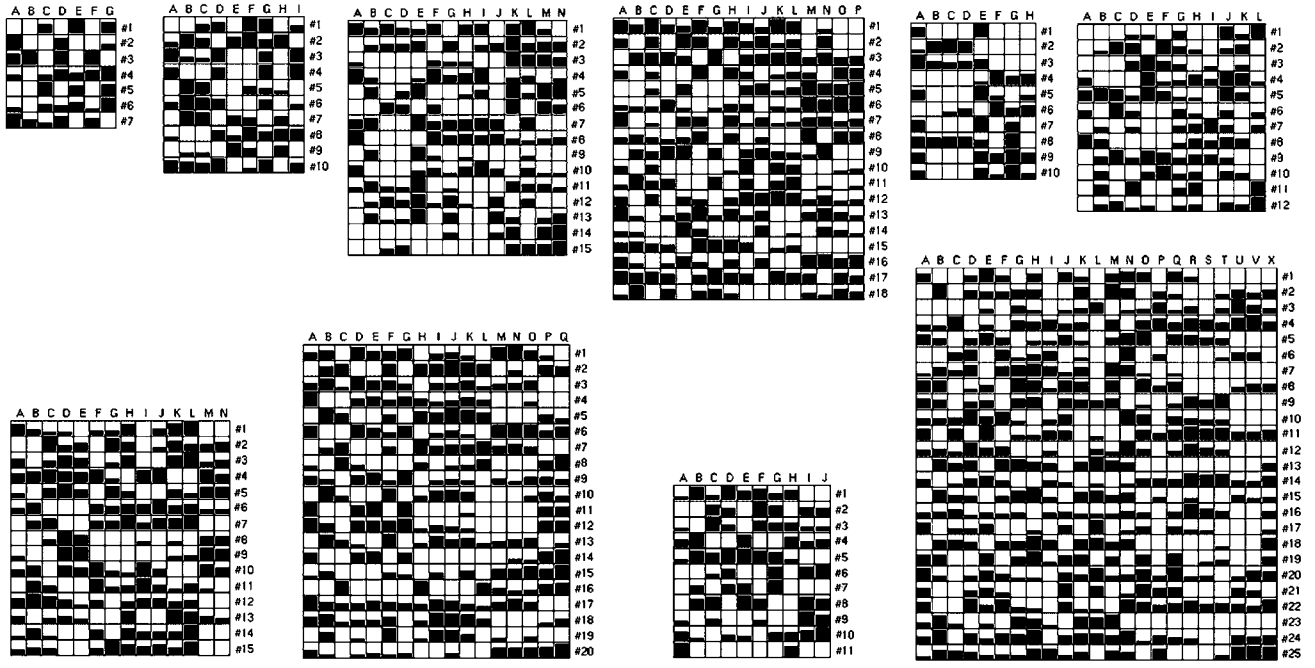

Figure 12 The collection of test matrices for paper and pencil task in the first experiment. Matrices 1 to 6 appear on the upper row and matrices 7 to 10 on the lower row.

(3) *Computer program with automatic ordering*. Subjects use a computer program implementation of the reorderable matrix. In addition to the manual ordering methods, they have an automatic method based on the barycenter heuristic.

In what follows, the three conditions are referred to as *Paper*, *Traditional*, and *Organize*, respectively.

The participants were divided randomly into two groups. The groups were presented with condition pairs (1,2) and (1,3). Both groups took the *Paper* condition first, then either *Traditional* or *Organize*.

The experiment was treated as $3 \times 2 \times 6$ balanced incomplete design – incomplete, since each participant saw only two of the conditions. *Group* was a between-subjects factor with two levels (six participants per group), and *UI* was both between-subjects and within-subjects factor with three levels *Paper*, *Traditional*, and *Organize*. The total number of inputs was six participants/group $\times$ two groups $\times$ two out of three conditions $\times$ 10 trials $=$ 240 observations.

Two measurements were collected for each trial: time spent (in s) and the given solution. The solutions were later transformed by hand into distances computed with the appropriate lattice.

Results The median time for completing the experiment was 55 min, ranging from 30 min to 1 h and 30 min. This was close to the designed 45-min session time.

The task execution times can be seen in Table 1. The grand mean for task execution was 171 s, or 2 min and 51 s. The manual solution with paper and pencil took 205 s, on the average, and provided the baseline for comparisons. The traditional user interface for the reorderable matrix was 24% faster in this experiment than the baseline. Finally, the proposed new interface offered a 41% improvement over the manual alternative, and a 23% improvement over the traditional interface.

