

ActiveGraph: A Digital Library Visualization Tool

LINN MARKS, JEREMY A.T. HUSSELL, TAMARA M. MCMAHON, RICHARD E. LUCE

*Research Library, Los Alamos National Laboratory, Los Alamos, New Mexico
87545*

Email: linn@lanl.gov, Telephone: 505-665-1035, Fax: 505-665-6452

Abstract

ActiveGraph is an information visualization tool designed to provide users with a concise, customizable view of objects in a digital library. A set of digital library objects is represented as a data set in a two- or three-dimensional scatter plot. The data set can represent any digital library objects in any medium: books, journals, papers, images, web resources, or even entire databases. Since ActiveGraph is intended for use in the context of digital libraries, data attributes consist for the most part of metadata fields such as title, author, date of publication, and number of citations. Data attributes are mapped to six visual attributes of the scatter plot: the X-, Y-, and Z-axes, color, size, and shape. The metadata for a selected data point are displayed in a control panel on the right-hand side of the screen. Users can edit the metadata and even add new metadata fields. Thus ActiveGraph allows users to both view and customize the contents of a library. Because of its flexibility in handling digital library objects and metadata of different types, ActiveGraph can be used in a variety of digital library applications. In this paper, we describe two: LibGraph, which is a visualization of a collaborative library, and CiteGraph, which is a visualization of citation statistics. Both applications were designed for use by scientists and engineers, for whom scatter plots are familiar and intuitive visualizations.

Keywords

Information visualization, interactive scatter plot, digital library, collaborative library, citation analysis

Introduction

As digital libraries have become larger and more complex, sorting through the content has become more difficult. Users typically sort through subsets of the content in the form of lists representing specific collections, personal libraries, or search results.

Sorting through lists is not efficient, however, and list interfaces do not take advantage of design techniques unique to digital information environments. At the Los Alamos National Laboratory, for example, scientists and engineers view the contents of their personal library and search results from ISI SciSearch, Inspec, BIOSIS and other databases in the form of lists. On a 1280 x 1024 monitor, they can view fewer than 20 items at a time. These kinds of list interfaces are typical of digital libraries and web search engines such as Google and Yahoo, but they are not visually efficient: users have to scroll or page in order to see all of the items on the list. They are not cognitively efficient, either, as paging and scrolling add cognitive overhead to the task of analyzing and comparing the items in the set.

In contrast, interfaces that use the techniques of information visualization can be designed so that they are both visually and cognitively efficient. Where a list interface displays fewer than twenty digital object representations per screen, a visual interface can display 200 or 2000: one or two orders of magnitude more. By using techniques unique to digital information environments, each object in the personal library or search results can be represented by a few pixels on the screen and each of these abstract representations can be linked to metadata and, in some cases, to the object, itself (for example, a journal article in the form of a PDF).

Linking abstract representations to objects, or compressed to expanded forms, is one of the characteristics of digital information environments [9]. While virtually all interfaces involve linking compressed to expanded forms, visual interfaces can be more efficient than list interfaces because they can use representations that are more abstract and compressed: a few pixels instead of a few words. Tufte refers to the importance of maximizing the data-ink ratio [22, 23]. By analogy, in digital information environments it is important to maximize the data-pixel ratio.

Visual interfaces that maximize the data-pixel ratio and represent a set of digital library objects in a visually efficient form allow users to view and analyze the set of objects without scrolling or paging. This simplifies visual scanning and facilitates pattern detection [2]. In order to ensure that the patterns are meaningful, the representations, themselves, have to be meaningful. They have to display data sets without distortion and with graphical integrity [22, 23].

Scatter plots

Scatter plots have been used for this purpose for the last two centuries and are among the most common forms of data graphics in scientific literature [22]. They provide an overview of a data set and show the distribution of data points clearly, revealing clusters and statistical outliers [8, 22]. By doing so, they make it possible for users to perceive meaningful patterns in the data.

Figure 1 shows an ActiveGraph scatter plot of citation data for postdoctoral researchers derived from the LANL digital library. For each paper published from 1998-2002 by a postdoc at LANL, the postdoc's last name is mapped to the X-axis and the number of citations is mapped to the Y-axis. The scatter plot reveals that one paper was cited more than any other paper, many papers were cited frequently, and most papers were cited a few times or not at all. Users can quickly scan the scatter plot, notice the clusters and outlier, and ascribe meaning to the patterns.

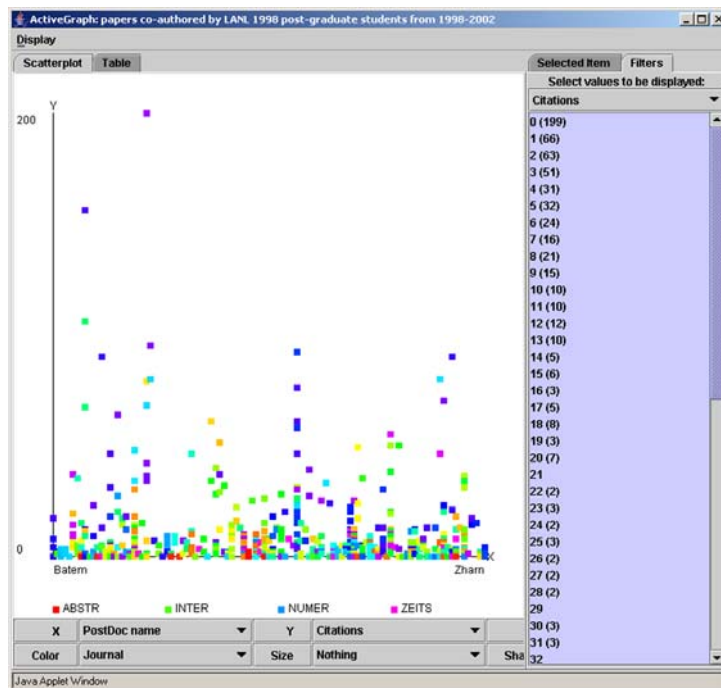


Figure 1. ActiveGraph scatter plot of citation data for papers authored by LANL postdocs

In Figure 1, quantitative information is mapped to one axis and categorical, or nominal, information is mapped to another axis. The variables of interest in a

digital library often consist of categorical or nominal information, such as subject or author. In order to be useful in the context of a digital library, then, a scatter plot interface has to function as a coordinate space for mapping this kind of categorical or nominal information.

