# U-Finder: An Interactive Tool for Data Exploration and Decision-Making

Jeanette Bautista and Micheline Manske
Department of Computer Science
University of British Columbia
2366 Main Mall, Vancouver, B.C. Canada V6T 1Z4
{bautista, manske}@cs.ubc.ca

## Introduction

Choosing a university or college to attend can be a daunting task. Each school has different qualities that must be taken into account and weighed against one another when making a decision. Location, cost of tuition, prestige, and size are just some of the variables that potential students take into account. The selection problem is two-fold. Firstly, information about colleges and universities is not found in a central location. In most cases students must seek this information for themselves, either by searching school web pages, reading brochures, speaking to current students, or visiting the schools. Secondly, students need to put all of this information together in a meaningful way that allows them to evaluate the pros and cons of each school and balance the trade-offs between them. Unfortunately, people are not very good at judging tradeoffs, particularly when the number of options to consider increases.

The goal of this project was to design a system, UFinder, for selecting a college or university. UFinder contains a comprehensive data set of over 1300 universities and colleges in the United States with information about schools ranging from admission numbers to quality of education. It allows the user to reduce the data set by filtering out schools that are not of interest, compare several selected schools against one another with attributes adjusted to reflect user preferences, and obtain details about any school on demand. Our system is targeted a high school students, their parents, high school guidance counsellors, and anyone else for whom ranking schools is an important task.

The organization of this report begins with a presentation of related work. We then explain the data set being used in our implementation. We propose a solution and provide a scenario of use. Next we give high-level details of implementation. We then discuss our evaluation and strengths and weaknesses of our solution. We conclude with lessons we learned and suggestions for future work.

## Related Work

Many systems have been developed to aid in exploring large data sets with many attributes. With VisDB (Keim and Kriegal, 1994), users submit queries by way of range sliders for each attribute. The results of the query are displayed on the screen with a pixel for each item. The pixels are colored according to the relevance of the point to the query. This system is good for exploring extremely large databases and looking for clusters or trends in the data. However, it is less appropriate for selecting a small number of data items, such as a few schools, as our problem requires.

Other researchers have investigated the more specific problem of choosing a small subset of items from among many based on their attributes. Both HomeFinder (Williamson and Shneiderman, 1992) and FilmFinder (Ahlberg and Shneiderman, 1994) use dynamic query filtering – buttons and sliders which alter the parameters of the query in order to narrow down the number of items to be considered. Homes (films) that fall within the desired range for each attribute are shown on a graphical display. These visualizations allow users to progressively modify the query as they explore the data set. Users can select individual data points for closer inspection known as details on demand. The limitation of these two systems is that they do not allow users to compare data items side-by side. Apart from looking at the individual details of items that meet the query criteria, the user is not supported in decision-making between items that fall in the acceptable range.

AttributeExplorer (Tweedie et al.,1994) uses linked highlighting to support the selection of data objects. For the example of buying a house, attributes are arranged in columns of house icons, where each column represents one attribute with a house icon for each value of that attribute. Selecting houses having a range of values for one attribute highlights the same houses in the other attribute columns. This approach is limited, however, in both the number of attributes and the number of data points it can display. As well, it does not allow for side-by-side comparisons of data objects.

ValueCharts (Carenini and Lloyd, 2004) approaches the problem of data selection from a different perspective. Using ValueCharts, users can weight attributes for each data element according to their preferences. Using the example of houses, the value for each attribute of each house is shown as a bar beside the house name. A second window shows the cumulative total of all the attributes (normalized to the range 0-1) for each house. Using direct manipulation, users can adjust the relative space allocated to each attribute, thereby specifying their preferences. While ValueCharts allows for side-by side comparison, it is limited in the number of schools or variables that can be considered at any one time.

For our project we wished to combine the positive features of the HomeFinder/FilmFinder and ValueCharts approaches, allowing users to dynamically filter schools using sliders and buttons similar to those in HomeFinder/FilmFinder, then view the filtered set of schools using a modified version of ValueCharts allowing users to compare schools and rank them using their preferences.

