InfoMapper: Coping With the Curse of Dimensionality in Information Visualization

Mark A. Foltz
MIT Artificial Intelligence Laboratory
200 Technology Square Room 808
Cambridge, MA USA 02139
mfoltz@ai.mit.edu

Audrey Lee
Department of Computer Science
Wellesley College
106 Central Street
Wellesley, MA 02481 USA
alee@wellesley.edu

ABSTRACT

Two goals of information visualization are the creation of automated and easily perceptible information displays. However, visualization tools are often faced with the *curse of dimensionality*: there are many more kinds of variation in the information than visual dimensions in the display medium. We have developed InfoMapper, an automated visualization system that supports the highly interactive exploration of the information space. InfoMapper allows the user to select and remove information dimensions on demand, so that the user is freed to focus on finding the information most relevant to his task, and can let the tool find a perceptible way to visualize that information.

InfoMapper is motivated by the Query by Attention framework for visual filtering interfaces. Query by Attention interfaces exploit the properties of the human visual system to allow the user to rapidly and effortlessly query information by controlling visual attention, instead of manipulating an interface. Here, we present a motivating scenario for the InfoMapper, the constraint satisfaction algorithm underlying the system, and discuss its implementation.

KEYWORDS: knowledge-based information design, perceptual user interfaces, information visualization, decision-support interfaces, sensemaking interfaces

INTRODUCTION

Two goals of information visualization are the creation of automated and easily perceived information displays. Automating the visualization process spares the user the task of information design, so he can focus on his goals for exploring the information. Effective visualizations are also easily perceived: they engage our inherent and specially tuned capabilities for visual perception, so that interpreting the information becomes an almost effortless perceptual task.

However, often information designers face the *curse of dimensionality*: there are many more kinds of information in the data to be displayed than visual dimensions in the display medium. Adding more and more visual variations to convey all the information results in a visualization too complicated to interpret easily, forcing the choice of a subset of information to be shown. But, the designer may not know ahead of time which subset to show; it depends on the user's context, goals, and the particular content of the information.

In these situations, we would like to give the user the freedom to choose the subset of information to view, without requiring him to design a new visualization for each subset. To meet this challenge we have developed InfoMapper, a system that creates automated, easily perceived visualizations while allowing the user to rapidly explore subsets of information attributes. (Attributes are fields describing properties of the items displayed, such as *Title*, *Year*, and *Won-An-Oscar?* for a movie.) The user can make rapid decisions about which information attributes are relevant and irrelevant, and the system automatically creates visualizations with the chosen information in response. InfoMapper is thus well-suited for situations where the information attributes relevant to the user are not known ahead of time, such as when purchasing a car or choosing a movie.

The system uses a novel constraint-satisfaction algorithm that automatically creates visualizations as the user adds and removes attributes from the display. The system does so by mapping the chosen attributes to the available visual dimensions in the display, in accordance with constraints motivated by the Query By Attention framework [5]. It also tries to keep the meaning of display elements consistent as the selected set of attributes is changed. InfoMapper is capable of producing scatterplot-style designs such as the ones in Figure 1.

Overview of Paper

First, we present background information on Query by Attention visual filtering interfaces, which motivated the development of InfoMapper. Then, we illustrate InfoMapper by a scenario of a user exploring a multi-attribute data set of cars he might want to purchase. Next, we present our constraint-

satisfaction algorithm that underlies the InfoMapper system. Finally, we discuss the implementation and usage features of InfoMapper.

BACKGROUND: QUERY BY ATTENTION

InfoMapper is motivated by the Query By Attention framework for visual information filtering interfaces. The Query By Attention framework proposes using visual dimensions like shape, color, and position that are known to support efficient visual search: subjects can find matching items in a display rapidly, regardless of the number of distracting elements. To develop this framework, we surveyed the scientific literature on visual search [11] and developed an information design method based on our results.

Some of the visual dimensions we studied include:

Position. A user can control visual attention to search within a spatially delimited region, making it appropriate to map to the X- or Y-axis attributes that are queried by range.

Color. Color is widely used to distinguish symbols in maps. The colors at the extrema of opponent processing [7, p. 113] (red/green, blue/yellow, and black/white) are considered good candidates for efficient search. Experimental results has shown that search for a color target among as many as nine distractor colors is efficient if the colors are spread far apart in color space [9].