The scatter plot in Figure 2 shows the same data set as the scatter plot in Figure 1, but with different mappings: the postdoc's last name is mapped to the X-axis and the first author's last name is mapped to the Y-axis. A diagonal line is formed by data points representing papers for which the postdoc was the first author. In this case, mapping nominal variables to both axes reveals a pattern that would not be evident otherwise and helps users locate and analyze an important subset of the papers.

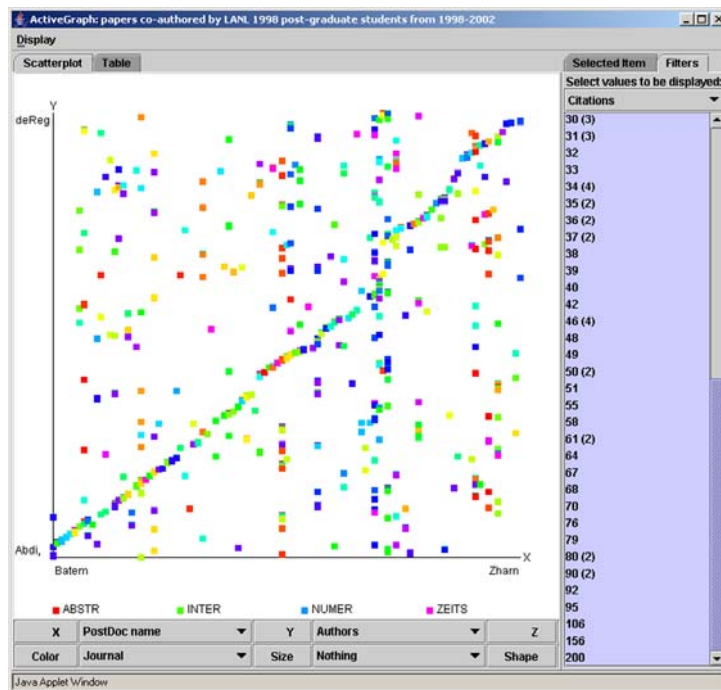


Figure 2. The same data set as Figure 1 with different mappings

The use of a scatter plot as an interface to a digital library is part of a long evolution of the form. One trend evident in the historical record is a progression from the representation of physical properties to the representation of abstract properties, from time and space to category. What might be considered the earliest precursor to the scatter plot dates from the tenth or eleventh century and includes a two-dimensional grid and an axis labelled with iconic representations of the

planets. The graph maps changes in planetary orbits to changes in time [22]. In the eighteenth century, time-series and time-space graphics evolved into more abstract forms, in which temporal or geographical coordinates were replaced by scales showing increments in quantitative information such as temperature and evaporation rate [22]. In the twentieth century, the need to display nominal and ordinal data in the biomedical and social sciences led to further changes and to extensive work in the modelling of qualitative or categorical information [1].

Interactive scatter plots

Another trend evident in the historical record is the difficulty of representing more than two variables. Interactive scatter plots can provide users with the option of controlling how many variables are represented, as well as the manner in which they are represented.

In the mid-1990's Shneiderman and his colleagues developed applications using scatter plots as interactive interfaces and demonstrated their usefulness in a wide range of applications, from finding a home to finding a film [2, 7, 18, 21]. HomeFinder, FilmFinder, and StarDOM allow users to map data attributes to the X- and Y-axes to create a two-dimensional display and Spotfire allows users to map data attributes to the X-, Y-, and Z-axes to create a three-dimensional display. FilmFinder allows users to assign a color to a category, such as "Drama" or "Mystery".

ActiveGraph allows users to map variables or data attributes to six visual attributes: the X-, Y-, and Z-axes, color, size, and shape. Increasing the number of mappings improves the data-pixel ratio, as more information can be displayed per pixel. A pixel of a particular color can provide two pieces of information, for example, by denoting a paper and the subject category of the paper. A group of pixels of a particular size, shape, and color can provide four pieces of information by denoting a paper, the subject category of the paper, whether the paper has been cited, and whether the paper has been read by another user of the collaborative library.

Mapping more than two variables adds complexity to the task of analyzing a scatter plot: three-dimensional scatter plots are more difficult to visually scan than two-dimensional scatter plots. The benefit to users of doing so is that visualizations of more variables provide different views of the data set and can

elicit different insights. Allowing users to control the mappings of data attributes to visual attributes ensures that, if they choose to analyze multiple variables, they can customize and optimize the visual representation of each one.

A related issue in designing interactive scatter plots is how to simplify interaction techniques. Shneiderman et al. have articulated several principles for doing so [2, 7, 18, 21]:

- Visual representation of objects and actions
- Direct manipulation of the visual representation by use of controls
- Tight coupling of the controls with the display
- Immediate update of the display
- Selection by pointing rather than typing

One of the key interface techniques in Shneiderman's scatter plot applications is filters. By applying a filter, users can select a range of data points to display and, in effect, zoom in on the data set. Filter sliders allow users to select a single continuous range of each data attribute to display and facilitate tight coupling of the filters with the display. Eick has demonstrated the use of filter sliders for selecting multiple, non-contiguous ranges [11].

ActiveGraph uses filters, but with some differences. Users can apply filters to display specific data points, a single continuous range of data points, and multiple, non-contiguous ranges of data points. In order to make it easy for users to multiple select individual items and/or multiple non-contiguous ranges, filter lists are provided instead of filter sliders. The filter lists consist of selection list boxes, one for each data attribute. In order for a data point to be displayed, its data attribute values must be selected. When the scatter plot opens, every data attribute value in every list is selected and every data point is displayed. After a filter has been applied, only the data attribute values that have been selected are displayed.