## Data Set

For this project we used the USNEWS data set of 1306 schools in the United States, although in theory any data set would be appropriate. The data set is taken from the U.S. News & World Report's Guide to America's Best Colleges, for the 1993-1994 academic year. It is available from the StatLib website at http://lib.stat.cmu.deu/datasets/colleges

From the 33 attributes for each school, we selected the 21 most relevant to potential students. These are presented in Table 1.

| **Attibutes in the USNEWS data set** | Average Math SAT score |
|---|---|
| College name | Average Verbal SAT score |
| Location (State) | Average ACT score |
| Public or private status | % students in top 25% of High Sch. class |
| Number of full-time students | Graduation Rate |
| Number of part-time students | Tuition |
| Number of applications received | Cost of books |
| Number of applications accepted | Additional fees |
| Number of students enrolled | Room costs |
| Student to Faculty ratio | Board costs |
| Expenditure per student | Average personal spending costs. |

**Table 1: Attributes from data set**

Examination of Table 1 reveals that there is an inherent hierarchy in the attributes. We use this to reduce the perception of an overwhelming number of variables. This is in line with Shneiderman's Information-Seeking Mantra: Overview first, zoom and filter, then details on demand (Shneiderman, 1996). In order to exploit the hierarchical nature of the data, we introduce intermediate attributes which represent an aggregate of several lower level attributes as shown in Figure 1. This allows us to selectively hide lower level attributes, helping the user to chunk information. The student needs to only consider four main attributes - caliber of students, cost per year, environment, and acceptance - when making her decision, but can drill down when necessary.

In the case of the cost tree, the intermediate attributes (school, room & board, living, cost/year) were calculated by summing the attributes immediately below in the tree. In all other cases the intermediate attribute represents an average of the attributes below (normalized to the range 0-1).

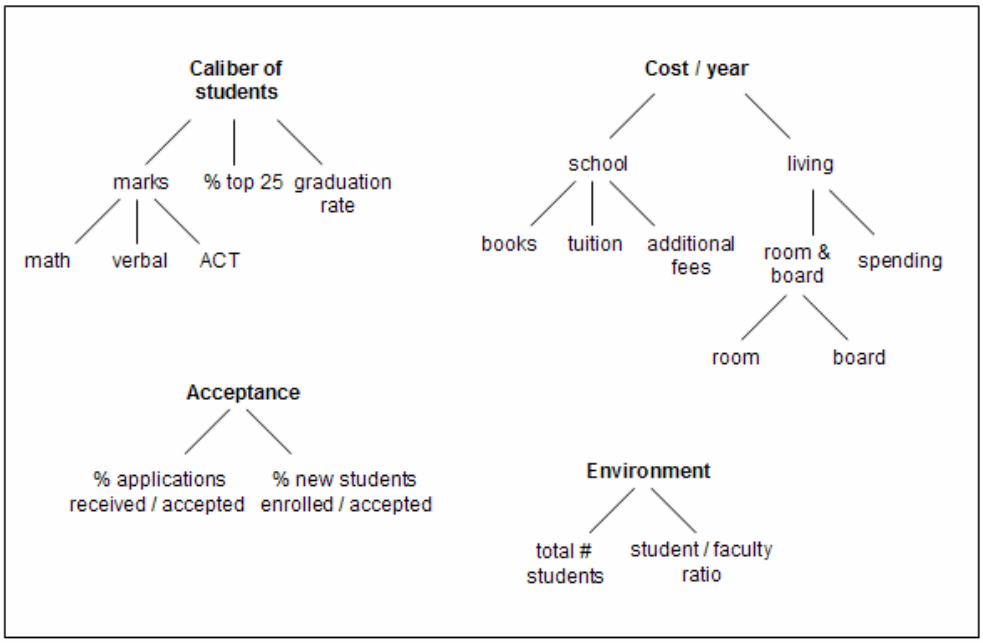



**Figure 1: Attributes arranged hierarchically**

One problem that we encountered with the data set was that many values were missing. For example, a school may have values for average Math and Verbal SAT scores, but is missing the value for ACT score. There are several different possible solutions to this problem such as ignoring the missing values or setting the missing values to the average value for that attribute. In our case, we chose to ignore the missing values when calculating the intermediate attributes. In the above example this would mean that the Marks attribute would be taken as the average of the two SAT scores, ignoring the ACT. We felt that this was more appropriate than inserting a possibly non-representative value in for ACT score which could skew the average. In the case of the cost tree, this was especially relevant as prices for Room and Board costs varied immensely, and it appeared that in many cases the Room value encompassed board as well, and Board was left blank. Substituting an average value in for Board would inflate the numbers and make the Room and Board costs unrealistically high. A different solution was used to overcome the problem of missing values when displaying ValueCharts, as will be discussed in a later section.

## Proposed Solution & Scenario of use

Our solution to the problem of selecting schools is two-fold. We first allow users to reduce the number of schools to consider by filtering using dynamic sliders. When the users have obtained an acceptable set of schools, they can explore the data and rank schools relative to one another using a ValueChart. A screenshot of the system is shown in Figure 2. Throughout both interactions we use the hierarchical nature of the data to show users only the level of detail that they wish to examine.

