Appendix E : Evaluation

Using GOMS to Evaluate our Automatic Design System

In chapters II and III we presented a framework for characterizing visualization techniques. This framework presented four classes of visualization primitives: data, mapping, graphical and rendering. It has been shown in many current hand-made visualization systems that utilizing all four classes of these primitives can significantly improve our ability to solve tasks and communicate information with visualizations. Previous work in automatic visualization design, however, only considered the use of mapping techniques. In chapter IV we outlined a set of metrics and heuristics that enable data techniques to be added into the automatic design process. These heuristics describe how data and mapping techniques can be successfully combined and effectively traded-off to best solve tasks. In appendix F we describe how graphical and rendering techniques might also be integrated into the automatic design process.

To evaluate the completeness, generality and practicality of our framework as well as the effectiveness of our metrics and heuristics, we used the theories and concepts developed in chapter II, II, and IV to implement an automatic design system (AVID – Automatic Visualization Interface Designer). We describe the implementation of AVID in detail in chapter V. Here we evaluate the results of AVID using GOMS [*Card, 1980*]. Specifically we want to test the following:

- Our theories can be implemented and they perform as expected. I.e. the design metrics and heuristics used in the designer result in output designs that are ordered according to complexity of use (by "complexity of use" we refer to cognitive, perceptual, and motoric complexity).
- Our work increases the breadth of designs that can be generated by automatic systems. I.e. our automatic system should be able to produce designs that cannot be previously generated.
- Our work improves the effectiveness of visualizations generated. I.e. the expanded design space contains visualizations that allow certain task classes to be solved more effectively.

To demonstrate the effectiveness of the expanded design space, we have chosen three tasks that span the three major classes of Exploratory Data Analysis (EDA) tasks as captured by previous work, and which we find interesting in our own work. Specifically, the three classes of tasks are: search, compute, and comparison. For each of these tasks we used our automatic system (AVID) to generate up to about 20 designs, exploring a maximum of 15000 nodes in the design space (i.e. 15000 different design states). A design state can be differentiated from all others either because the graphical elements used are different, the data encoded within the design is different, or the constraints (graphical or data) placed on the design are different.

For each task, we chose from the 20 designs, a set of about 7 or 8 that had interesting design differences and performed a GOMS analysis on each of them to estimate the total time taken by experts to perform the intended tasks. Specifically, for each task and each design we developed a procedure or an algorithm that might be executed by an **expert user** to solve the task. Implicit in these procedures are many assumptions about the cognitive, perceptual, and motoric steps executed by the user. There are obviously many different correct algorithms that may be used and a complete analysis would try to model all of them. For our purposes however, we chose a single straightforward efficient algorithm for each layout assuming **complete understanding of the graphic and task** (i.e. no time to allocated for interpreting the graphic and recalling the task). Taking the difficulty of representational interpretation into account during the automatic design process would be an interesting area of study for future work. Based on this "complete understanding" assumption, we presuppose the following rules in **all** of our GOMS evaluations:

Baseline Assumptions for ALL tasks and ALL designs

1. The user has the task committed to memory and the task does not change during the experiment.
2. The user has complete understanding of all graphic designs used and thus we **do not** account for time taken to understand a graphic (i.e. time taken to parse what data attributes are shown, what data attributes are mapped to which graphical properties, etc).

Generating a single procedure for each design is sufficient for our purposes because our goals are to contrast the different visualizations produced by our system and get some general time estimates for determining the correctness of its design ranking. For this purpose we do not need very accurate total task time measurements. It is sufficient that we determine general design groups based on approximate total time differences and based on these groups ensure that our system does indeed assign meaningful costs to its output visualizations.

We expect that this simplification will cause our models to under predict average performance time because not all users are likely to adopt efficient algorithms. It is also important to note that our procedures are based on a single data set, i.e. the data set used to generate the visualization designs. Different data sets will invariably cause different GOMS sequences and total times to be generated. Depending on how important the data distributions are to the chosen procedure, a change in data set may cause a simple multiplicative change to the total time based on number of data elements, or it may cause significant changes to the actual procedure used to solve the task. We have discussed in appendix C-4.3, how our automatic design system takes some of these data size and data distribution effects into account. A more complete treatment of these issues however is left for future work.

Most of the GOMS operators and estimated times used in our analysis are taken from Lohse et al. [*Lohse, 1993*] and John et al.[*John, 1990*]. A summary of these operators are shown below in Table E-1.

Lohse and John in turn based many of their operator times on previous empirical work as is shown in the *original source* column of Table E-1.

GOMS operator	Estimated time (msec)	Original source	Explanation
Perceive simple binary signal	100	Card et al., 1983 [*Card, 1983*]	
Perceive complex visual signal (word or code)	290	John & Newell, 1989 [*John, 1989*]	As was done in *John, 1990*, we divide the 340 msec estimated time by *John & Newell, 1989* into a perceive component (290 msec) and a verify component (which is a cognition operation taking 50 msec). This change makes a difference when processing multi-word entries where only a single verification is needed at the end of the entry. This 290 msec time also conforms well to the time listed by *Lohse, 1993*.
Cognition (mental step)	50	John & Newell, 1989 [*John, 1989*]	As was listed by *Lohse, 1993*, *Cavanaugh 1972*, *Olson, 1990*, and *Welform, 1973*, all measured various cognitive time estimation to perform a mental step, or to compare various types of objects in working memory. These empirical times range from 33 msec to 92 msec. The average time of all these operations came up to 55 msec. This figure is similar to the 50 msec time estimated by *John & Newell 1989* which we will adopt in our GOMS sequences.
Eye saccade (travel time)	30	John & Newell, 1990 [*John, 1990*]	*Card 1983*, estimated eye movement times to be 230 msec. This figure was later refined in *John & Newell 1990* and divided up into smaller steps taking into account the time needed to initiate the eye movement, the actual travel time, and the fixation time. In our analysis we will use the latter model for better time estimation accuracy.
Homing between devices	350	Card et al., 1983 [*Card, 1983*]	In general, *Card 1983* estimated that the time taken to move between devices is 350 msec. In our analysis however, the only devices that the user needs to move between include the numeric keypad and the mouse. Since both these devices are commonly placed in close spatial proximity (assuming right handed mouse use), the time taken to move between the two devices will likely be smaller. Using Fitts Law, we estimated the time taken for the move to be approximately 132 msec (assuming 6 inch distance between mouse and numeric keypad, and 3 inch width for the numeric keypad).
Hand movement between numeric keypad and mouse	132	derived from Fitts Law [*Card, 1983*]	
Horizontal movement within numeric keys	40	derived from [*John, 1988*]	*Lohse, 1993* estimated a keypunch entry to take 372 msec, however in our analysis we decided to break up keypunch entries into smaller steps including upstrokes, downstrokes and finger movements as was done in *John, 1990*. This will allow us to more accurately model time savings for numbers with many repetitious digits.
Upstroke	60	[*John, 1988*]	
Down-stroke	60	[*John, 1988*]	