This kind of interface widget does not facilitate as tight a coupling of the filters with the display as filter sliders do, but it does provide users with functionality that is important in the context of digital libraries. Users need to be able to view any combination of data points: all of the papers written by two or more authors whose last names begin with different and non-adjacent letters of the alphabet, for example, or all of the papers published in two or more fields scattered throughout an alphabetical list.

Figure 3 shows the same data set as Figures 1 and 2 after a filter has been applied and two such non-contiguous ranges have been selected. The filter for "Journal super-category" is visible on the right-hand side of the screen and values "COMP" and "MATH" are selected. In the scatter plot on the left-hand side of the screen, papers published in computer science and mathematics journals are displayed.

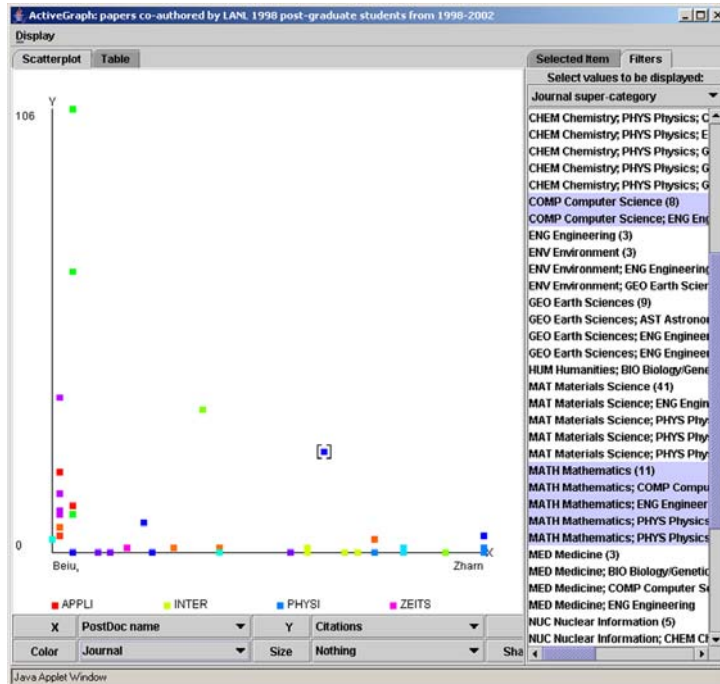


Figure 3. The same data set as Figures 1 and 2 after a filter has been applied

ActiveGraph allows users to manipulate the display of data in another manner, by applying a logarithmic transformation. This kind of transformation uses a mathematical function to redistribute the data points in an exponential distribution more evenly across the scatter plot. Data points that would otherwise be clustered together are spread out. Citation data sets frequently have an exponential distribution: a few papers are cited frequently and most papers are not. Figures 1 and 2 show a scatter plot of such a data set. Figure 4 shows a scatter plot of the same data set after a logarithmic transformation has been applied.

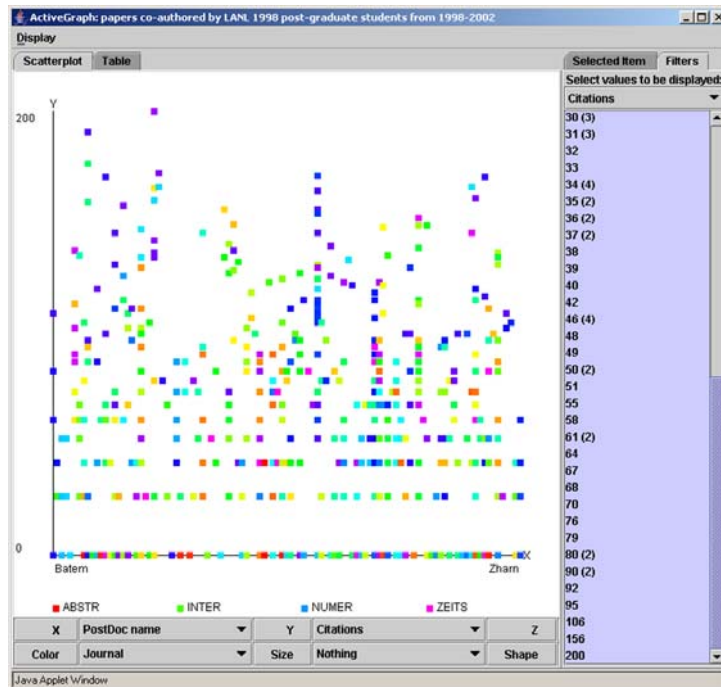


Figure 4. The same data set as Figures 1 and 2 after a logarithmic transformation has been applied

Since citation data sets are important in the context of digital libraries, a logarithmic transformation is an important function to include in a scatter plot visualization designed for digital libraries.

Other data transformations and visualizations may be important in some cases, as well, such as parallel coordinates for displaying citation statistics for the same group of researchers at different points in time. One question is whether these should be included in an interface to a digital library or whether data sets extracted from digital libraries should be visualized using statistics visualization tools such as Diamond [10] and GGobi [13].

ActiveGraph includes several features that are not related to information visualization, per se, but are important to retrieval and analysis in the context of digital libraries. A form fillin window makes it possible for users to edit metadata and add annotations. By doing so, they can preserve the intellectual work involved in sorting through results sets or analyzing the contents of a personal library, and store it with the data set.

ActiveGraph also allows users to add metadata fields. This is important in order to accommodate the needs of different users. A scientist who is interested in specific technical devices may want to add a metadata field for that information to

a personal library. A manager who is interested in tracking collaboration among authors at certain institutions may want to add a metadata field for that information to a data set of papers.

Allowing users to add a metadata field is important in order to accommodate different kinds of resources, also. Web resources do not have the kinds of metadata that traditional library resources have, such as date of publication and citation counts, and the kinds of metadata they do have may change over time. By allowing users to add metadata fields, ActiveGraph can accommodate these changes.

Alternatives to scatter plots

A scatter plot interface to a digital library provides a high data-pixel ratio and a clear view of the patterns in the data: for example, the relative number of citations for a set of papers or the relative number of papers published in various fields. Other kinds of visual interfaces to digital libraries provide other views that may be more appropriate for other tasks.