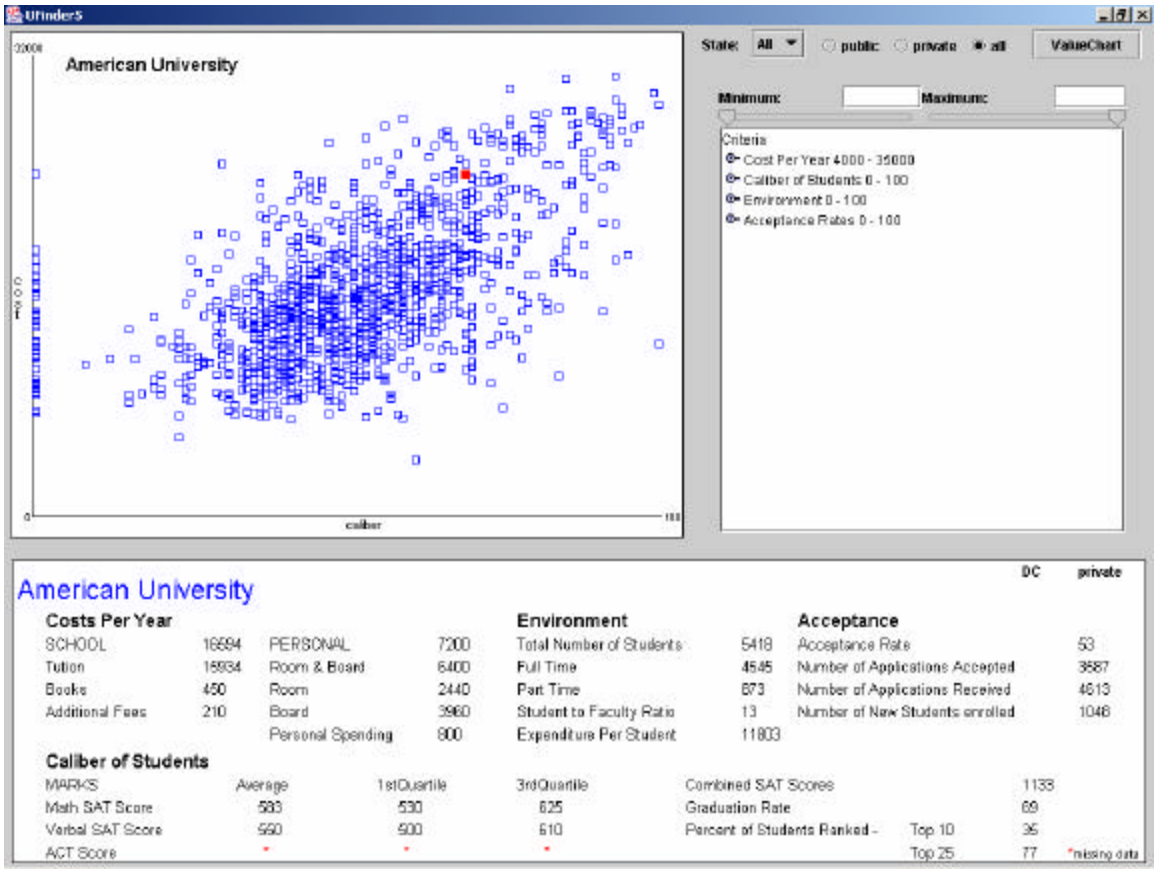



**Figure 2: Screenshot of UFinder**

An example scenario of use would go as follows. Susie wishes to attend either a University or college in the United States and uses UFinder to help make her decision. First she wants to narrow down the available schools to consider, so she looks to the right pane (Figure 3). She changes the default state All to California, the only state she is considering, and selects Private as the type of school she would like. This immediately reduces her choices, which are represented as squares on a scatter plot to the left (Figure 4).