Shape. Some determining features for distinguishable shapes include line termination (presence/absence), closure, holes, and possibly intersection [11, pp. 31-4]. Most experiments described report efficient search with homogeneous distractors, so a conservative approach would map shapes like X and O to a binary attribute. Orientation also supports efficient search, for example using bars oriented at 0 or 90 degrees.

Depth. Pictorial depth cues such as shading, occlusion, or shadows can support efficient search [11, p. 39]. This cue is best used for binary features, as search in more than two depth planes is less likely to be efficient.

The Query By Attention method maps information types to visual dimensions like these to permit visual searches for information that are perceptually easy. Query By Attention is an alternative to standard database query interfaces. In database interfaces, the user must specify criteria, submit the query, and wait for tabular results before querying again. Instead, in Query by Attention interfaces, the user can find the matching items at a glance, and if he wants to change the parameters of the query, he simply attends to a different set of properties. Information exploration is done by controlling visual attention, without manipulating an interface at all.

SCENARIO

InfoMapper supports multi-criteria decision processes, like purchasing a car, where the user wants to find a set of items matching his criteria, but doesn't know which criteria at first. This is why it is crucial to support the incremental exploration of the data space. InfoMapper is also particularly suited for situations that have both quantitative and qualitative criteria, where the program can use its flexibility in mapping visual dimensions by data type.

Our first scenario shows a user who wishes to learn about cars he would be interested in purchasing. When he starts, he has a general idea about what he would like in a car, but does not have a fixed set of criteria for choosing one. Our example uses a data set of recent car models with nine attributes, more that the maximum of six visual dimensions that the InfoMapper can render. (In this case, over 180,000 attribute-dimension mappings are possible).

We use this scenario to illustrate the four goals of our constraint satisfaction algorithm. The detailed interface actions that play out this scenario are discussed in the implementation section.

Goal #1: Produce a reasonable initial visualization. Given no information from the user about the relative importance of the attributes, InfoMapper tries to create a default visualization using as many of the attributes as it can. The initial visualization created for a data set of recent car models is shown in Figure 1, top. Here, InfoMapper has mapped all six visual dimensions, creating a visualization that shows the maker, type, passenger capacity, number of cylinders, cargo capacity, and weight for each car.

For a variety of reasons believe it is important to begin with an initial visualization, even if imperfect, rather than a blank screen. This approach allows the user to begin learning about the information without any effort (for example, how many items there are in the data set); it provides an initial visualization that may suggest where to go next, or surprise the user with what it presents; and the initial visualization may turn out to be close to what the user wants.

In this case, by examining the initial visualization (Figure 1 top), our user finds the car with the largest passenger capacity is the Ford Excursion, the top-most plus symbol in the display.

Goal #2: Allow incremental exploration. In this scenario, although information about cargo capacity and weight is interesting to our user, he cares most about the cars' miles per gallon and price. He drags those two attributes from the *not-visualized* list to the *visualized* list, and removes car type and passenger capacity (which he decides are not as relevant at this point).

The display updates to reflect his preferences, as in Figure 1, bottom left. Miles per gallon is assigned to the X axis, and price to the Y axis. The attributes cargo capacity and weight are no longer used, since the algorithm can't find a way to render them in the display, so they are moved to the not-visualized list.