Table 1 The mean task execution time and the distance from the correct solution for each condition

Condition	*Time*	*Distance*
Paper	205	2.4
Traditional	156	1.8
Organize	120	1.0
Overall	171	1.9

Since the F test indicates a statistically significant difference among the execution times of the three user interfaces ($F_{2,11} = 19.5$, $P < 0.001$), we can proceed to pairwise comparisons. The adjusted comparisons (Tukey contrasts) indicate that all the pairwise differences for *Time* are statistically significant (*Organize–Paper*: $t = -6.10$, $P < 0.001$, *Traditional–Paper:* $t = -3.30$, $P = 0.002$, *Traditional–Organize*: $t = -2.43$, $P = 0.015$). Figure 13 illustrates the situation.

Similar analysis for the distance from the correct answer, *Distance*, reveals that only the *Paper–Organize* difference is statistically significant ($t = -4.02$, $P < 0.001$),

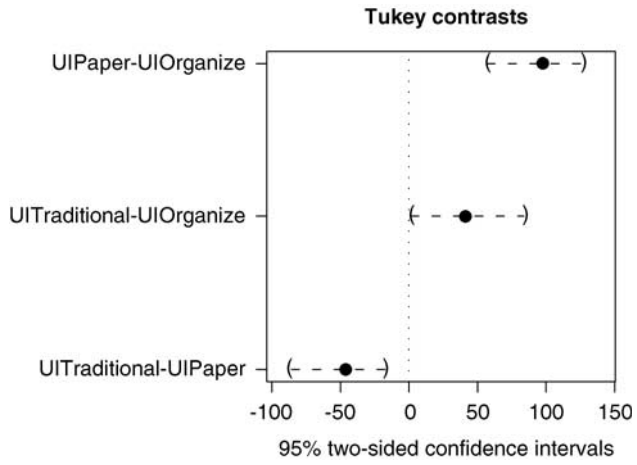

Figure 13 The 95% simultaneous confidence intervals for pairwise comparisons of *Time* with asymptotic adjustment.

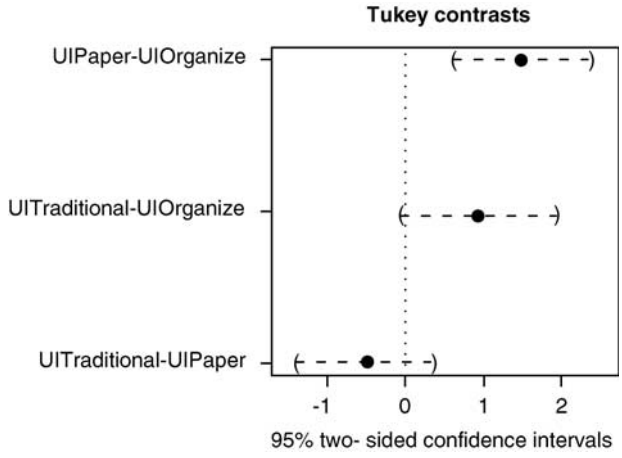

Figure 14 The 95% simultaneous confidence intervals for pairwise comparisons of *Distance* with asymptotic adjustment.

and that the *Traditional–Organize* difference is approaching significance ($t = -2.20$, $P = 0.05$). Improvement in the first situation is about 58%. The overall situation is shown in Figure 14.

The results of the first experiment can be visually summarized by creating a 2D kernel density plot for the estimated distributions of *Time* and *Distance* (Figure 15).

Subjective satisfaction

The subjective user satisfaction was studied with a post-test questionnaire. The following discussion is about the 20 propositions in the questionnaire. In Figure 16 and in the following discussion all the propositions have been rewritten to have their favoring meaning as 'agreeing' to make the interpretation easier.

The propositions

- *It is easier to find similar columns with the matrix application than with a pencil and paper (4).*
- The correlation view makes it easier to find similar objects (10).
- It is more useful to show matrix cells as graphics than numbers (15).
- The threshold slider makes is easier to detect patterns in a matrix (3) and
- The matrix application gives a good overview of the data matrix (9)

got the most positive evaluation – none of the participants disagreed with these statements.

One of the participants disagreed with the proposition

- *The correlation view makes it considerably easier to find similar columns (17)*

but generally this was accepted – eight out of 12 participants agreed strongly with the statement.

None of the participants disagreed with the following statements:

- *The threshold slider makes it more interesting to study a matrix (5).*
- *Both views have an important role in the user interface (19).*
- *Two views to the same data matrix does not make the user interface confusing (8).*

This indicates that the two-view approach was accepted.

Two out of 12 participants disagreed and one was undecided with the following statements:

- *I could use this kind of application in my everyday work (2).*
- *The feedback from matrix application is fast enough (16).*

The response to proposition (2) is especially encouraging for the prototype-level application.