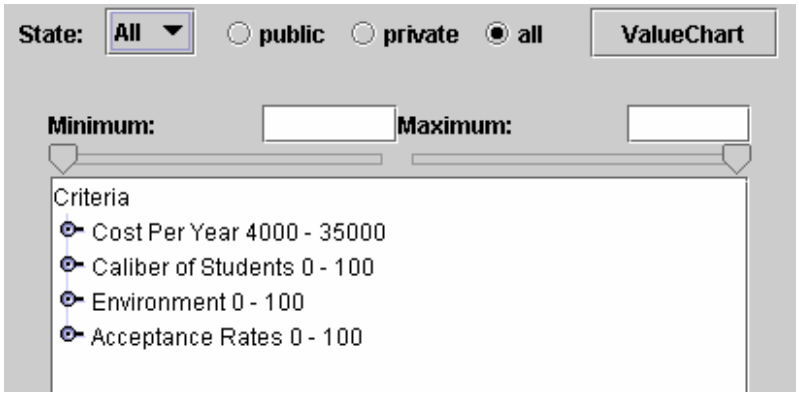



**Figure 3: Filtering control panel**

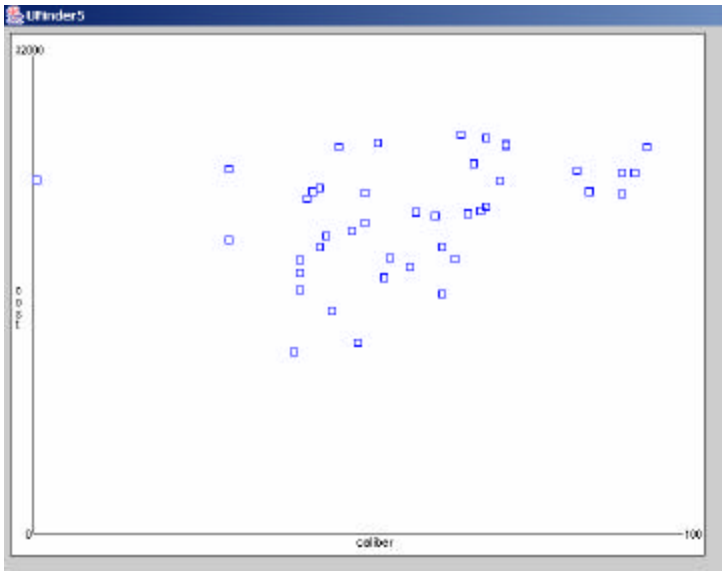



**Figure 4: Scatterplot of schools in the current range.**

Susie then turns her attention the rest of the control panel. There are still too many schools to consider, so she begins setting ranges for acceptable values of some of the attributes using the slider (Figure 5). She clicks on Cost Per Year and changes the slider to represent the range $4,000 to $25,405 (Figure 5a). Susie drills down further to the level of School Costs and Living Costs (Figure 5b). She double-clicks School to reveal the more specific attributes Books, Tuition, and Additional fees (Figure 5c). Deciding that this is too specific for her needs, she goes back up and uses the slider to adjust the School Costs range to be $0 to $16012 (Figure 5d). She notes satisfactorily that the Cost Per Year range has been adjusted accordingly. She does the same for some of the other attributes, setting them to satisfactory ranges and leaving attributes that do not concern her at their default setting - the entire range.

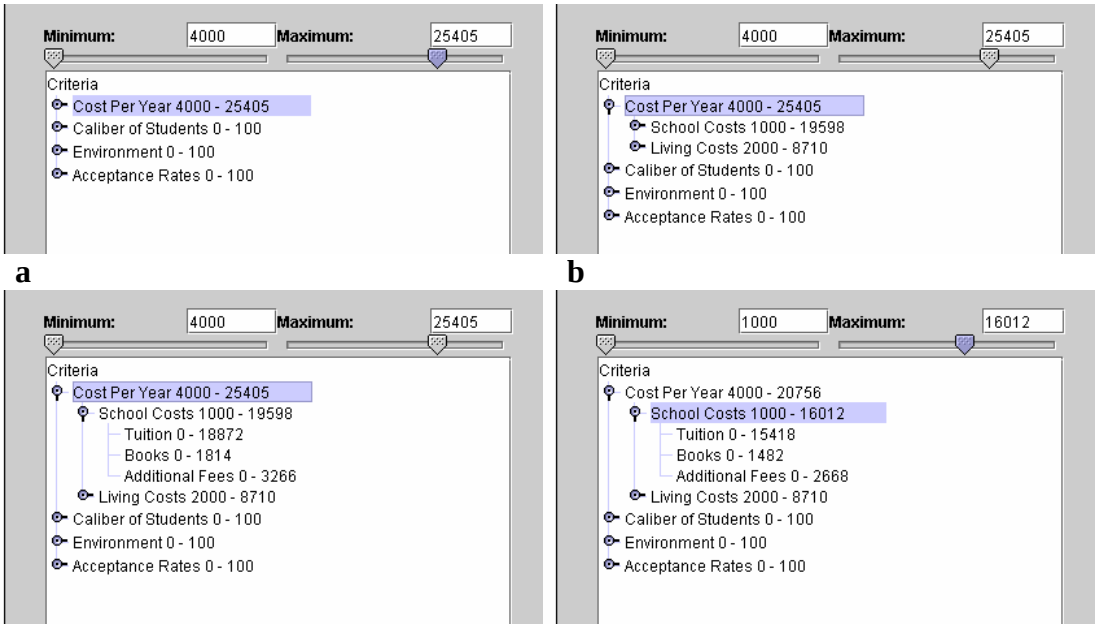
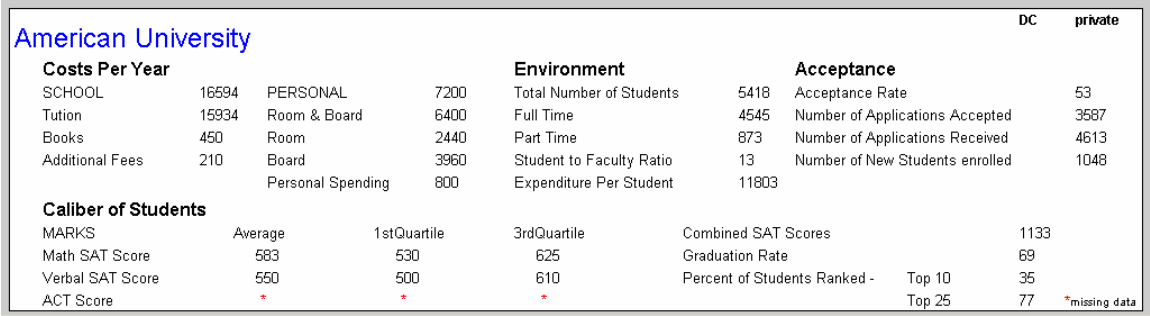