Table E-1: Summary of all GOMS operators used in the evaluation sequences listed in this appendix

The results of all the GOMS analyses for each of the three EDA tasks (search, computation, comparison) are shown in subsequent sections. Essentially our evaluation test designs and GOMS estimates show the following:

- The output order of our automatic designer (i.e. our design heuristics used) does indeed conform to cognitive, perceptual, and motoric complexity as computed by GOMS. I.e. the theoretical concepts developed here for characterizing and expanding the visualization techniques design space for automatic visualization generation can be implemented and the results are meaningful (i.e. conforms to GOMS computed times).
- Adding data transform techniques into the automatic design process expands the visualization design space and enables whole new sets of interactive and non-interactive visualizations to be generated. Many of these designs are shown in the following sections.
- Some of the new designs generated as a result of work developed in this thesis (i.e. pure data transform techniques and hybrid data + mapping designs) perform much better than the designs that can be generated with current state of the art technology (i.e. pure mapping designs) for the task classes we considered (*computation, search, comparison*); with the highest gain in computation tasks.

In section E-5 we outline our reasons for using GOMS as an evaluation method and describe why it can be appropriately applied for our purposes.

E-1 Task 1: Find Task

Finding a "Good" University based on Graduation Rates and Test Scores

Suppose we are looking for a "good" university to attend. Some of the attributes we may be interested in include the quality of students attending the school (which can be deduced by looking at their ACT and SAT scores) as well as whether the school has a good track record for graduating its students. The task specification entered into our automatic designer is shown below:

```
(lookup
  (and-obj
   (find-obj '(VALUE . GREATER-THAN)
             (lookup '(OBJECT . nil) '(VALUE . AVG_COMBINED_SAT))
             '(VALUE . ?))
   (find-obj '(VALUE . GREATER-THAN)
             (lookup '(OBJECT . nil) '(VALUE . AVG_ACT))
             '(VALUE . ?))

   (find-obj '(VALUE . GREATER-THAN)
             (lookup '(OBJECT . nil) '(VALUE . GRADUATION_RATE))
             '(VALUE . ?))
   )
  '(VALUE . OBJECT-NAME)
  )
```

Task E-1: Search for universities based on the average SAT and ACT scores of attending students as well as graduation rates.

From analyzing the set of visualization displays generated by our system, we were able to determine the schools that performed well based on the three search attributes: *Duke, Emory, Vanderbilt, University of Pennsylvania, Massachusetts Institute of Technology,* and *Brown University*. It is perhaps not too surprising that all of these universities are private schools.

The task specification entered into our designer provides for some amount of flexibility in the search parameters. The wild card "?" symbols indicate to the designer that we are not sure of the exact *scores* and *graduation rate* thresholds of interest in our search, so that the designs generated will allow users to experiment with different search thresholds. In the interest of simplicity, however, we only estimate the time for setting one set of thresholds in the GOMS sequences below. Increasing the number of different threshold conditions tested will not change the general GOMS procedures used, but merely result in a higher time multiplicative cost commensurate to the number of different threshold tests.

In the next sections we present a set of seven output designs generated by our designer, ordered from best (least system assigned cost) to worst (highest system assigned cost). Each design is accompanied with a description of the visualization as well as a GOMS evaluation showing the steps taken to complete the above task and the total time taken. In the summary section we compare the GOMS estimated times for all seven designs and provide an analysis of the results and its impact on our goals.

In all following GOMS analysis for this task we assume:

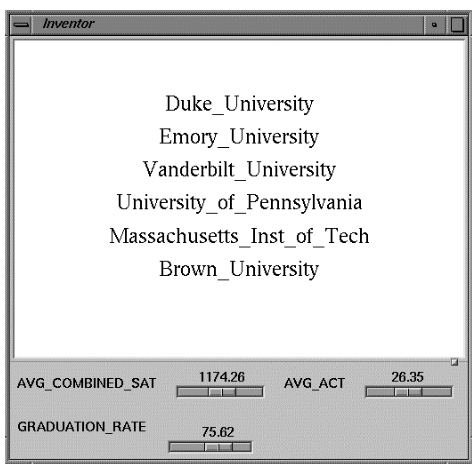
***Baseline Assumptions for Task 1 (Search Task)*:**

1. Only one set of threshold conditions is used. In tasks of this type it is common for users to experiment with different search constraints until a satisfying number of universities are retrieved. Setting overly high threshold values may cause no universities to fit the conditions of the search while setting threshold values that are too low may cause too many universities to fulfill our search criteria. However, to model the actual number of task conditions a user would actually try out in a session, we would need to account for user preferences and experience, the difficulty of the interface, the importance of the task, etc. Such modeling, if possible, is beyond the capabilities of the analysis techniques used here. As a result, all the GOMS sequences for this task only accounts for one set of search conditions. We will however show how experimenting with multiple search conditions can change the total estimated time for each design.
2. Assume that the user's hand is already on the mouse at the start of the task. We believe that this is a reasonable assumption since the user is most likely already working or interacting with the computer when the designs are shown to them.
3. We assume that the user has some approximate threshold values in mind for each of the three attributes. To get to each of the threshold values, the user must either make multiple moves on a slider

input device, or scan around the different encoding positional axes. In general we assume that three adjustments are needed to get to the correct value on a slider or to find the desired value on a positional axis. This figure is based on our own experiences in using the generated interfaces. We want to point out, however, that since both operations have comparable time estimates (320 + 490*n* msec for manipulating a slider and 470*n* for an axis lookup, where *n* is the number of adjustments needed), there isn't much of a time effect if this number (*n*) turns out to be slightly higher or lower. We anticipate that this number may range from about 1 to 20 depending on the adeptness of the user in manipulating devices (physical or virtual), the sensitivity of the slider, the length of the slider, the accuracy of manipulating the slider, the task accuracy required, the task importance etc. Such variation however does not cause significant time changes among the designs. When $n = 20$, the estimated time difference is 400 msec. only.