The following statements received a weak positive response:

- *The reorderable matrix application is easy to use (1).*
- *It is easy to learn the basic idea of the reorderable matrix (7).*
- *The ordering of rows and columns could not be easier (12).*
- *It is easy to drag the rows and columns manually (6).*

The propositions that received the most undecided answers are:

- *The connection between value view and correlation view is clear (11).*
- *The matrix application gives sufficient feedback to the user (14).*
- *The matrix application always functioned as I expected (20).*
- *The value and the correlation view should be visible at the same time (18).*

Based on post-discussion, it was determined that subjects found the feedback to be inadequate during the row and column movements. There should be an outline of row or column visible during the move operation to indicate what is moving. The current implementation updates the matrix view as the mouse moves, but does

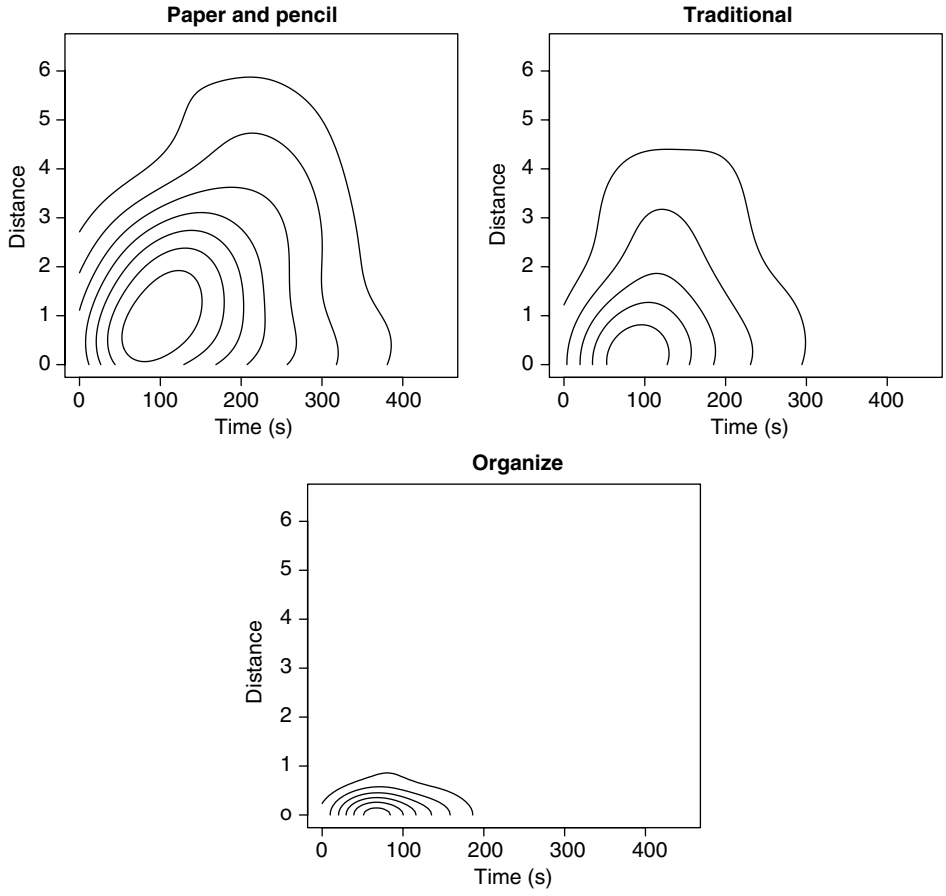

Figure 15 2D kernel density estimates for distributions of variables *Time* and *Distance*.

not indicate the current selection. This is clearly something that should be implemented to make the matrix application more usable. This problem is also related to proposition (20), because the result of a movement operation surprised the user at times.

The most negative outcome was received by the following proposition:

- *Finding similar columns with pencil and paper is easy (13).*

Half of the responses to this proposition were undecided. In the interview, these participants commented that the pencil and paper approach is just laborious, but it is not difficult to complete.

To summarize the questionnaire, it seems that the subjects found the proposed tool to be quite satisfactory. The reaction to proposition (2), *I could use this kind of application in my everyday work*, is especially promising.

Discussion The first experiment shows that the reordering and sorting capability brings about 24% reduction to the time to solve a cluster recognition task, and an approximately similar improvement to the task accuracy; although the difference in accuracy is inconclusive.

The time improvement between the traditional Bertin interface and the proposed new interface is about 23%, and about 41% when compared it to the manual alternative. The improvement in accuracy between the baseline and the new method is about 58%.

The following problems were identified in this experiment:

- Four of the 10 test matrices were too trivial – they could be solved without any kind of interaction.
- The between-subjects experimental design led to weak statistical tests.
- Many participants found the separate correlation coefficient view confusing, especially with the *Traditional* condition.
- The correlation coefficient was regarded as a difficult similarity measurement to understand.

Relatively small and easy matrices were included in this experiment for the sake of *Paper* condition. Even with this set, some of the participants were really frustrated and close to giving up.