**Figure 5 a-d: Adjusting the ranges of the attributes in Cost Per Year**

Throughout this process Susie has been keeping an eye on the scatter plot to the left, which updates with each change of the slider. This visual representation of schools that meet her criteria helps her to see trends in the data, such as the fact that schools with a higher student/faculty ratio have lower costs, but also lower caliber of students. She also gets further information by a linked view; clicking on a point reveals detailed information about that particular school in the bottom pane (Figure 6). She uses this information to refine her filters, sometimes broadening her ranges when conflicting filters have excluded interesting schools.



American University — DC — private

**Costs Per Year**

| | | | |
|---|---|---|---|
| SCHOOL | 16594 | PERSONAL | 7200 |
| Tution | 15934 | Room & Board | 6400 |
| Books | 450 | Room | 2440 |
| Additional Fees | 210 | Board | 3960 |
| | | Personal Spending | 800 |

**Environment**

| | |
|---|---|
| Total Number of Students | 5418 |
| Full Time | 4545 |
| Part Time | 873 |
| Student to Faculty Ratio | 13 |
| Expenditure Per Student | 11803 |

**Acceptance**

| | |
|---|---|
| Acceptance Rate | 53 |
| Number of Applications Accepted | 3587 |
| Number of Applications Received | 4613 |
| Number of New Students enrolled | 1048 |

**Caliber of Students**

| | Average | 1stQuartile | 3rdQuartile |
|---|---|---|---|
| MARKS | | | |
| Math SAT Score | 583 | 530 | 625 |
| Verbal SAT Score | 550 | 500 | 610 |
| ACT Score | * | * | * |

| | | |
|---|---|---|
| Combined SAT Scores | | 1133 |
| Graduation Rate | | 69 |
| Percent of Students Ranked - | Top 10 | 35 |
| | Top 25 | 77 |

*missing data

**Figure 6: Detailed information pane.**

When Susie is satisfied with the collection of schools that meet the criteria she has set - knowing that she can change the ranges at any time - she turns to the task of picking a few top schools from this list. Although all of the schools meet her criteria, she wishes to investigate further the tradeoffs between different attributes. For example, she may discover that a high expenditure per student also means high tuition. How important are each of these criteria to her decision? To aid in this task she uses a ValueChart by clicking on the ValueChart button. What appears is shown in Figure 7.