E-1.1 Design 1

In this design the search results are pre-computed by the system and only the universities that pass all of the search constraints are shown. The visualization interface generated by our automatic designer provides three sliders, each allowing the user to input a threshold constraint for each of the search attributes (*avg_combined_SAT*, *avg_ACT*, and *graduation_rate*). A big advantage of this design is the reduction of visual clutter due to the fact that the search data need not be shown to users, as their results have been pre-processed by the system. The nature of the search task allows even greater visual benefits because we can prune the number of objects in the display so that only the relevant ones are shown.



To solve the task using this design users must set the sliders to the relevant threshold values and then read off the university names. The time taken for this design is mostly attributed to the effort of manipulating the sliders. We assume that to get the desired threshold value on a slider, the user needs to adjust the slider an average of three times.

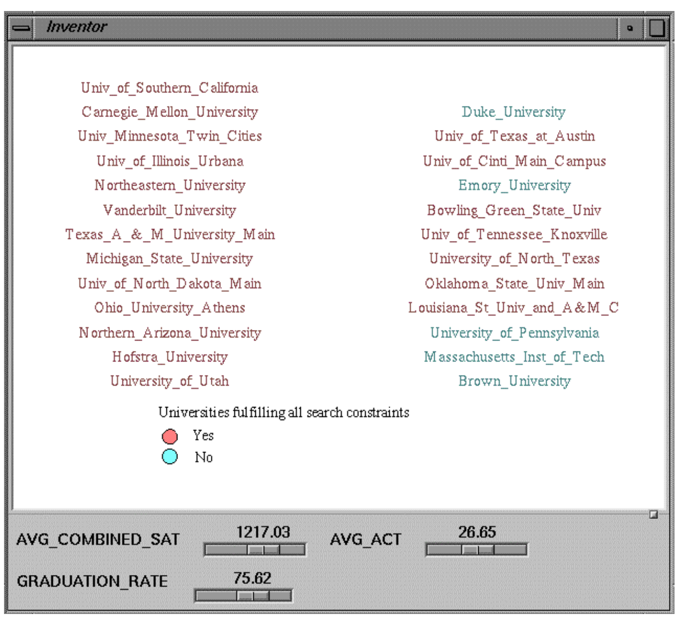
<table>
<tr><th>General goal</th><th colspan="2">Cognitive, perceptual, or articulatory step taken by user</th><th>Time taken (msec)</th><th>Sub total (msec)</th><th>Total time (msec)</th><th>Target objects (data or graphics)</th></tr>
<tr><td rowspan="4">Alter first slider to required SAT threshold score</td><td colspan="2">Attend alter slider</td><td>50</td><td></td><td></td><td rowspan="4"></td></tr>
<tr><td colspan="2">Initiate finger drop</td><td>50</td><td></td><td></td></tr>
<tr><td colspan="2">Finger drop</td><td>60</td><td></td><td></td></tr>
<tr><td colspan="2">Sub-total</td><td></td><td>160</td><td></td></tr>
<tr><td rowspan="6">Shift the slider until the correct threshold value appears. Note that here, the process of performing the finger moves necessary to complete the task is performed in tandem with scanning the slider to determine the current threshold value. In this case since the hand move required is minute, the time is dominated by the scan and reading operation.</td><td colspan="2">Attend hand move</td><td>50</td><td></td><td></td><td rowspan="6">approx. 1180 for SAT
approx. 26.5 for ACT
approx. 76% for graduation rate</td></tr>
<tr><td colspan="2">Initiate hand move</td><td>50</td><td></td><td></td></tr>
<tr><td>Hand move</td><td>Attend check the correctness of current threshold</td><td>50</td><td></td><td></td></tr>
<tr><td></td><td>Read threshold</td><td>290</td><td></td><td></td></tr>
<tr><td></td><td>Verify results</td><td>50</td><td></td><td></td></tr>
<tr><td colspan="2">Sub-total</td><td></td><td>490</td><td></td></tr>
<tr><td>As was previously noted, we assume that three slider moves are necessary to get to the desired threshold value.</td><td colspan="2">3 * 490</td><td></td><td></td><td>1470</td><td></td></tr>
<tr><td rowspan="4">Finish manipulating slider</td><td colspan="2">Attend finger lift</td><td>50</td><td></td><td></td><td rowspan="4"></td></tr>
<tr><td colspan="2">Initiate finger lift</td><td>50</td><td></td><td></td></tr>
<tr><td colspan="2">Finger lift</td><td>60</td><td></td><td></td></tr>
<tr><td colspan="2">Sub-total</td><td></td><td>160</td><td></td></tr>
<tr><td>Total time for manipulating ONE slider</td><td colspan="2">160 + 1470 + 160</td><td>1790</td><td></td><td></td><td></td></tr>
<tr><td>Total time to manipulate all sliders</td><td colspan="2">3 * 1790</td><td></td><td></td><td>5370</td><td></td></tr>
<tr><td rowspan="8">Read university names. Here we assume that there are a limited number of names (six), thus the user can get to them with a limited number eye movements. We assume the user can get to three names with one eye movement based on the required font size for reading.</td><td colspan="2">Attend name read</td><td>50</td><td></td><td></td><td rowspan="8">1. Duke,
2. Emory,
3. Vanderbilt,
4. University of Pennsylvania,
5. Massachusetts Institute of Technology,
6. Brown University.</td></tr>
<tr><td colspan="2">Initiate eye movement</td><td>50</td><td></td><td></td></tr>
<tr><td colspan="2">Eye movement</td><td>30</td><td></td><td></td></tr>
<tr><td colspan="2">Read 3 university names with info. verification
3 * (3 * (290 + 50))</td><td>3060</td><td></td><td></td></tr>
<tr><td colspan="2">Initiate eye movement</td><td>50</td><td></td><td></td></tr>
<tr><td colspan="2">Eye movement</td><td>30</td><td></td><td></td></tr>
<tr><td colspan="2">Read 3 university names</td><td>3060</td><td></td><td></td></tr>
<tr><td colspan="2">Sub-total</td><td></td><td>6330</td><td>6330</td></tr>
<tr><td>Total time</td><td colspan="2">5370 + 6330</td><td></td><td></td><td>11700</td><td></td></tr>
</table>