Second experiment

Figure 17 shows the user interface for the second experiment. Now the data and the similarity matrix are displayed simultaneously, side-by-side. The similarity metric was a Euclidian distance where a white cell

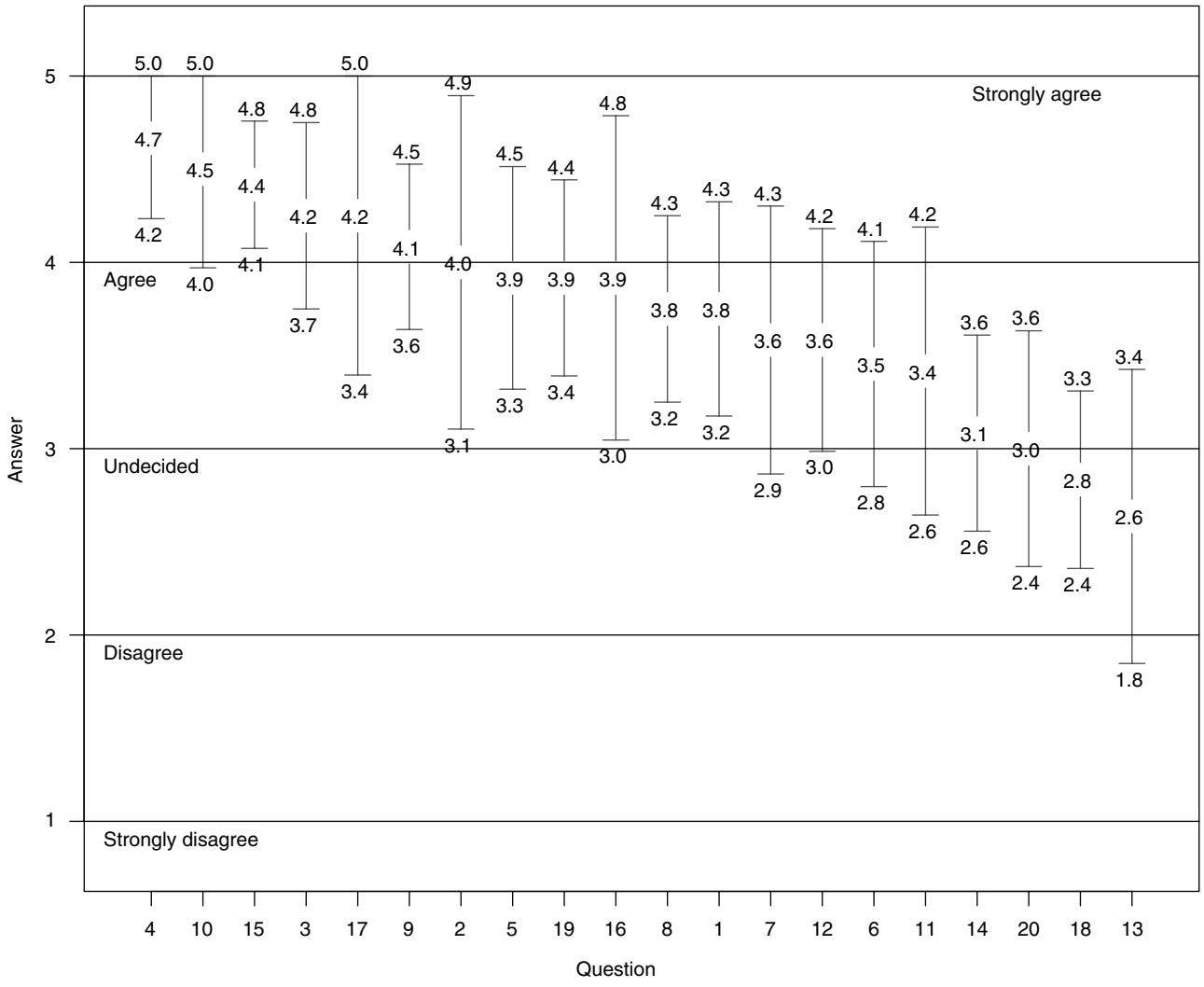

Figure 16 The 95% confidence intervals and mean values for the 20 answers of post-test questionnaire. The scale is from *Strongly disagree = 1* to *Strongly agree = 5*.

indicates a complete match. The interaction involves only moving the slider – the similarity matrix is permuted whenever the slider value changes and the new column order are copied into the data matrix also. The computation was implemented in a separate thread and it does not slow down the user interface. A rapid change in the threshold value may lose some of the intermediate states, but this does not happen in a typical use.

The rectangular areas around the diagonal in the similarity view, which are below the threshold, are emphasized by being outlined with a red rectangle. The same information is displayed in the data matrix view with a red line above the column labels. The line alternates between two elevations and can indicate overlapping rectangles as well.

Participants In all, 16 volunteer senior undergraduate and graduate students participated in the experiment as a part of a course assignment. None of these students participated in the first experiment, and the majority of the participants were foreign students.