A table visualization, for example, is used to display digital library objects in the Envision Project interface [14]. The interface allows users to rate whether a paper is useful or not and uses color coding to indicate the ratings. A table interface is more intuitive than a scatter plot, especially for users who are not scientists. Compared to a table interface, however, a scatter plot interface has a higher data-pixel ratio and is more efficient for displaying large numbers of digital library objects.

Another kind of visual interface, a link-node visualization that reveals relationships between data points, is used in the co-citation map and the author co-citation map developed by Lin and his colleagues [6, 15]. The same kind of visualization is used in the co-author browser created by Liu and his colleagues [16]. Both of these visualizations use the appropriate visual interface for displaying invariate relationships between and among data points, such as co-authorship. Scatter plots, in contrast, allow users to sort the data points along a number of dimensions. They can provide an overview of a range of digital library objects, including web resources and databases, which have no author or citation data.

Skins and overlays are used effectively in the context of digital libraries by Boyack and his colleagues [3, 5]. Like many other visual interfaces that provide a level of visual sophistication beyond that provided by the scatter plot, skins and overlays involve some learning on the part of the user. User feedback regarding the digital library at LANL suggests that many users do not want to spend time learning a new interface and user feedback regarding ActiveGraph suggests that they do not need to when they are viewing a scatter plot. Therefore, work on ActiveGraph has focused on preserving what is familiar about scatter plots and simplifying what may not be familiar about the controls.

Visual attributes

When ActiveGraph draws a scatter plot, single data points are represented by a symbol called a "glyph". Within the universe of scatter plots, many kinds of visual attributes can be applied to glyphs: at least 18, based on an analysis of visualization tools. Each of these visible attributes can be used to denote a meaningful difference between data points or to improve the clarity of the visual display. In ActiveGraph, ten of the visual attributes are controlled by the user and two are controlled by the program. The other six are not used in the current design.

- Controlled by the user: the X-, Y-, and Z-axes, foreground color, background color, size, shape, halo, density, motion
- Controlled by the program: brightness, proximity
- Not used in the current design: transparency, shadow, pulse, lines, texture, skins and other overlays

Controls for six of the visual attributes – the X-, Y-, and Z-axes, color, size, and shape – are grouped together below the scatter plot and controls for density (filters) are on the right-hand side of the interface. Halo and motion are controlled by the mouse.

ActiveGraph includes options for controlling visual attributes so that users can customize their view of the scatter plot. The scatter plot first appears in a simple form, however, as a set of square, white glyphs on a black background, in two dimensions. Those users who want to customize their view of the data set can use the initial display as a base for exploring the alternatives. Those users who do not want to spend time customizing their view of the data set can view and

analyze the data in the simple form. In addition, users who are color-blind and cannot detect differences between some colors may need to view the scatter plot in black and white, and users who find it difficult to understand three-dimensional scatter plots may need to view the scatter plot in two dimensions.

ActiveGraph determines the initial visual representation of data points by scaling visual attribute values to a 0-1 range. The program needs a way to describe all the possible values of each visual attribute and using the same scale for every visual attribute makes it easy to convert from one to another. Thus a value of 0.5 represents a position halfway between the ends of the X-axis, or halfway between red and purple, or halfway between the largest and smallest sizes. This is an idealized model, since no visual attribute has an infinite number of distinctions to match the infinite set of numbers between 0 and 1. In practice, for four shapes, any value ≥ 0 and < 0.25 is represented as the first shape, any value ≥ 0.25 and < 0.5 is represented as the second shape, and so on.

Position on the X- and Y-axes: ActiveGraph assigns a position on the X- and Y-axes after scaling data attribute values to a 0-1 range. This is straightforward for numerical information, such as citation counts, but not for nominal variables, such as alphabetically sorted text. Nominal variables have no meaningful order and the sense of order conveyed by position may suggest that the arbitrary order is, in fact, meaningful. Nonetheless, ActiveGraph allows any data attribute, whether it is a nominal variable or not, to be mapped to any visual attribute, including the X- and Y-axes, because mapping nominal variables to position sometimes reveals important information to the user. Figure 2 provides an example.

Foreground and background color: While position can be rendered with precision on a computer display, color can be rendered with even more precision, technically. Color provides more resolution than position on a computer display with 24-bit color and a 1280 x 1024 resolution. 24-bit color can display more than 16 million shades. If only hue is used (and saturation and brightness are both fixed at 1.0), this is reduced to 1536 possible values. This is still more than the maximum possible number of distinctions that can be made on the horizontal axis, which is 1280 minus the pixels used by the operating system for the application window frame, legends, and blank space. Given the limits of human perception,

however, very few people can easily distinguish among 1280 positions or 1536 colors. In addition, selecting a color mapping that does not introduce perceptual distortion is a non-trivial task as Rogowitz and Treinish have demonstrated [19, 20].

Position on the Z-axis, brightness, and motion: Three-dimensional visualizations create their own set of perceptual problems: specifically, how to create the impression of depth without obscuring glyphs and labels. ActiveGraph uses two depth cues: brightness variation and motion.

On a black background, glyphs become darker as they recede into the "background" of the display and brighter as they approach the "foreground" of the display. On a white background, the opposite occurs. One effect is that the color scale used for mapping data points in three-dimensional ActiveGraph displays must be limited to hue or saturation variations.

Darkening and brightening glyphs is not effective without an additional depth cue, such as rotation about the center of the display. Therefore ActiveGraph allows users to move the three-dimensional display by dragging it with their mouse. The display rotates continually with the direction and speed of the mouse. Combined with variations in brightness, this motion causes the depth of the glyphs to become obvious to users.

Size, shape, and proximity: Size and shape are more limited than position and color. Size has the same upper limit as position and the additional limit that, in two-dimensional displays, a glyph cannot be so large that it obscures another glyph. In three-dimensional displays, glyphs in the foreground must be allowed to obscure glyphs in the background in order to maintain the illusion of depth, but obscured glyphs can be revealed by rotating the scatter plot.