It is important to note that a major part of the total time in the design above (approximately half) can be attributed to the time taken to manipulate the three sliders. This component increases the more sets of search conditions we want to test. Each change in condition increases the total time by 1790 msec for manipulating a slider and an additional 280 msec for a quick scan of the results to see whether we have the approximate number of data concepts desired (as is shown below). Thus total time for each search condition change is 2070 msec. Note that this figure assumes that three minute adjustments are needed to get to the desired threshold value on the slider. As a user gets more familiar with the slider mappings and sensitivity, this figure may be reduced at which time the cost of each change may only take 810 msec + 280 msec = 1090 msec (assuming that no adjustments are needed). Also note that when scanning for the number of elements retrieved, we assume that no counting is required because we are not interested in the exact number of elements, just in whether the approximate number of elements retrieved falls within our task requirements.

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Quick scan of results	Attend see approximate number of objects returned by search	50			
	Initiate eye movement	30			
	Perceive results (This is a pre-attentive operation)	100			
	Compare general size with desired size	50			
	Verify results	50			
	Sub-total		280		

E-1.2 Design 2

This design is very similar to the previous one in that the search results are pre-computed. However, instead of filtering the visualization display to only show the universities that fulfill the search, we map the search results to *hue (color)*. *Blue* universities indicate those that pass the search and *red* universities indicate those that don’t. Because more data concepts are shown in this design, and an additional graphical property (*color* or *hue*) is used to show the search results, this design is rated lower than the previous one.



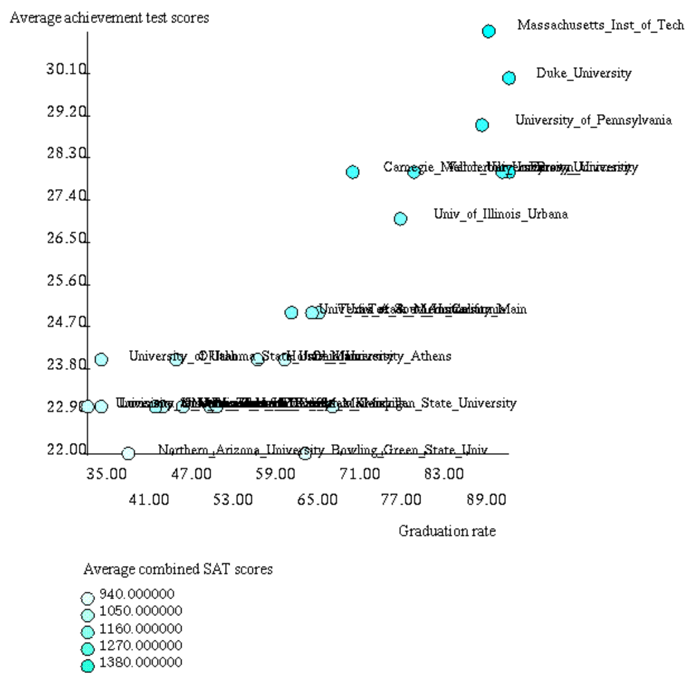
The GOMS procedure used for this design is identical to the previous one, except in the last portion when we are reading the university names. Rather than just reading the names top down as in the previous example we need to move our eye to each *blue* colored university before reading their names. This results in a slightly higher time cost for processing the end results.

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Total time taken for slider manipulation (taken from time calculated in Design 1)		5370		**5370**	approx. 1180 for SAT approx. 26.5 for ACT approx. 76% for graduation rate
Scan to first blue label	Attend get next blue label	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive label	100			
	Verify label is blue	50			
	Sub-total		280		
Read label name	Attend label read	50			1. Duke, 2. Emory, 3. Vanderbilt, 4. University of Pennsylvania, 5. Massachusetts Institute of Technology, 6. Brown University.
	Read university name	870			
	Verify results	50			
	Sub-total		970		
Total time to process each blue label	970 + 280	1250			
Repeat for each blue colored label	6 * 1250			**7500**	
Total time	5370 + 7500			**12870**	

An advantage of this design is that all of the universities are shown, thus users will not get disoriented from changes in the number and positions of the universities caused by changing search conditions as in the previous case. In most cases however, the effect of such disorientation is minimal, and it can only become significant when we are trying to track particular data concepts across many different search conditions. Because the slider manipulation portion of this design is identical to the previous one, the cost of experimenting with different search conditions is also identical at 1790 msec or 1090 msec. for experts.

E-1.3 Design 3

This design unlike the two previous ones has a completely perceptual design, i.e. all of the task data are mapped to graphics and it is up to the user to perform the search perceptually. Each mark cluster represents a university. *Avg_combined_SAT* score is mapped to mark *saturation*, *avg_ACT* score is mapped to *y-position*, *graduation_rate* is mapped to mark *x-position*, and *university_name* is mapped to a *text label* next to each mark. An advantage of this design is that no input device manipulation is required. A weakness is the additional load of performing the search perceptually. In addition, more data has to be mapped and this might sometimes cause readability problems, as is the case below where some of the objects are occluded. Currently our automatic designer is unable to provide solutions to such readability issues but refer to appendix F to see how such additions might be integrated into our automatic design system in the future. Our designer assigned a higher cost to this visualization because of the two reasons stated above: additional perceptual load and additional visual clutter.