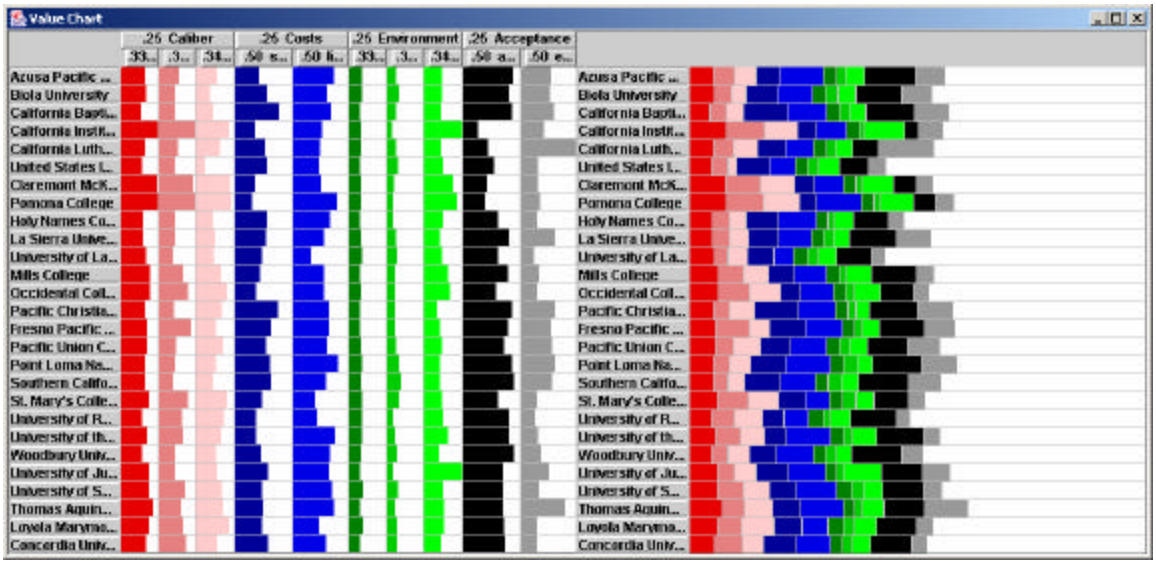



**Figure 7: ValueChart**

Susie sees all of the schools that meet her criteria down the left-hand side. Across the top are the top two levels of each attribute tree. For each school, a different coloured bar is used to represent each attribute. Initially each group of attributes is given equal spacing. The box on the right shows the combined values of all the attributes. Thomas Aquinas College looks to be the highest ranked.

Susie has very high marks and knows she'll get into the school of her choice. For this reason, acceptance rates are not important to her. She does not, however, have a lot of savings, and so she values the cost of a school highly. To reflect this, she eliminates the Acceptance attribute column, and expands the size of the Costs attribute column. The other columns adjust accordingly (Figure 8). Ponoma College is now the highest ranked school, for Susie's preferences. Susie can manipulate the different attributes to investigate tradeoffs between them. Double-clicking on an attribute allows her to reorder the schools by that column. If she wants more information on a school, she clicks on it to reveal a pane with details.

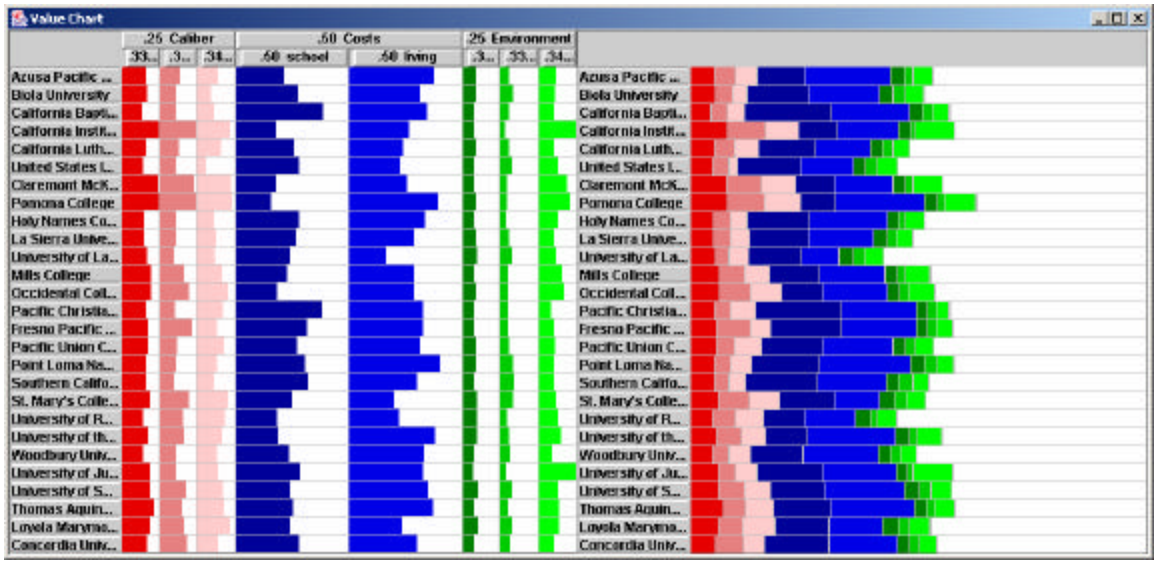



**Figure 8: ValueChart adjusted to a student's preferences**

When she has narrowed down choice down to five schools, Susie prints off the detailed information on each school to discuss with her parents and school guidance counsellor at a later time.