Seven of the participants were male and nine female, and the median of their ages was 24 years, ranging from 22 to 29. The medians of 'computer use' and 'daily computer use' were 10 and 6 years, respectively. The participants were also asked to evaluate their own expertise as a computer user on a continuous scale of 0 to 100, and the median for this was 72, ranging from 43 to 95.

Apparatus The apparatus was the same as in the first experiment.

Task materials Two isomorphic sets of 15 matrices were generated with a program, with matrix size ranging from 234 to 440, the median size being 409 cells (more than twice the median size of the matrices in the first

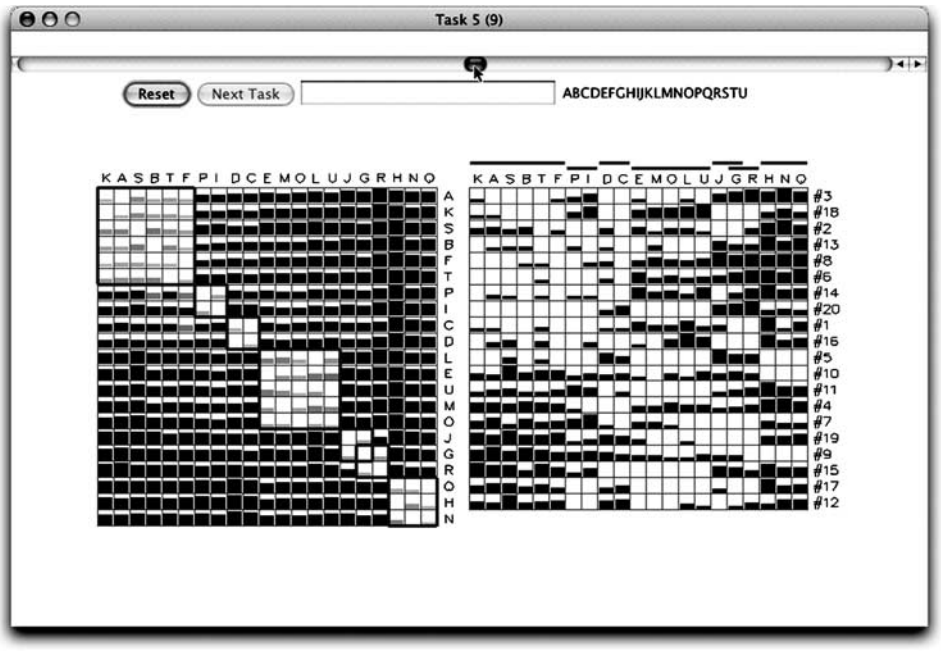

Figure 17 The user interface in the second experiment.

experiment, 179). The matrix generation process was parameterized so that the resulting matrices contained varying types of clusters. The median number of clusters in a matrix was 6 and the median size of a cluster was 3.

Procedure Participants were given the same general instructions as in the first experiment, but in English this time. The procedure was repeated 15 times using one of the interfaces and then another 15 times using the other interface. In both sets, the first three tasks were used for training and practicing, and no data were collected for them.

Participants gave the answer by filling a text field that forced them to mention all of the column names before leaving the task. The groups were indicated by leaving a space between column labels. On the right side of the text field, there is a text label that displays the yet unentered column names as a hint. As a shortcut, by writing an asterisk the participants could indicate that the not yet entered columns are all in their own groups.

Design In the experiment, there were two conditions called *Normal* and *Slider*. The *Normal* condition does not have the column similarity view, but is otherwise similar to the first experiment's *Traditional* condition.

The participants were divided randomly into two groups. One group took the *Normal* condition first, and then *Slider* and the other group had the opposite order. Counterbalancing involved the two groups and the two task sets with randomized task order, requiring the number of participants to be a multiple of four.

The experiment was treated as $2 \times 2 \times 12$ mixed design. *Group* was a between-subjects factor with two levels (eight participants per group). The within-subjects factors were *UI* with two levels (*Normal vs Slider*) and *Task* with 12 levels. The total amount of input was 8 participants/group $\times$ two groups $\times$ two layouts $\times$ 12 trials $=$ 384 observations.

Two measurements were collected for each trial: time spent (in seconds) and the given solution. The solutions were later transformed manually into distances computed from the appropriate lattice.

Results The median time participants spent in the whole experiment was 1 h and 22 min, ranging from 51 min to 1 h and 50 min. This was almost twice as much as anticipated. The participants were more patient and thorough than was expected, and many of them wanted to explore the tasks with more than one strategy. This applied to both interface styles.

Two of the participants did not complete the experiment. One session was aborted by the supervisor, and one other participant gave up commenting that the task was incomprehensible. In both cases, the problem seemed to be language-related – it appears that the supervisor failed to communicate the instructions. Two more volunteers were recruited to complete the counter-balancing.