To solve the search task using this design, we first refer to the *saturation* legend to determine the appropriate saturation color for our desired *avg_combined_SAT* threshold. Since there are only 5 gradations to the saturation legend, we can zero in on the last two values in the legend and interpolate between them (i.e. only two values need to be processed). After processing the saturation legend we look to each of the positional axes to determine the position of the *avg_ACT* score and *graduation_rate* thresholds that we are interested in for the search. As with the case of the slider adjustments, we assume that the user will not be able to get to the exact position on an axis with the desired threshold value on the first try, but an average of three eye movements are required. Once the positions are determined, we can perceptually demarcate an area containing all the universities that fit our *avg_ACT* score and *graduation_rate* criteria. Based on our task we are interested in high *avg_ACT* score and *graduation_rate* and this is captured by objects in the upper-right hand corner of the chart. From all the marks in this area we choose the ones with greater *saturation* values than our desired threshold (which we determined at the start of the procedure). When we have identified the relevant marks, we read off the associated university name that is encoded next to it.

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Lookup *saturation* legend for desired *SAT score saturation* value.	Attend legend lookup	50			
	Initiate eye movement	50			
	Eye movement	30			
	Read last value	290			
	Verify value	50			
	Read previous value	290			
	Verify value	50			
	Attend perform saturation interpolation based on values	50			
	Perceive saturation values	100			
	Interpolate	50			
	Verify saturation	50			
	Sub-total		1060	1060	
Note that in this legend lookup step we assume that the user is able to keep the determined saturation value and use it when it is required later. Also note that since the saturation legend scale only has five gradations, it is probably sufficient for the user to zero in on the last mark (most saturated mark) and do some slight interpolation estimate of saturation value with the previous mark to get the general saturation level required. As a result no adjustments are needed to get to the desired saturation threshold, as is the case with the two axis positions next.					
Scan to desired *graduation rate* on *x-axis*	Attend scan	50			
	Initiate eye movement	50			
	Eye movement	30			
	Read value	290			
	Verify value	50			
	Sub-total		470		
Repeat 3 times to get to desired value	(3 * 470)	1410		1410	
Scan to desired *ACT score* on *y-axis* with 3 repetitions to get desired value		1410		1410	
Scan to xy-position which corresponds to desired *graduation rate* and *ACT score*	Attend scan	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive general area	100			
	Sub-total		230	230	
Scan for marks with the requisite saturation values as determined in a previous step. This is a pre-attentive operation thus users should be able to zero in on the relevant saturation without having to attend to each mark	Attend scan for marks with requisite saturation	50			MIT, Illinois, Pennsylvania, Duke (Emory, Vanderbilt, and Brown are occluded)
	Initiate eye movement	50			
	Eye movement	30			
	Perceive mark saturation	100			
	Verify mark saturation	50			
	Attend university name read	50			
	Read university name (3 * 290)	870			
	Verify university name	50			
			1250		
Repeat 7 times for each relevant university. Note that in actuality there are only 6 universities that pass all search conditions but because of saturation inaccuracies 7 are found instead. (assume for here that there is no	(7 * 1250)	8750		8750	1. Duke, 2. Illinois, 3. Emory, 4. Vanderbilt, 5. University of Pennsylvania, 6. MIT,

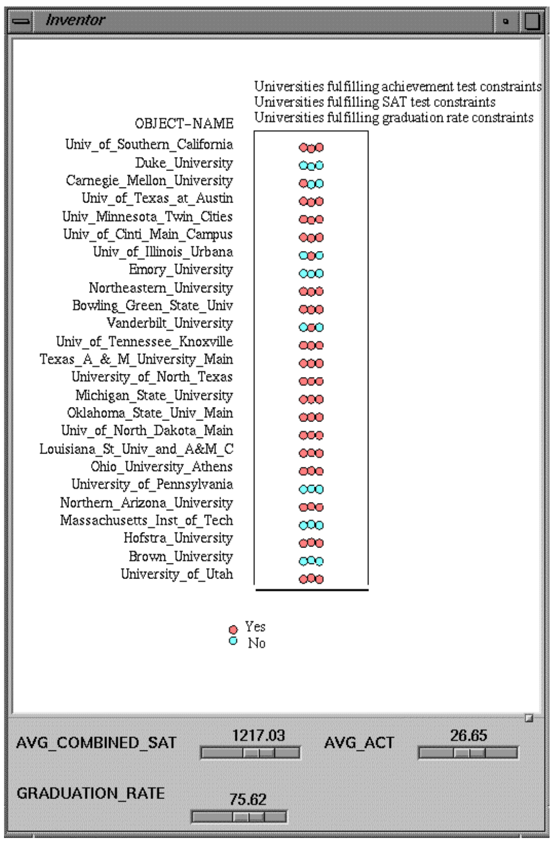
occlusion so all labels can be read)					7. Brown University.
Total time	1060 + 1410 + 1410 + 230 + 8750			**12860**	

It is interesting to note that this design actually has a very slightly lower GOMS estimated total time compared to Design 2. However it should be pointed out that the GOMS estimation does not take into account the fact that there is occlusion in the display and the task **cannot** be fully completed using the given design. In particular *Emory, Vanderbilt* and *Brown* are occluded and cannot be extracted from the visualization display. A better label placement algorithm would improve this situation. With interactive enhancements, it is also possible to solve this problem as with labels on demand [*Plaisant, 1996*], however, these enhancements will require input device manipulation and this will result in increases in the total time taken. Another important issue here is the fact that saturation is not a very accurate encoding property and as a result mistakes may be made in identifying universities with the required *avg_combined_SAT* score (which is the data attribute encoded with saturation). An example in this design is *University of Illinois* which passes both the *avg_ACT* and *graduation_rate* conditions but not the *avg_combined_SAT* condition. However because of the lack of accuracy in saturation, it is incorrectly chosen as an acceptable search candidate.

It should be noted however that the time taken for task condition adjustments here are negligible. This is because once the general search area (i.e. upper right hand corner) is located and the general required saturation values are noted, minor adjustments in the search would only involve some small adjustments to the current search area or acceptable saturation values and we do not even need to consult the axes or legends. In contrast, even small changes in search conditions require the same slider movements in the previous designs. Thus in cases where the data set is small (not high probability of readability problems) but the search conditions need to be changed often, a purely perceptual design is probably superior to the pre-computed ones. In fact as was shown in appendix C-4.2.2, in search tasks that require less information to be shown or fewer search attributes, our designer picked the perceptual design over the data computed design because of the additional input device manipulation load associated with the latter.