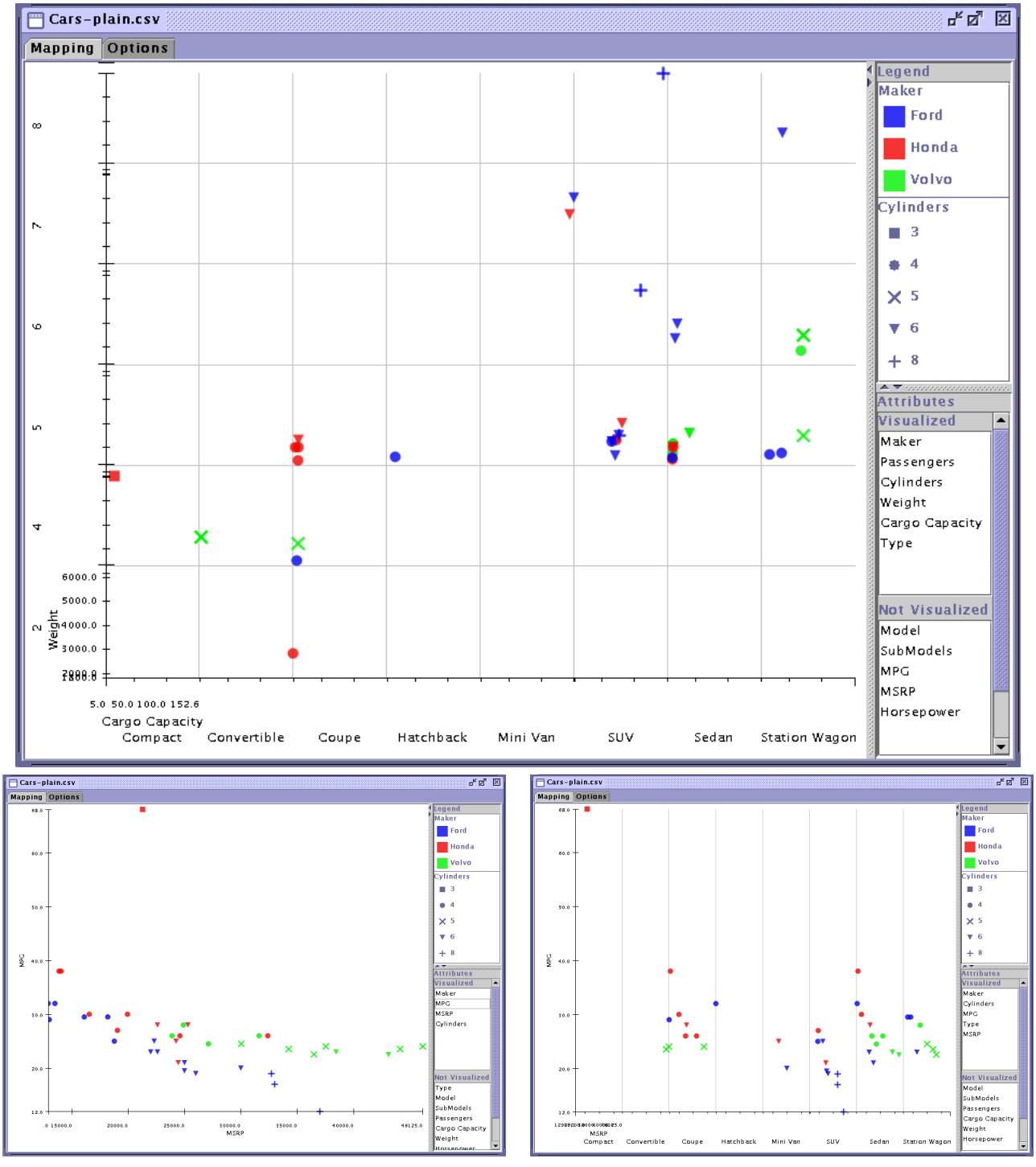

Figure 1: Top: The initial visualization in the car-buying scenario. The display area is divided into a grid of regions, in which the X axis is partitioned by car type, and the Y axis by passenger capacity. Within each region, cargo capacity and weight are plotted on the X and Y axes. Bottom left: The display after the user has chosen miles per gallon and price as preferred attributes, and removed passenger capacity and type. Bottom right: The display after the user has restored car type.

This illustrates another goal of our algorithm, which is to allow the user to tell the system what is important in the information and have the system design a visualization that conveys the relevant information.

Goal #3: Update the display stably. Now suppose that the user does want to see car type as well as miles per gallon, price, car maker, and number of cylinders. When he adds it, the system does not change the meaning of color or shape; instead, it subdivides an axis to render the new attribute, as in Figure 1, bottom right.

This illustrates another goal of our algorithm: to keep the mapping of existing dimensions as stable as possible even as attributes are added and removed from the visualization. The user has invested some effort in learning which colors and shapes map to which values; reassigning these will confuse the user and throw away what he has learned about these associations, and hence should be avoided whenever possible.

Goal #4: Add new constraints easily. Our example uses position, color hue, and shape type to convey information about the data. However, there are many more ways to present information we would like to make available to the user. For example, a natural way to convey the capacity of a car would be to vary the size of its icon's shape.

Increasing the vocabulary of attribute types and visual dimensions requires adding new design knowledge to the system, but ideally should not significantly change the underlying mechanism for satisfying the constraints, or require rewriting all of the existing constraints in the system.