The main effect and interactions for *Group* were not statistically significant. The grand mean for *Time* was 154 seconds. The time for the *Normal* user interface was 217 seconds, and for the *Slider* user interface, it was 91 seconds, approximately 60% improvement. The difference was statistically significant ($F_{1,15} = 8.71$, $P < 0.001$).

The grand mean for *Distance* was 2.9, indicating that about three single-item shifts were required to reach the correct solution. The distance for the *Normal* user

interface was 4.1, and for the *Slider* user interface, it was 1.7. Again, the improvement was about 60% and the difference was statistically significant ($F_{1,15} = 29.33$, $P < 0.001$).

Perhaps the most effective visualization for the difference is to look at the distributions for variables *Time* and *Distance* together. This is accomplished with a kernel density plot shown in Figure 18.

The two task sets that were treated as similar in the analysis were inspected for a statistically significant difference with respect to *Time* or *Distance*, but none were found.

Subjective satisfaction

The usability of the two interfaces and the attitude towards them was evaluated with a post-experiment questionnaire containing propositions that were rated on a five-point Likert scale. Some of the propositions have been rewritten in the following discussion as in the case of the first experiment, and the control propositions have been merged. The 95% confidence intervals for the answers can be seen in Figure 19.

The most positive feedback was given for the following propositions:

- *The distance view in the Slider user interface makes it easy to spot the similar columns (3).*
- *Two different views does not make the Slider user interface confusing, and the connection between views is easy to understand (5).*

The distance view was clearly found helpful, and the simultaneous visibility of matrices was found a better solution than the previous one-at-the-time approach.

The propositions

- *The slider makes it more interesting to search for the similar columns (7).*
- *The effect of slider movement in the Slider user interface is easy to understand (4), and*
- *The Slider user interface gives a clearer overview of the data set (6)*

got a clear agreement and indicate that there are no conceptual problems in the user interface. Likewise, users seemed to like the interface and found it helpful in performing the task.

Most of the participants found the traditional permutation user interface acceptable for the task, as the outcome of the following proposition shows:

- *It is easy to move and sort the matrix's rows and the columns, and it is easy to find the similar columns (2).*

However, three participants disagreed with this proposition.

The most undecided rating was given to the first proposition:

- *Overall, the experimental application was easy to use and performed as expected, and did not do unexpected things (1).*

The negative response has a two-fold explanation. When interviewed, four of the participants said that the reason for the rating was the uncomfortable mouse behavior – especially the proportional acceleration was different from what they were used to. The other reason was the heuristic nature of the reordering algorithm. For some participants, it was difficult to accept that the same setting of the slider would sometimes produce a new ordering.

Discussion Results from the second experiment suggest an approximate 60% improvement in both time and accuracy over the classic manual user interface. The results are also consistent: the difference was in the same direction with each task and with each participant. The questionnaire also shows that this was perceived by the participants as well.

It seems that the simultaneous display of the data matrix and the similarity measure matrix was the right decision. Several participants commented in the first experiment that the connection between views was difficult to understand, but now there was no problem.

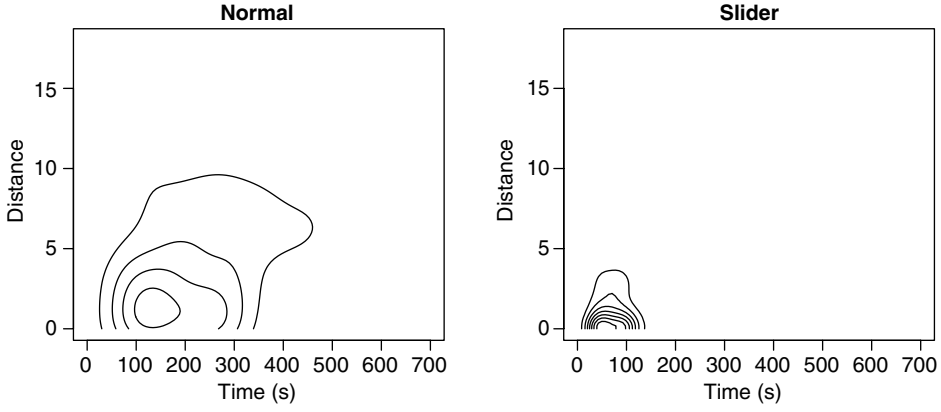

Figure 18 2D kernel density estimates for distributions of variables *Time* and *Distance.*