E-1.4 Design 4

This design is a hybrid design where each of the search conditions in the task are computed by the system, but the *AND* task is left to be performed perceptually by the user. Each university concept therefore has three marks associated with it, indicating whether it passed each of the search conditions. A *blue* mark indicates that a search condition was fulfilled while a *red* mark indicates that a search condition was **not** fulfilled. Similar to Design 1 and Design 2, the interface has a set of sliders through which users may use to set the current search condition thresholds. This design is ranked lower than the previous designs because unlike Design 1 and Design 2 which only shows the university names, this design shows four objects, the university names together with a cluster of three marks representing each of the search conditions. The left-most mark represents the *graduation_rate* search condition, the middle mark represents the *avg_combined_SAT* score search condition, and finally the right-most mark represents the *avg_ACT* score condition. In addition to the increase in perceptual complexity of the graphic, the perceptual load is also increased because now users must deduce the results of the *AND* task perceptually. However, the advantage of this design is that each of the search conditions are shown, and in cases where the user must trade off one condition against another, it is useful to have each of the search condition results available for perusal. If we compare Design 3 and Design 4, our system gave preference to Design 3 because while there is slightly less perceptual complexity here (.i.e. fewer different graphical properties are used and the results are accurate), there is the added cost of slider manipulation.



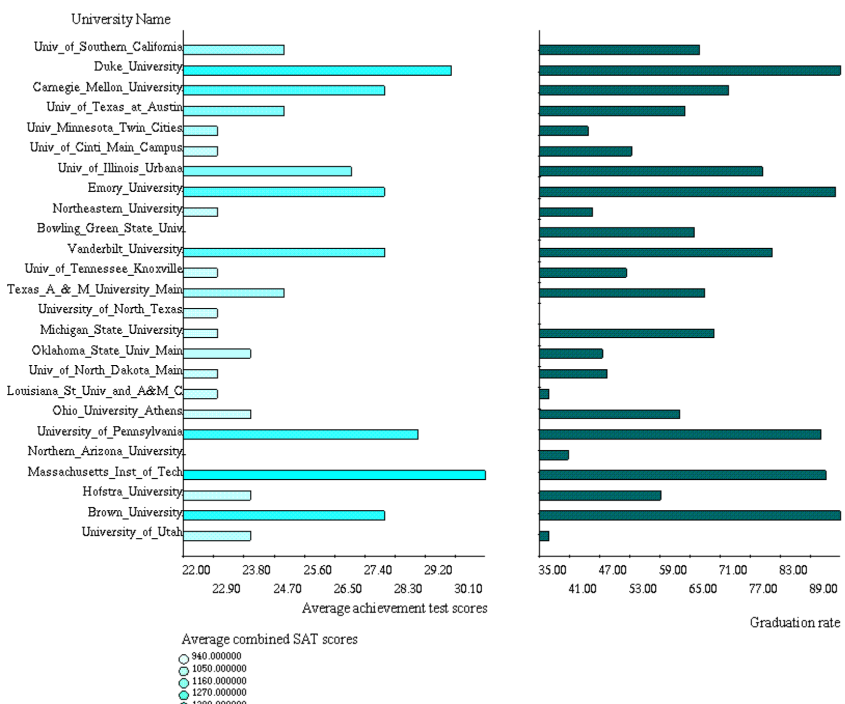
To solve the search task using this design, we must initially put the sliders to the desired threshold values (similar to Design 1 and Design 2). Once we have done this, we can find all relevant universities by scanning down the table and picking universities that have a three-mark cluster that is all *blue*. While scanning for *blue* objects is pre-attentive, scan for rows with **all** *blue* objects is not. Therefore we assume that the user must visit each row that contains at least one blue object. In the example above, this only increases the number of rows visited by two. However, depending on changing task conditions and the data distribution, this figure may increase.

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Total time taken for slider manipulation				**5370**	approx. 1180 for SAT approx. 26.5 for ACT approx. 76% for graduation rate
Scan to next row with blue in it, and verify that all three marks are blue	Attend get next row with blue objects	50			1. Duke, 2. CMU, 3. Illinois, 4. Emory, 5. Vanderbilt, 6. University of Pennsylvania, 7. Massachusetts Institute of Technology, 8. Brown University.
	Initiate eye movement	50			
	Eye movement	30			
	Perceive results	100			
	Verify all three marks are blue	50			
	Sub-total		280		
Repeat for each row with blue in it	8 * 280	2240		**2240**	
If row contains all blue marks, scan to left to read university name.	Attend read university name	50			
	Initiate eye movement	50			
	Eye movement	30			
	Read university name (3 * 290)	870			
	Verify results	50			
	Sub-total		1050		
There are 6 rows with three blue marks	6 * 1050	6300		**6300**	
Total time	5370 + 2240 + 6300			**13910**	

We would like to point out that although the total estimated time for this design is fairly close to those of previous designs, there is a significantly higher cost associated with changing the search conditions. This is because with this design, it is more difficult to quickly determine the number of universities that actually pass all search conditions. In all previous cases this operation is fully pre-attentive, and users can get a general feel for the size of the search results by just looking once at the visualization display as a whole. In contrast, this design requires users to attend to each row with blue objects in it to determine whether it actually passes all search conditions. Specifically, even though there may be a lot of *blue* dots in the display (which we can determine pre-attentively) it does not mean that many universities passed all search conditions (i.e. there may be many rows with one or two *blue* marks but not three). To determine the actual number of objects that fulfill the search, we need to attend to each row containing *blue* objects. Thus the cost of each search condition change is 1790 msec for the slider manipulation + 280*n* msec for processing each *blue* row (where *n* represents the number of rows with *blue* in them). For larger *n*, the time taken could become very significant. We also want to point out that when there are many *blue* dots in the display, an effective alternative for finding concepts that fulfill all search conditions might be to look for rows with *red* in them, and then just subtract the total number of rows from the number of rows with *red* dots (i.e. concepts that did not pass at least one search condition). Thus *n* is bounded by half the size of the entire data set. Another alternative strategy is to only visually process the column of marks that correspond to the changing condition, and then only attend to those rows where there are changes. This strategy however results in approximately the same time estimation because each candidate row has to be attended to individually.