ALGORITHM

Our constraint satisfaction algorithm is designed to fulfill those four goals. Given lists of attributes to visualize and visual dimensions, it finds a one-to-one mapping between them that satisfies the constraints.

InfoMapper divides the visual design space into a set of channels. Conceptually, a channel is a stream of human visual processing, that can be filtered along a number of visual dimensions, such as color hue.

Table 1 summarizes the visual dimensions for which constraints have been written; InfoMapper can render X and Y position, X and Y regions[1], color hue, and shape type.

Information attributes are assigned one of six types: real, integer, date, enumerated (chosen from a fixed set of values), binary, and text (arbitrary text, which is not mapped).

To find the mapping, the algorithm uses a set of design constraints devised by the authors to match the design method proposed in the Query By Attention framework [5]. The constraints try to produce designs that take advantage of the inherent filtering capabilities of the human visual system.

[1] InfoMapper can subdivide the X and Y axes to create a grid of regions, and sort data items into those regions according to the values of two discrete attributes.

Channel	Dimensions
Space	X position
	Y position
	X region
	Y region
Color	Hue
	Saturation
	Value
Shape	Type
	Size
	Texture

Table 1: Visual channels and dimensions.

Constraint Types

InfoMapper uses seventeen constraints to determine the validity of attribute-dimension mappings. These constraints are divided into four kinds:

Dimension-attribute compatibility constraints. These constraints ensure that visual dimensions convey information appropriate to their type. Since color hues and shape types are chosen from fixed sets, they are constrained to map to enumerated information types. The use of X and Y regions creates a discrete grid over the display space and so they are also constrained to map to enumerated types. X and Y position must be mapped to real-valued types to fully determine the position of each information item.

Dimension-dimension compatibility constraints. These require that dimensions which overlap in perceptual space are not both used in the display. These constraints are appropriate when using both dimensions would prevent the user from accurately perceiving the underlying information values. For example, we use this constraint to prevent color value and color saturation from both being used in a mapping, since a change in either varies the apparent intensity of an underlying hue.

Dimension-dimension dependencies. Dimension-dimension dependencies ensure that a mapping fully determines the appearance of each data item for rendering. For example, if the algorithm chooses to subdivide the display space into regions, it is still necessary to determine the exact position of the item within each region. To make sure this is the case, X region depends on X position, and Y region on Y position.

Mapping consistency constraints. Mapping consistency constraints keep the meanings of some visual dimensions the same between updates to the mapping. We constrain color hue and shape type in this way, so the user does not have to repeatedly re-learn what shape and color represent in the display.

Adding new constraints. Constraints are modular in InfoMapper, in keeping with Goal #4. For example, the authors found that occasionally an enumerated attribute with more that five values would be mapped to the five shape types available, creating an ambiguous display. To resolve this, we added a new dimension-attribute compatibility constraint that prevents this from occurring. No changes to other constraints were necessary.

Constraint Satisfaction Algorithm

MAP, our algorithm for finding attribute-dimension mappings, takes as inputs a list of visual dimensions, a list of information attributes, a set of constraints like those described above, and the current attribute-dimension mapping. The output is a new attribute-dimension mapping that satisfies the constraints, which is used by InfoMapper to render the current data set.

The input list of attributes is sorted by the user's preferences, with the explicitly preferred attributes appearing first (i.e., those the user has dragged from the not-visualized to visualized list). The input list of visual dimensions is sorted by a preference ordering intended to produce more pleasing designs. For example, color is assigned before shape because overlapping items of different colors are more easily perceived (through alpha blending) than overlapping items of different shapes.

MAP proceeds as follows.

```
ALGORITHM MAP(D,A,S,M)
INPUTS: A list of unmapped visual dimensions D
        A list of unmapped information attributes A
        A set of constraints S
        The current mapping M, initially empty
OUTPUT: A one-to-one mapping M(a) = d of attributes
        to dimensions

for each dimension d in D
  for each attribute a in A
    if (a,d) is not
        dimension-attribute compatible, then
      continue A loop;
    endif
    for each dimension d' in M
      if (d,d') is not
          dimension-dimension compatible, then
        continue A loop;
      endif
    endfor
    // All the constraints relevant to (a,d) are
    // satisfied at this point.
    // Add (a,d) to the mapping.
    M' = M + (a,d)
    A' = A - a
    D' = D - d
    for each dimension d' in D,
      if d' depends on d and is not in M', then
         move d' to the beginning of D'
      endif
    endfor
    return MAP(D',A',S,M')
  endfor
endfor
return M
```

Although it is exponential in complexity, in practice it finds solutions quickly (under one second on a current workstation). This is in part because it is a satisficing, not an optimizing algorithm; it finds the first mapping consistent with the constraints, not the mapping with the most elements (or best by some other quality measure).