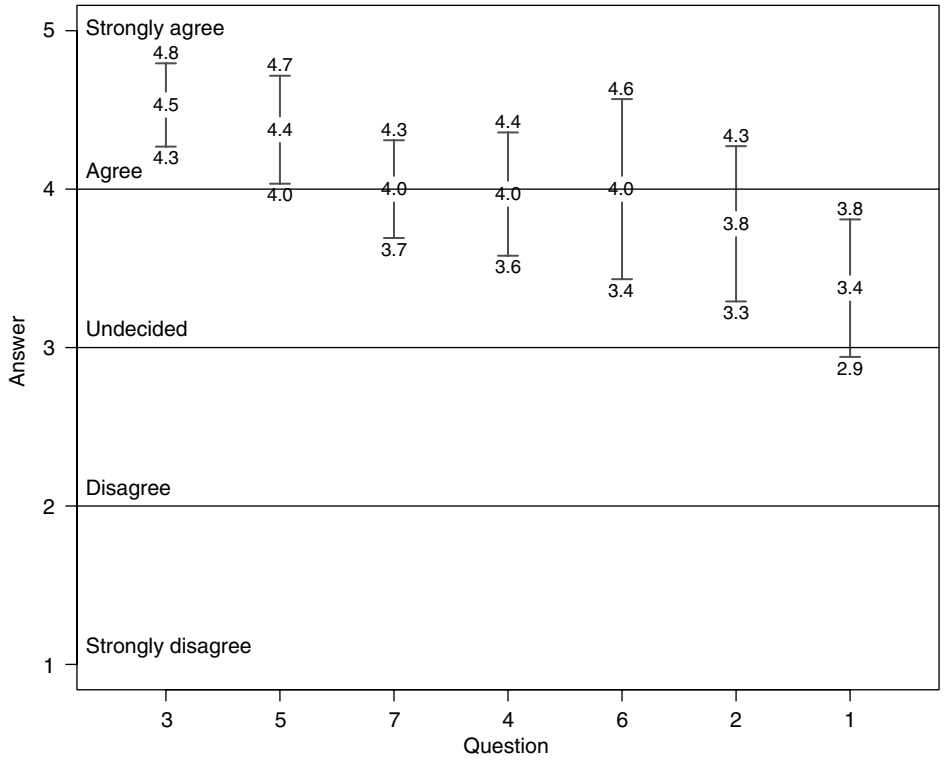

Figure 19 The 95% confidence intervals for the seven propositions in the questionnaire.

Another major change between conditions *Organize* and *Slider* was the elimination of the button for reordering. Some of the participants commented during the experiment and in the interview that the continuous reordering feels a bit disturbing, although this does not show in the questionnaire results. They felt that a small change in the slider should not result a major change in the matrix ordering. This was the initial reaction, and most of the participants were able to accept this later in the experiment as a characteristic of the user interface. This behavior is due to the nature of the barycenter heuristic and cannot be avoided.

The only difference between conditions *Traditional* and *Normal* was the removal of the column similarity view. The column correlation coefficient view in condition *Traditional* was found confusing by the participants, and many of them ignored it completely.

Summary and discussion

The *rearrangement* is one of the fundamental techniques of explorative information visualization. Although the first experiment failed to show consistent difference between the two computer interfaces, the experiment shows the effect of rearrangement between static and dynamic image of a matrix. The result indicates that the simple ability to reorder the rows and columns of a matrix will improve the task performance significantly. Both task execution time and accuracy improved by about 24% when the rearrangement capability was added, although the difference in accuracy was inconclusive.

The second experiment shows that the proposed interface is about 60% more efficient in both task execution time and solution accuracy than the baseline, the manual permutation interface. In this experiment, as the test cases were too large and complex to be explored with paper and pencil only, the comparison was limited to the two computer interfaces.

The achieved improvement is large enough to be of practical significance. The proposed technique could be developed into an interactive tool for cluster analysis, and provide the rapid exploration of different solutions.

The current prototype implementation was constructed to see the user reaction to the lively output of the permutation heuristic, and to see if the performance of the heuristic is fast enough for rapid interaction. The users were initially surprised about the rapidly changing display, but seemed to get used to it in a short time. The time complexity of the barycenter heuristic is in order of the number of cases multiplied by the number of rows, with a considerable constant. We are constructing a more carefully tuned implementation to explore the upper limit of matrix size for the rapid interaction.

The experiments gave several ideas for further exploration. One obvious enhancement is to allow the reordering of any rectangular sub-area in a matrix. This allows the subsets of cases and variables to be inspected.[22]

The current slider implementation adjusts the upper limit of similarity constraint only. An obvious extension to this would be to allow a range to be selected, and to provide a selection of metric.

Another feature that was assumed useful in the experiments is the zoom functionality. In order to enhance the recognition of patterns, many of the participants read a matrix by backing off and squinting at it. This could by supported easily by implementing a zoom function that would allow observing the matrix conveniently from various altitudes.

Conclusions

A survey of known applications of the reorderable matrix was conducted. In the survey, we limited our discussion to adaptations that are close to the original idea – otherwise there would have been many more. Still, we believe that we have covered the essential developments of matrix reordering methods.