The size of glyphs is partly dependent on the number of data points in the display. In two-dimensional displays, a good rule-of-thumb is that if two glyphs overlap each other's central point, then the largest allowable size is too large or the glyphs need to be grouped. Another rule of thumb is that if only one of a pair of glyphs overlaps the other's center, then the one that is being overlapped should be drawn after (and therefore over) the other, larger glyph. This ensures that one glyph will never obscure more than half of another, although there is still no

guarantee that a glyph will not be completely obscured by multiple smaller glyphs.

On occasion, two glyphs overlap because they are co-located. For example, if an author has published two or more papers in the same year and author name is mapped to one axis and year to the other, the glyphs representing the papers will be located in the same place on the scatter plot. In this case, ActiveGraph displays a set of text labels when the user's cursor moves over the data points.

Shape interacts with size. At a size of one pixel, there is only one shape. If the smallest glyph is limited to a 3 x 3 block of 9 pixels, the display can support a set of four or five easily distinguished shapes. The shapes should be distinct enough to be distinguishable even if a significant fraction (50%+) of their area is obscured.

Density: Another aspect of visualization is density. When a data set is very large, a scatter plot ceases to be useful for revealing patterns because the display becomes too cluttered with glyphs. ActiveGraph allows users to select ranges of data attribute values to display using the filter widget. ActiveGraph's filter widget allows users to select any combination of data points to display: individual data points, single ranges of data points, or multiple non-contiguous ranges of data points. This is comparable to using a zoom function to study one part of the display and causes no loss of detail, but may obscure large-scale patterns that are not visible at a local level.

The opposite method is to reduce the number of glyphs on the display by merging groups of similar data points into single glyphs. ActiveGraph allows users to select "Conglomeration" and, in effect, zoom out. This method does cause a loss of detail, but may reduce visual clutter sufficiently to reveal large-scale patterns.

The density of clusters in a scatter plot with an exponential distribution can be manipulated by applying a logarithmic transformation to redistribute the data more evenly.

Halo: Users can select a specific data point in the scatter plot by clicking on it. A halo consisting of two brackets appears around the data point and the data

attribute values or metadata for that data point appear in the control panel on the right-hand side of the display.

Architecture

ActiveGraph is written in Java. Java was chosen because of its well-developed interface toolkit (Swing), its cross-platform portability, and the widespread support for Java applets by web browsers. Java terminology is used in this section of the paper, including class (a type of object or module) and method (a function or subroutine associated with a class). Class names are always capitalized, as this is a Java convention.

A table of data can easily be transformed into a scatter plot. Each row of the table becomes a data point. Each column of the table becomes a data attribute shared by all of the data points, such as title or author. Each table cell's content becomes a value for a particular combination of data point and data attribute: for example, a specific title or author. The various visual attributes of the scatter plot, such as position, color, size, and shape, are mapped to data attributes and values for one are converted to values for the other. The values for all the data attributes of a data point are converted to values for visual attributes and combined to form a glyph on the scatter plot.

ActiveGraph automates this transformation of a table of data into a scatter plot using three main classes: the Data class, which stores, sorts, and transforms the raw data; the Interface class, which creates interface widgets for every user task and handles their calls to the other classes; and the Visualization class, which combines visual attribute values to draw the scatter plot.

Data class

The Data class handles the storage and manipulation of data, including sorting, grouping, scaling the raw data to values between 0 and 1, and applying mathematical functions such as logarithmic transformations.

There are several types of data: single lines of text, blocks of text, numbers in small ranges, numbers in large or infinite ranges, and data with an ordered, finite set of possible values. This classification is based on the necessity for a different set of interface widgets for each type of data.

Data types are also handled differently during sorting and transformation. Numbers, whether selected from a small range or a large range, are sorted numerically. Single lines of text are sorted alphabetically. Blocks of text can also be sorted alphabetically, but this is of little use in practice. Sorting by the number of words in each block is more useful. This kind of sort will quickly reveal which objects in a collaborative library have extended annotations and which have brief or non-existent annotations, which in turn reflects the degree of interest in each object.

In the case of ordered lists, the sort order is the same as the list order. For example, the days of the week are sorted beginning with "Sunday" and ending with "Saturday".

The Data class also handles grouping. This allows users to apply filters and zoom in, or to select "Conglomeration" and zoom out.

Interface class

The Interface class creates interface widgets for every necessary task. The tasks include: displaying a selection of data points; adding, deleting, and editing data points; adding, deleting, and editing data attributes; and selecting ranges of data to display.

The Interface class is, in fact, a collection of smaller classes, one for each task. Each class creates the appropriate widget for the given type of data and interprets the widgets' outputs as the appropriate commands to the Data and Visualization classes.

Table 1 shows the widgets used for displaying and editing five types of data: one line of text, several lines of text, a small range of numbers, a large or infinite range of numbers, and an ordered list.


		Tasks	
		Display	Edit
Type of Data	One line of text	Title <input type="text" value="Information Visualization"/>	Title <input type="text" value="Information Visualization"/>
	Several lines of text	Summary This is the introduction to the book "Information Visualization: Using Vision to Think", edited by Card, Mackinlay, and Shneiderman. It gives a general overview of the history of the field and the cognitive-perceptual issues involved in using visualizations.	Summary This is the introduction to the book "Information Visualization: Using Vision to Think", edited by Card, Mackinlay, and Shneiderman. It gives a general overview of the history of the field and the cognitive-perceptual issues involved in using visualizations.
	Small range of numbers	Year Written = 1999	Year Written = 1999 
	Large or infinite range of numbers	Year Written = 1999	Year Written = 1999 <input type="text" value="1999"/>
	Ordered list	Type <input type="text" value="Book Chapter"/>	Type <input type="text" value="Technical Report"/>

Table 1. Table of widgets

Visualization class

The Visualization class takes input from the Data class and displays a graph in the form of a scatter plot with legends, determines which glyphs overlap, decides the order in which they should be drawn, and determines which visual attribute, if any, should be used to display each data attribute. Users can change the mapping of visual attributes to data attributes at any time. For example, color can be mapped to author and size can be mapped to number of citations, or vice versa.