Occasionally, a subproblem is over-constrained: A and D are nonempty, but no further mappings can be made. This prevents poor design choices from being made, but the user may want the poor design anyway because it has the information he wants. In the future, we plan to enable some constraints to be soft, so that they can be lifted when the problem becomes over-constrained in this way.

Another algorithm, MAP-UPDATE, updates the current mapping when the user adds or removes an attribute from A. It initializes the new mapping with dimensions from the old mapping that are constrained to be consistent between updates. These consistency constraints are intended to carry over the meanings of color and shape associations the user has already learned. It then invokes MAP to fill in the rest of the mapping.

```
ALGORITHM MAP-UPDATE(D,A,M,S)
INPUTS: A sorted dimension list D
        An updated, sorted attribute list A
        The current mapping M(a) = d
        A set of constraints S
OUTPUT: A one-to-one mapping M(a) = d of attributes
        to dimensions

M' = {} // an empty mapping
for each dimension d in M
  if d is to be kept consistent, then
    a' = M^-1(d)
    if a' is in A, then
       M' = M' + (a', d)
       A = A - a'
       D = D - d
    endif
  endif
endfor
return MAP(D,A,S,M')
```

IMPLEMENTATION

InfoMapper is written in Java. It takes as input comma-separated text files containing the data to be visualized; multiple files may be visualized at once on the InfoMapper virtual desktop. When a file is opened, its contents are parsed and each field is categorized into one of the attribute types. The values of each attribute are analyzed to guess the best attribute type to use (though the user can override this choice). Note that the system requires at least two real-valued attributes in the data set, to fully determine the (x,y) position of each data item.

Once the file is parsed, the MAP algorithm is run to find an initial attribute-dimension mapping and the display rendered. The axes are labeled and marked with "logical" tick marks and values, i.e. with numeric increments like 2, 2.5, 5, or 10 commonly used in charts and graphs. A legend is placed at the right hand side of the display.

The user selects and removes attributes from the display in a direct-manipulation fashion, by dragging attributes between the visualized and not-visualized lists at the lower right of the display.

The user can also interact with the data directly. Moving the mouse over an item produces a details-on-demand popup window, as in Figure 2. The user can customize the contents of this window by checking off a list of attributes to include, or by writing a custom HTML format string. The format string contains tokens which are substituted with values from the moused-over data item. The user can also click and drag to create a rectangular selection box, which pops up a a sortable table in a separate window with the enclosed data (Figure 3).

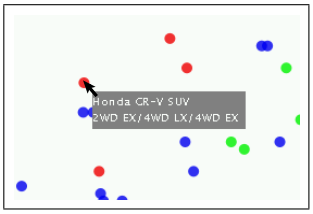

Figure 2: A custom details-on-demand window for the car visualization.

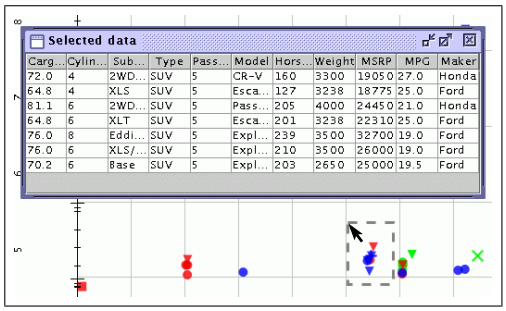

Figure 3: A pop-up sortable table of selected data.

CONCLUSIONS

Related Work

There is a rich tradition of work that articulates principles for mapping data variation to ink variation, such as [10] and [3]. Work on automated visualization systems aims to capture some of those principles in systems that automatically create effective visualizations [1, 8, 6]. Our work extends previous work by allowing the highly interactive exploration of the space of information dimensions. The algorithm allows the user the freedom to choose which information is relevant, when there are too many information dimensions to visualize at once. We also contribute information design constraints grounded in experimental results from visual psychology that suggest which visual channels that support effective visual search. These channels permit almost-instantaneous filtering of display items based on visual properties such as shape, color, and position.