Various problems related to the reordering of the reorderable matrix are NP-complete. This suggests that it is reasonable to try heuristic approaches in reordering the matrix. We have shown that applying the barycenter heuristic for minimizing the number of edge crossings when drawing bipartite graphs is well suited for reordering the reorderable matrix.

A new method for interactively reconstructing a reorderable matrix was developed. We implemented this new method and conducted a user study where two variations of the method and the former techniques were compared. In these experiments, the task was to perform cluster analysis on a matrix. The results show that the proposed method produces results that are more accurate in less time than the former methods. Moreover, the users found the method usable and interesting to use.

Acknowledgments

We thank Kari-Jouko Räihä and Natalie Jhaveri for their comments. This work was supported by the Academy of Finland (projects 35025 and 178099) and by the National Technology Agency Tekes (project 20287).

References

1 Bertin J. *Graphics and Graphic Information Processing*. Walter de Gruyter & Co.: Berlin, 1981, (Originally *La graphique et le traitemente graphique de l'information*, 1967, translated into English by William J. Berg and Paul Scott).
2 Bertin J. *Semiology of Graphics – Diagrams Networks Maps*. The University of Wisconsin Press: Madison, 1983 (Originally *Sémiologue graphique*, 1967, translated into English by William J. Berg).
3 Bertin J. Matrix theory of graphics. *Information Design Journal* 2001; **10**: 5–19.
4 Spence R. *Information Visualization*. ACM Press: Essex, 2001.
5 Adams W, Daru R. MATCHMAKER – an instrument to match demand and supply of buildings and revealing discrepancies. In: *Proceedings of DDSS'94*, Vaals, the Netherlands, 1994.
6 Veenendaal MH. Subjective orderliness versus mathematically defined order in graphical data matrices. In: *Proceedings of DDSS'94*, Vaals, the Netherlands, 1994.
7 Ulrich K, Eppinger S. *Product Design and Development*. McGraw-Hill: New York, 1999.
8 Mäkinen E, Siirtola H. Reordering the reorderable matrix as an algorithmic problem. In: *Theory and Application of Diagrams, Diagrams 2000, Lecture Notes in Artificial Intelligence 1889,* 2000; 453–467.
9 Daru R, Adams WTCF. Interactive graphic heuristic procedure. In: Kimura F, Rolstadas A, (Eds). *Computer Applications in Production and Engineering*. Elsevier Science Publishers: Amsterdam, 1989; 783–807.
10 Snijder HPS. The use of genetic algorithms in spatial optimisation problems. In: *Proceedings of DDSS'94*, Vaals, the Netherlands, 1994.
11 Lohninger H. Multivariate exploratory data analysis by means of INSPECT. In: Moll R (Ed). *Software Development in Chemistry*. Vol. 9. *Proceedings of the 9th Workshop 'Computers in Chemistry'*, Bitterfeld 1994, GDCH, Frankfurt, 1995; 91–98.
12 Rao R, Card S. The Table Lens: merging graphical and symbolic representations in an interactive Focus+Context visualization for tabular data. In: *Proceedings of CHI'94* (Boston, Massachusetts, USA), ACM Press: New York, 1994; 318–322.
13 Schmid C, Hinterberger H. Comparative multivariate visualization across conceptually different graphic displays. In: *Proceedings of the Eight International Working Conference on Scientific and Statistical Database Management, SSDBM'94* (Charlottesville, Virginia, USA), IEEE: Los Alamitos, CA, USA, 1994.
14 Eades P, Wormald NC. Edge crossings in drawings of bipartite graphs. *Algorithmica* 1994; **11**: 379–403.
15 Di Battista G, Eades P, Tamassia R, Tollis IG. Algorithms for graph drawing algorithms: an annotated bibliography. *Computational Geometry: Theory and Applications* 1994; **4**: 235–282.
16 Siirtola H. Interaction with the reorderable matrix. In: Banissi E, Khosrowshahi F, Sarfraz M, Tatham E, Ursyn A (Eds). *Information Visualization IV'99, Proceedings of the International Conference on Information Visualization* (London, UK), IEEE Computer Society Press: Silver Spring, MD, 1999; 272–277.
17 Ware C. *Information Visualization – Perception for Design*. Morgan-Kaufmann: San Francisco, 2004.
18 Venables WN, Ripley BD. *Modern Applied Statistics with S*. Springer: Heidelberg, 2002.
19 Sawitzki G. Extensible statistical software: on a voyage to Oberon. *Journal of Computational and Graphical Statistics* 1996; **5**: 263–283.
20 R Development Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org/, 2003.
21 Erickson MJ. *Introduction to Combinatorics*. John Wiley & Sons: New York, 1996.
22 Siirtola H. Interactive cluster analysis. In: Banissi E *et al.* (Eds). *Information Visualization IV 2004, Proceedings of the Eight International Conference on Information Visualization* : (Los Alamitos, CA, USA), IEEE Computer Society Press: Silver Srping, MD, 2004; 471–476.