E-1.5 Design 5

This design is a purely perceptual design similar to Design 3. Unlike Design 3 however, the three search attributes are spread over two different charts. *University name* is mapped on the *y-axis*, *avg_combined_SAT* score is mapped to bar *saturation* on the left chart, *avg_ACT* score is mapped to *x-length* on the left chart and *graduation_rate* is mapped to *x-length* on the right chart. One notable difference between this design and Design 3 is that here, there is no occlusion among the elements because now the information is separated over a greater number of spaces. While not having occlusion problems is a big plus, this less integrated design also increases the perceptual load and subsequently the total time taken to complete the task.



The GOMS procedure for this design begins with a lookup on the *saturation* legend to determine the saturation value corresponding to our desired *avg_combined_SAT* threshold, similar to what was done in Design 3. Next, we begin with the left chart (assuming the Western convention of reading from left to right) and lookup the appropriate *avg_ACT* threshold value on the *x-axis* and then proceed to the right chart to lookup the appropriate *graduation_rate* threshold there as well. Once we have determined the *x-positions* of our desired threshold values, we begin scanning up the display, and processing any bar that intersects our scan line. For each of these bars, we first check their *saturation* value to ensure that they do indeed pass our *avg_combined_SAT* threshold value. If they do, we scan to the right chart and determine whether the associated *graduation_rate* passes our desired threshold *x-position*. We do this by looking to the rightmost end of the bar and then comparing that with our remembered threshold position for *graduation_rate*.

Note that an important consideration for the GOMS procedure of this task is the number of times the axes of the two charts must be consulted to get the proper *avg_ACT* and *graduation_rate* search threshold positions. In Design 3, we assumed that the axes only had to be referred to once in the entire data analysis process and the user is able to maintain the general search area and *saturation* in STM (short term memory). However, this assumption no longer holds true here because the information is separated over multiple spaces, and when processing each space we lose our positional context in the other space. Since we must switch from space to space when processing each university row it is difficult to accurately maintain the search threshold positions in either of the spaces. As a result several references may need to be made to the axes as a reminder of the search threshold positions. In the GOMS procedure below we assume that references are made to the axes for those bars that are relatively close to the desired threshold positions. This is because when bars greatly exceed or do not meet the positional thresholds, users can tell whether a search condition was met without having to accurately remember the exact search positions.

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Lookup *saturation* legend for desired *SAT score saturation* value. (computed in Design 3)		1060		1060	
Scan to desired *graduation_rate* on *x-axis* of right chart with 3 repetitions to get desired value. (computed in Design 3)		1410		1410	
Scan to desired *ACT score* on *x-axis* with 3 repetitions to get desired value. (computed in Design 3)		1410		1410	
Scan in left chart and determine the next bar above the *avg_ACT* threshold value (assume that approximate threshold position can be kept in STM)	Attend determine if left bar length passes threshold position	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive rightmost edge	100			
	Verify if greater than threshold *avg_ACT score* position	50			
	Sub-total		280		
Check that saturation value passes desired *avg_combined_SAT* threshold.	Attend check saturation	50			
	Compare saturation	50			
	Verify results	50			
	Sub-total		150		

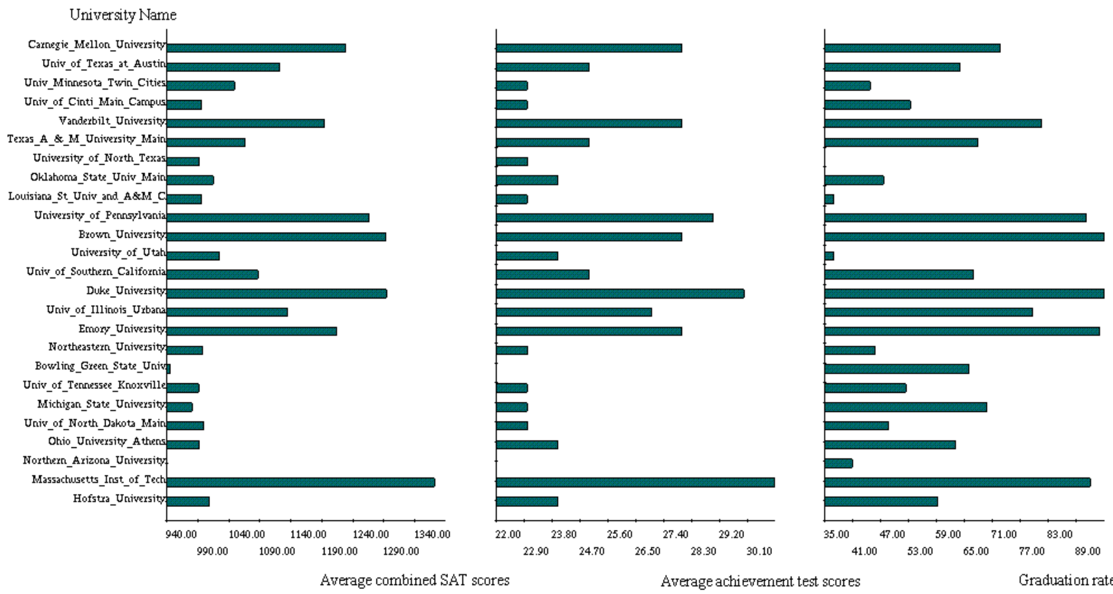
Scan to rightmost edge of bar on the right chart in the current row	Attend determine if right bar length passes threshold position	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive rightmost edge	100			
	Verify if greater than threshold *graduation_rate* position	50			
	Sub-total		280		
Total time to process one bar row (assuming no references to axes)	(280 + 150 + 280)	710			
Repeat for each candidate bar row	8 * 710	5680		5680	1. Duke, 2. CMU, 3. Illinois, 4. Emory, 5. Vanderbilt, 6. University of Pennsylvania, 7. Massachusetts Institute of Technology, 8. Brown University.
Time taken for axes references: Note that axes references are needed for bars in the first and second charts that are considered in the above process, and that fall close to the *x-position* threshold positions of their respective charts.					
Time taken for each axis reference	Attend axis reference	50			
	Initiate eye movement	50			
	Eye movement	30			
	Read closest label	290			
	Compare label with desired search threshold	50			
	Verify if axis passes the desired search threshold.	50			
	Sub-total		520		
Three axis references are required in the left chart	3 * 520	1560			Brown, Vanderbilt, Illinois
Three axis references are required in the right chart	3 * 520	1560			Vanderbilt, Illinois, CMU
Total time for axes references	1560 + 1560			3120	
For each university row that fulfills all search conditions, scan left and read the relevant university name. There are in actuality 6 universities that pass all search conditions, however in this design 7 universities are found, including *University of Illinois* because of inaccuracies in the saturation encoding.					
Scan to left to get single university name	Attend get university name	50			
	Initiate eye movement	50			
	Eye movement	30			
	Read university name (3 * 290)	870			
	Verify name	50			
	Sub-total		1050		
Repeat for each university that passed all conditions	7 * 1050			7350	1. Duke, 2. Illinois, 3. Emory,