The architecture described above preserves the uniqueness of statistical outliers and ensures that their visual attributes will reflect their uniqueness. For example, in the scatter plot of citation statistics shown in Figure 1, individual papers with exceptionally high numbers of citations are displayed in such a way that they attract more attention than individual papers with low numbers of citations.

In the case of alphabetically sorted text, the distance between values is not informative and highlighting values that are exceptional – lines starting with "Z", for example – is likely to be distracting rather than useful to users. Therefore, for alphabetically sorted text, the minimum is 1, the maximum is the number of unique strings, and the value of a given string is its rank in the sorted list of unique strings. The final visual attribute values are evenly distributed across the possible range and thus evenly distributed across the display. For example, if author names are mapped to the X-axis, the glyphs will be evenly distributed along the axis, even if half the names start with the letter "A" while the other half start with the letter "S". The ordered list type also uses rank rather than actual value, as it is meaningless to define the distance between values such as "Sunday" and "Monday".

Every time a user adds, deletes, or edits data, the sorted lists of unique values for each field and for every data point, displayed or not, must be updated. Also, the list of data points being displayed and the maximum and minimum values being displayed must be recomputed. Then ActiveGraph adjusts the scales of the scatter plot so that the available display space is used most efficiently.

Applications

ActiveGraph is the full-featured version of the digital library visualization tool described in this paper. In order to accommodate different user scenarios, specific applications and applets have been developed using the same code base, but different sets of features.

LibGraph

LibGraph allows users to view, analyze, and annotate the contents of a collaborative library. It facilitates collaboration and personalization by enabling researchers who are working together to share the results of their reading and

analysis. In the particular library discussed here, the contents include books, book chapters, journals, and papers, but any digital library object can be included in the collection.

The initial design of LibGraph was motivated by interest in collaboration and personalization, as well as information visualization. An existing collaboration tool at LANL did not provide functionality for modifying metadata or for including collaborative rankings and critiques on the content of the library. Based on these facts, LibGraph was designed to support the following collaborative user scenario:

1. A member of the group locates a book, book chapter, journal, or paper of interest and adds it to the collection.
2. A member of the group is assigned to read and review the work.
3. This member of the group reads it and, using the editing tools in LibGraph, writes a summary of it, categorizes it, and gives it an importance ranking to indicate whether other members of the group should read it.
4. This information becomes part of the metadata for that object.
5. The other members of the group read the metadata and, if necessary, edit it.
6. The results of reading and analyzing the work are preserved in LibGraph, along with the object.

In Figure 5, the main part of the display shows a scatter plot of data in two dimensions. The view-manipulation controls beneath the plot allow users to assign different metadata fields to the X-, Y-, and Z-axes, and to the other visible attributes of color, size, and shape.

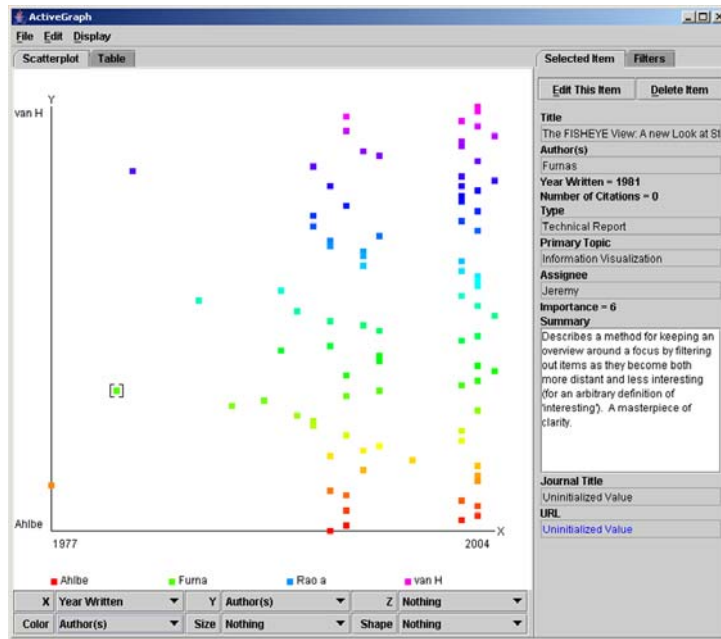


Figure 5. LibGraph visualization of the contents of a collaborative library

The data set consists of papers in the field of information visualization. Year of publication is mapped to the X-axis and author is mapped to the Y-axis and to color. It immediately becomes apparent that there is a gap in the collection. At the time this screen shot was taken, the collaborative library included content from two sources: a collection of important papers in the field which was published in 1999 and contains papers written from 1977 to 1999 (*Readings in Information Visualization: Using Vision to Think*) and papers from a journal that was first published in 2002 (*Information Visualization*). Data points appear for the years 1977-1999 and 2002-2004, but not for the years 2000 and 2001. The scatter plot makes this fact obvious.

In addition to mapping data attributes to visual attributes, users can edit the metadata. When they click on the scatter plot, the point nearest the cursor is selected and indicated by brackets. Detailed information about the selected point appears in the control panel on the right-hand side of the screen. The control panel has two buttons: "Delete Item" and "Edit This Item". When users click on "Edit This Item" a dialog box appears with controls for changing the metadata associated with the selected point. Once users close the dialog box, any changes they made are immediately reflected on the scatter plot.

In addition to editing metadata, users can change the structure of the data by adding, editing, or deleting data fields. When they select "Add a Field" from the edit menu, a dialog box appears that allows them to add a field of a specific name and type: text area, text field, URL field, list field, or number field. They can also configure any additional variables that the field might need.

Figure 6 shows the dialog box for adding a number field. Note that the variables for this particular field include lower bound and upper bound. These values are necessary in order to define the end points of the sliders that will appear in the dialog box for editing metadata after the new data field has been added.