Another approach is to reduce the dimensionality of the data with statistical or clustering techniques such as clustering or principal-components analysis. InfoMapper currently does not compute dimension-reducing statistics, but they are compatible with our algorithm. If these statistics are available as attributes derived from the original data, and the user decides they are useful, the derived attributes can be rendered instead of the original ones. It would be interesting to apply these algorithms automatically as part of the constraint satisfaction process, for example to allow a real valued attribute to map to a discrete shape-type by clustering the values.

Future Work

We intend to conduct a usability study of InfoMapper. The particular hypotheses we would like to address are:

1. Do users understand InfoMapper's conceptual model, i.e. does the system behave predictably and intuitively when users select or reject attributes?

2. Does the system support multi-criteria decision making, such as a purchasing decision?

3. Do users prefer the system to a conventional, text-based database query interface for such a task?

We would also like to enrich the vocabulary of visual dimensions available to render information. Our current system can render only fixed set of color hues and shapes. Texture, orientation, and motion are all visual properties that support rapid visual search. Color value, color saturation, and shape size can also convey information when rapid search is not as important (since they may confound other, searchable properties). Our extensible constraint algorithm supports this kind of incremental evolution of the InfoMapper's capabilities.

However, simply adding more visual dimensions may not be the best way to render more information; cognitive constraints ultimately limit the complexity and interpretability of visualizations. Adding more visual channels to the palette does increase the flexibility of the algorithm and the variety of possible displays.

An alternative is to allow multiple, spatially distinct views of the same data. This is another route to coping with the curse of dimensionality, at the expense of introducing multiple display areas which can't be attended simultaneously. However,

techniques like multiple view brushing [4, 2] can perceptually re-link the views to show data correlations in multiple displays.

Summary

We have described an algorithm and its implementation for addressing the curse of dimensionality problem in information visualization: when there are more kinds of variation in the information than visual channels to display them. Our algorithm produces a reasonable initial visualization, allows the user to explore the information space by selecting and removing information attributes, and tries to maintain the meaning of the mapping when possible. The algorithm combines design knowledge about how to render different information types effectively with the user's preferences about what information to include and remove. InfoMapper lets the user focus on choosing the information relevant for his data exploration and decision-making tasks, instead of designing visualizations.

ACKNOWLEDGMENTS

The authors thank Randy Davis for detailed comments on a draft of this paper and the MIT AI/LCS Project Oxygen partners for sponsoring this research.

REFERENCES

1. AHLBERG, C. Spotfire: An information exploration environment. *SIGMOD Record 25*, 4 (December 1996), 25–29.

2. BECKER, R. A., AND CLEVELAND, W. S. Brushing scatterplots. *Technometrics 29*, 2 (1987).

3. BERTIN, J. *Semiology of Graphics: Diagrams, Networks, Maps [orig. Semiologie Graphique]*. University of Wisconsin Press, Madison, WI, 1983. Translated by William J. Berg.

4. DERTHICK, M., KOLOJEJCHICK, J., AND ROTH, S. F. An interactive visual environment for exploring data. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '97)* (1997), pp. 189–198.

5. FOLTZ, M. A., AND DAVIS, R. Query by attention: Visually searchable information maps. In *Proc. Fifth International Conference on Information Visualisation* (London, 2001).

6. MACKINLAY, J. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics 5*, 2 (April 1987), 110–141.

7. PALMER, S. E. *Vision Science: Photons to Phenomenology*. The MIT Press, Cambridge, MA, 1999.

8. ROTH, S. F., KOLOJEJCHICK, J., MATTIS, J., AND GOLDSTEIN, J. Interactive graphic design using automatic presentation knowledge. In *Proceedings of CHI '94: Human Factors in Computing Systems* (Boston, Massachusetts, 1994), pp. 112–117.

9. SMALLMAN, H. S., AND BOYNTON, R. M. Segregation of basic color in information displays. *Journal of the Optical Society of America A 7*, 10 (1990), 1985–1994.

10. TUFTE, E. R. *Envisioning Information*. Graphics Press, Cheshire, Connecticut, 1990.

11. WOLFE, J. M. Visual search. In *Attention*, H. Pashler, Ed. University College London Press, London, 1996.