## Implementation

The UFinder interface was written in Java using Java Graphics API and Swing on a Windows Platform. Initially we intended to use the InfoVis toolkit (Fekete, 2004) but found that it did not suit our needs. We preferred to paint our own scatterplot, and the Java Graphics package provided us with this flexibility.

The slider is arranged as a tree. Changes to the ranges at any level of the tree are propagated upwards and downwards to recalculate the ranges for attributes on other levels. Changes to the slider submit a query to the data, and schools that are returned are painted to the screen using their calculated Caliber field as the x coordinate and Cost as

the y coordinate. Clicking on an item in the scatterplot returns the details of the school which are painted in the Details pane.

Clicking on the ValueChart button prints the list of filtered schools and their attributes to a text file which is used as input to the ValueChart program. At this stage the attributes' values are changed to reflect meaningful values in ValueCharts. All values are normalized to the range (0,1) so that they represent a percentage of a bar to be drawn. As well, all bar lengths represent positive attributes, so an inverse relation is applied to some. For example, a large Student/Faculty Ratio will be represented by a small bar and vice versa. Lastly, it would be misleading visually not to include a bar for a missing value. When there is no value for an attribute, it is replaced with the average for that attribute.

The code for ValueCharts is built upon the existing program kindly loaned to us by Guiseppe Carenini and John Lloyd. Modifications include linking clicks on the ValueCharts interface to the Details pane. The colours for ValueCharts were chosen to visually chunk the material into four distinct groups using four different shades, with brightness varying for attributes within the same tree. The information in the colours is coded redundantly for colour-blind users in the panels grouping across the top, and the varying shades beside one another.

## Results & Evaluation

To test our interface and how it supports the task of choosing a University or college, we ran an informal evaluation with 6 users. We were particularly interested how a ValueChart enhances the decision-making process over standard dynamic queries.

We split the schools into two sets, A and B. Users were presented with two interfaces, the straight UFinder interface without ValueCharts (NVC), and the interface including ValueCharts (VC). The interfaces were counterbalanced so that half the subjects saw the NVC interface first, and the others the VC interface first. Subjects received a brief demonstration of the interface before performing the task. To make the task as realistic as possible users were asked to find three schools that they would consider attending. They were not timed because speed does not indicate satisfaction with the interface. Rather, they were told to take as much time as needed to explore the interface and stop when they were completely satisfied with their choices. They then repeated the demonstration and the task with the second interface and the second set of schools. Afterwards we conducted a semi-structured interview to find out which interface the participants found more useful, which they preferred, and which led them to have the greatest confidence in their decision.

Participants found the interface with ValueCharts more useful on average than the interface without ValueCharts (Figure 9). However, they rated interfaces equally when asked how satisfied they were with the schools they chose. This indicates that although ValueCharts is perceived to be more useful, enough information was derived from the NVC interface to come to a satisfactory decision. This trend would be interesting to investigate further in a formal user study. One user commented "UFinder wasn't hard to figure out. I wouldn't base a decision solely on this, but would find it useful as a starting point."

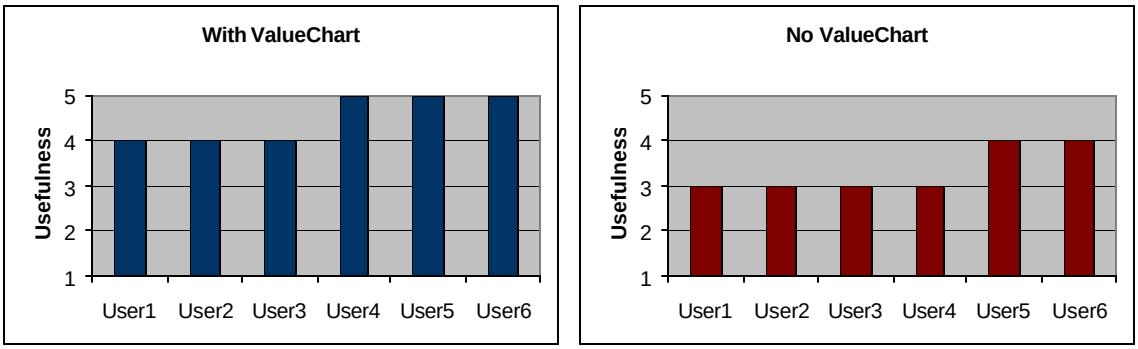



**Figure 9a-b: Subjects rankings of Usefulness (1=not useful to 5=very useful)**

When asked to select which interface they would prefer, more users preferred ValueCharts (Figure 10). Comments included "both interfaces were quite easy to use, but I felt that the second (with ValueCharts) was much more informative", and "(with ValueCharts) was easier to use because the criteria for choosing a school can be sorted in order of priority."