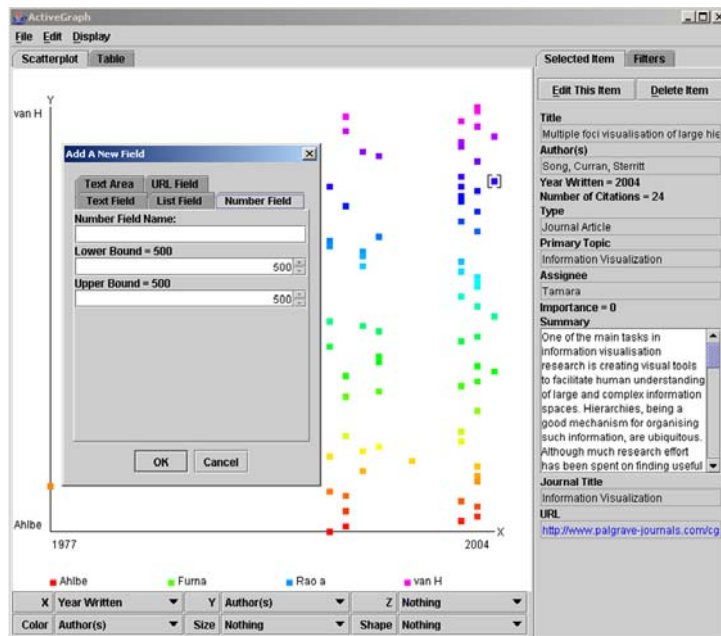


Figure 6. LibGraph dialog box for adding a field

Functionality for allowing users to modify fields was added because it became clear that predicting all the fields that might be useful for various groups of researchers was not feasible.

CiteGraph

CiteGraph allows users to view and analyze citation statistics for papers written by researchers at LANL. Citation counts are an important metric in internal and

external reviews and managers involved in the review process need to be able to analyze them as they would any other data set.

Based on these facts, CiteGraph was designed to support a complex user scenario:

1. A citation count is requested.
2. A script extracts the citation counts from a digital library application that contains citation statistics for journal papers in ISI SciSearch.
3. CiteGraph reads the extracted file and renders the data set in a read-only scatter plot visualization.
4. CiteGraph facilitates analysis of the data by the target end user: the manager who initially requested the citation count.

Figure 7 shows a CiteGraph scatter plot visualization of citation counts for LANL authors for papers published in 2002 and cited in other papers indexed by ISI SciSearch through August 2003. The main part of the display shows a scatter plot of data in two dimensions. As in the case of LibGraph, the view-manipulation controls beneath the plot allow users to assign different metadata fields to the X-, Y-, and Z-axes, and to the other visible attributes of color, size, and shape. The color coding here is by subject, in alphabetical order. In the color version of this figure, blues and purples represent articles in physics, which are particularly important at LANL.

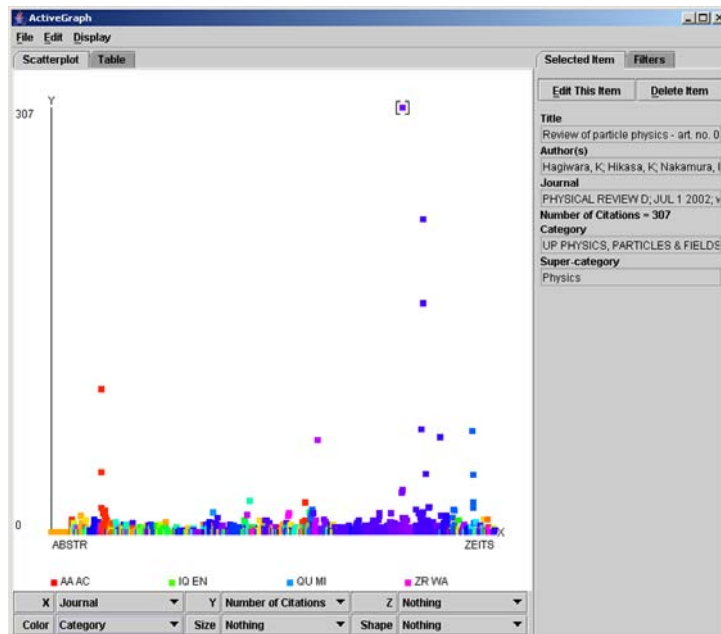


Figure 7. CiteGraph visualization of citation statistics for LANL authors

This data set demonstrates an exponential distribution: many data points are clustered together while a few are very far apart, reflecting the fact that a few papers have been cited frequently. The data set can be reconfigured, using a logarithmic transformation, so that it is easier to see the data points that are clustered together. This feature was considered critical by four senior scientists who viewed and critiqued the design.

Figure 8 shows a CiteGraph scatter plot view of the same data set as Figure 7 after a logarithmic transformation has been applied. All the data points are still visible, but they are more evenly distributed in the available space, allowing users to see the scatter in previously dense areas more clearly. In this case, it becomes obvious that more papers have received 0 citations than have received 1 citation, 2 citations, or any other single number of citations.

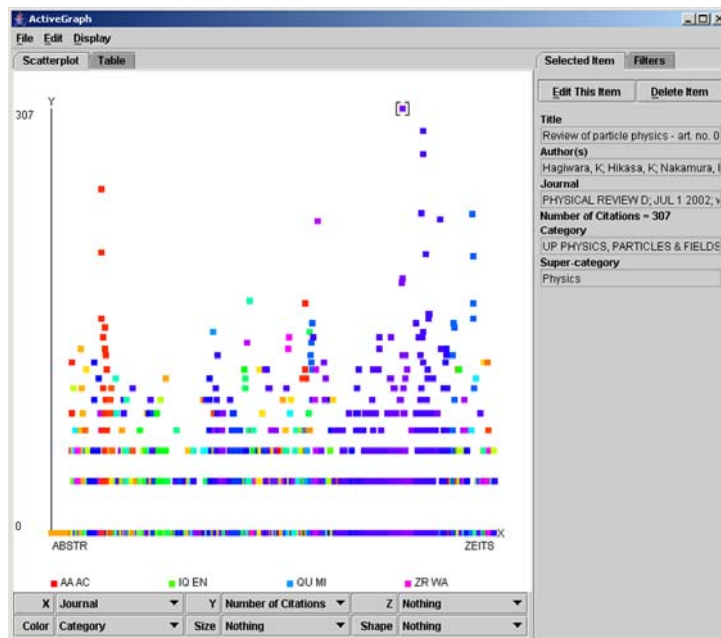


Figure 8. CiteGraph visualization of the same data set as Figure 7 after a logarithmic transformation has been applied

It is also possible to change the visualization of the data set by applying filters in the control panel on the right-hand side of the screen. Users can select or deselect individual items or groups of items on the filter list. To select or deselect

individual items, they hold the Ctrl key while clicking on the items in the list. To select a range of items in the list, they click on the first item in the range then hold the Shift key and click on the last item in the range.

Empirical work on CiteGraph with target end users has revealed interesting behavior. The combination of the scatter plot and the controls seems to escalate discussions about the data set in a cyclical manner. When users view the initial scatter plot they quickly perceive the obvious patterns. They then ask questions that require the use of filters and logarithmic transformations. They analyze the new views and try to detect other patterns and, in turn, new patterns give rise to new questions. By making it possible to see the simple patterns, then experiment with other configurations of the data set, CiteGraph facilitates in-depth analysis and high-level discussion.

Future work

In addition to LibGraph and CiteGraph, two other ActiveGraph applications are in development.

PubGraph allows users to view and analyze publication data for which there are no citation counts. This increases the number of data sets that can be used with an ActiveGraph scatter plot since many data sets do not include citation counts.

SearchGraph allows users to view and analyze search results from many databases in real-time. As a first step in this direction, pre-selected data sets from these databases will be made available to users. After issues in scalability and usability have been resolved, the ActiveGraph interface will be provided as an optional interface for these databases.

Conclusions

ActiveGraph makes it possible for users to view objects in a digital library in the form of a scatter plot. This provides them with an efficient view of the objects and makes it possible for them to analyze the data set as they would any other data set. For scientists, this view of a digital library is intuitive and intriguing.

View-manipulation controls make it possible to manipulate the display in order to explore patterns in the data. Editing controls make it possible to edit metadata and to add new metadata fields. This functionality provides a means of

preserving the intellectual property created in the process of viewing and analyzing content and improves the usability of the digital library, as a whole.

Acknowledgements: The authors would like to thank the reviewers for their comments on an earlier version of this paper, Mark Martinez whose work on citation statistics made CiteGraph possible, and Joe Liberty and Kent Hettinga whose contributions to code reviews improved the quality of the work.

References

1. Agresti A (2002) *Categorical data analysis*. John Wiley and Sons, Inc., Hoboken, NJ
2. Ahlberg C, Shneiderman B (1994) Visual information seeking: tight coupling of dynamic query filters with starfield displays. *Proceedings CHI '94* 313-317, 479-480
3. Börner K, Chen CM, Boyack KW (2003) Visualizing knowledge domains. *Annual review of information science and technology* 37: 179-255
4. Börner K, Feng Y, McMahon T (2002) Collaborative visual interfaces to digital libraries. *Second ACM+IEEE Joint Conference on Digital Libraries* 279-280
5. Boyack KW, Wylie BN, Davidson GS (2002) Domain visualization using VxInsight for science and technology management. *Journal of the American Society for Information Science and Technology* 53: 764-774
6. Buzydlowski J, White HD, Lin X (2001) Co-cited author analysis as an interface for digital libraries. *JCDL Workshop on Visual Interfaces to Digital Libraries*, Roanoke, VA
7. Card SK, Mackinlay JD, Shneiderman B (1999) *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers, San Francisco, CA
8. Cleveland WS (1993) *Visualizing data*. Hobart Press, Summit, NJ
9. Collins MLM (2002) *Interface macrostructures: content representations for interactive information spaces*. Doctoral dissertation, Columbia University, New York, NY
10. Diamond, a statistics visualization tool. URL: <http://researchweb.watson.ibm.com/visualanalysis/tools.html>
11. Eick SG (1994) Data visualization sliders. *Proceedings UIST '94* 119-120
12. Friendly M, Denis DJ (2004) Milestones in the history of thematic cartography, statistical graphics, and data visualization: an illustrated chronology of innovations. URL: <http://www.math.yorku.ca/SCS/Gallery/milestone/sec2.html>
13. GGobi data visualization system (2004) URL: <http://www.ggobi.org>
14. Heath LS, Hix D, Nowell LT, Wake WC, Averbach GA, Labow E, Guyer SA, Brueni DJ, France RK, Dalal K, Fox EA (1995) *Envision: a user-centered database of computer science literature*. *Communications of the ACM* 38: 52-53
15. Lin X, White HD, Buzydlowski J (2001) AuthorLink: instant author co-citation mapping for online searching. *Proceedings of National Online* 233 - 241
16. Liu X, Bollen J, Nelson ML, van de Sompel H, Hussell J, Luce R, Marks L (2004) Tools for visualizing co-authorship graph. Poster session, *Fourth ACM+IEEE Joint Conference on Digital Libraries*

17. McMahon TM, Börner K (2002) A Study on the effects of web page panel size and layout density on information access in a 3-dimensional collaborative virtual world. LA-UR 03-0339
18. North C, Shneiderman B (1999) Snap-together visualization: coordinating multiple views to explore information. URL: <ftp://ftp.cs.umd.edu/pub/hcil/Reports-Abstracts-Bibliography/99-10html/99-10.pdf>
19. Rogowitz BE, Treinish LA (1998) Data visualization: the end of the rainbow. IEEE Spectrum 35:52-59
20. Rogowitz BE, Treinish LA (1994) Using perceptual rules in interactive visualization. Proceedings of SPIE 2179:287-295
21. Shneiderman B (1998) Designing the user interface: strategies for effective human-computer interaction. 3rd ed. Addison-Wesley, Reading, MA
22. Tufte ER (1983) The visual display of quantitative information. Graphics Press, Cheshire, CT
23. Tufte ER (1990) Envisioning information. Graphics Press, Cheshire, CT