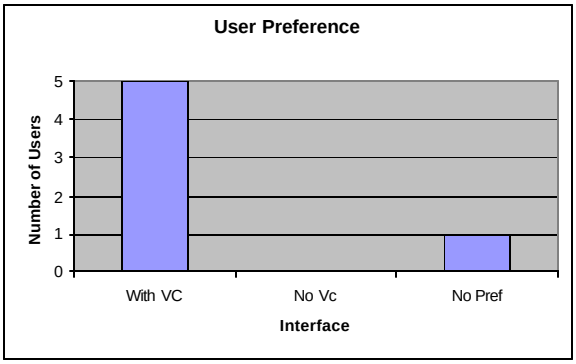



**Figure 10: User interface preference**

Several user comments pointed out improvements that could be made to the interface, such as colouring the squares in the scatterplot based on public/private status or state, adding a button to reset the sliders to their original values, or including the option to search for a particular school by its name. One user suggested making "selected" schools stay permanently on the display despite other changes of the sliders.

Our own evaluation of the interface revealed the following strengths and weaknesses. We find that the hierarchy was helpful to use when browsing the data set. It made the number of attributes much more manageable and allowed us to manipulate ranges as an appropriate level of granularity.

There are some weaknesses with the system that could be improved upon. The outliers in the data set cause many of the values to fall within a narrow range on the sliders. This also affects the relative size of bars in ValueCharts. We attempted to overcome this problem in ValueCharts by setting the outlying values to 1, and normalizing the rest over the resulting range.

A second weakness is that the interface requires a certain amount of understanding by the user. Optimally, the ValueCharts should be viewed only after filtering to roughly 20 schools or less. Secondly, the user must be encouraged to interact with the ValueCharts pane by adjusting the widths of the columns. Often the best strategy is to eliminate columns for unvalued criteria altogether. A tutorial on the how to use ValueCharts with guidelines might help first-time users fully appreciate the powerful support for decision-making that ValueCharts offers.

## Conclusions and Future Work

This project was an incredible learning experience for both students involved. As our first Java program we learned a great deal along the way about Swing components. We learned how to manipulate data, and the difficulties in dealing with a large data set for which many values are missing. We gained experience running an informal user study to test our design.

In the future we would have looked to get an initial medium-fidelity prototype off of the ground before working with the entire database. In particular this would have helped us to see the problems with outliers and missing data values earlier on, and how they would skew the range calculations and visual impact of the data. More testing with users throughout the design process would have overcome problems that were not anticipated.

Future directions for this work are to implement linked ValueCharts, so that clicking on an attribute would bring up a ValueChart for the attributes lower in that tree. It would be desirable to run a full user study to find out whether the attribute set that we chose are true factors that influence selection of which University to attend. This work could be extended to apply not just to selecting schools, but to any domain with data that are arranged hierarchically.

## References


Ahlberg, Christopher and Shneiderman, Ben. Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Display. Proc. ACM CHI 1994, pp 313-317.

Carenini, Giuseppe, and Loyd, John. ValueCharts: Analyzing Linear Models Expressing Preferences and Evaluation. In publication.

Fekete, Jean-Daniel. The InfoVis Toolkit. Version 0.6alpha2, 2004. http://www.lri.fr/~fekete/InfovisToolkit/

Keim, Daniel A., and Kriegel, Hans-Peter. VisDB: Database Exploration using Multidimensional Visualization. IEEE CG&A, 1994.

Shneiderman, Ben. The eyes have it: A task by data type taxonomy for information visualizations. Proceedings IEEE Visual Languages1996, pp. 336-343.

Tweedie, Lisa, Bob Spence, David Williams, and Ravinder Bhogal. The attribute explorer. Conference on Human Factors in Computing Systems 1994, pp. 425-436.

Williamson, Christopher, and Shneiderman, Ben. The Dynamic HomeFinder: Evaluating dynamic queries in a real-estate information exploration system. Proc. ACM SIGIR'92, pp 338-346.