					4. Vanderbilt, 5. University of Pennsylvania, 6. Massachusetts Institute of Technology, 7. Brown University.
Total time	1060 + 1410 + 1410 + 5680 + 3120 + 7350			**20030**	

As can be seen from the GOMS procedure, the lack of integration in this design, in addition to the loss in accuracy from the *saturation* encoding resulted in a greater total processing time compared to the previous examples. The cost of changing the search conditions in this design is also relatively large. This is because any change in condition requires new candidate rows to be processed for acceptability or current rows to be re-processed for possible rejection. As was calculated in the GOMS table above, processing each row takes approximately 710 msec without including time for the positional axes reference. Each change in condition may require any number of rows to be processed depending on the size of the change. Even with a processing cost as low as two to three rows per change, the time taken exceeds that of previous designs. This is because unlike Design 1, Design 2, and Design 3, we cannot pre-attentively get a feel for the change in number of universities that fulfill our search conditions with each alteration. Also note that, as with Design 3 there are some inaccuracies here due to the *saturation* encoding of the *avg_combined_SAT* attribute. Specifically *University of Illinois* was chosen as an acceptable search candidate even though it failed to pass the required *avg_combined_SAT* threshold of 1180.

E-1.6 Design 6

Like the previous design, this visualization requires that the task be performed completely perceptually. However, the design separates out the search attributes into three charts, with each chart showing a single search attribute as bar lengths. The left-most chart shows *avg_combined_SAT* scores, the middle chart shows *avg_ACT* scores, and the right-most chart show *graduation_rate*. All three charts are aligned on their *y-axis* which encodes the *university names*. This design is rated lower by our designer specifically because of its extreme lack of data integration. As we have shown in the previous GOMS estimation, lack of integration can often cause significant additional perceptual loads from having to switch our attention from space to space, and keeping perceptual context during these switches. As we will show below this design is no exception. An interesting difference between this design and the previous one however is that *saturation* is no longer used to encode any attribute. As a result there are no errors stemming from our inability to interpret *saturation* values with great enough precision, as was the case previously.



The GOMS procedure for this design is very similar to the previous one except that here, instead of processing *saturation* we process an additional graphical chart. Processing again begins with getting the general threshold positions in each of the three charts. Once we have determined the threshold positions we process bars in each of the three charts, determining in each case whether the length is greater than our threshold position. Initially we look for a bar that passes the *avg_combined_SAT* condition in the first chart, then we proceed to subsequent charts to determine whether the other bars for the current university pass the other two conditions (*avg_ACT* and *graduation_rate*) as well. As was done previously, we will assume that axes references are only needed in those cases where the bar lengths are close to the search threshold positions.

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Scan to desired *avg_combined_SAT* on *x-axis* of left-most chart with 3 repetitions to get desired value. (computed in Design 3)		1410		1410	
Scan to desired *ACT score* on *x-axis* of the middle chart with 3 repetitions to get desired value. (computed in Design 3)		1410		1410	
Scan to desired *graduation_rate* on *x-axis* of right-most chart with 3 repetitions to get desired value. (computed in Design 3)		1410		1410	

Now we begin processing each candidate bar row:					
Scan in left-most chart and determine the next bar above the *avg_SAT* threshold value (assume that approximate threshold position can be kept in STM)	Attend get next bar that passes the *avg_ACT* threshold	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive results	100			
	Verify results	50			
	Sub-total		280		
Scan to rightmost edge of bar on the middle chart in the current row and determine whether it passes the *avg_ACT* threshold value.	Attend determine if bar passes *avg_ACT* threshold	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive rightmost edge	100			
	Verify if greater than threshold *avg_ACT* position	50			
	Sub-total		280		
Scan to rightmost edge of bar on the right-most chart in the current row and determine whether it passes the *graduation_rate* threshold value.	Attend determine if bar passes *graduation_rate* threshold	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive rightmost edge	100			
	Verify if greater than threshold *graduation_rate* position	50			
	Sub-total		280		
Total time to process one bar row (assuming no references to axes)	(280 + 280 + 280)	840			
Repeat for each candidate bar row. Note that *University of Illinois* is considered a candidate in the first chart, but rejected right off, thus this additional processing adds 280 msec to the total processing time.	(7 * 840) + 280	6160		6160	1. Duke, 2. CMU, 3. Emory, 4. Vanderbilt, 5. University of Pennsylvania, 6. Massachusetts Institute of Technology, 7. Brown University.
Time taken for axes references: Note that axes references are needed for bars in any of the three charts that are considered in the above process, and that fall close to the *x-position* threshold positions of their respective charts.					
Time taken for a single axis reference	Time taken from computed value in Design 5.	520			
Three axis references are required in the left-most chart (Note that Illinois is rejected as a candidate here)	3 * 520	1560			Illinois, Emory, Vanderbilt
Two axis references are required in the left chart	2 * 520	1040			Brown, Vanderbilt
Two axis references are required in the right chart	2 * 520	1040			Vanderbilt, CMU

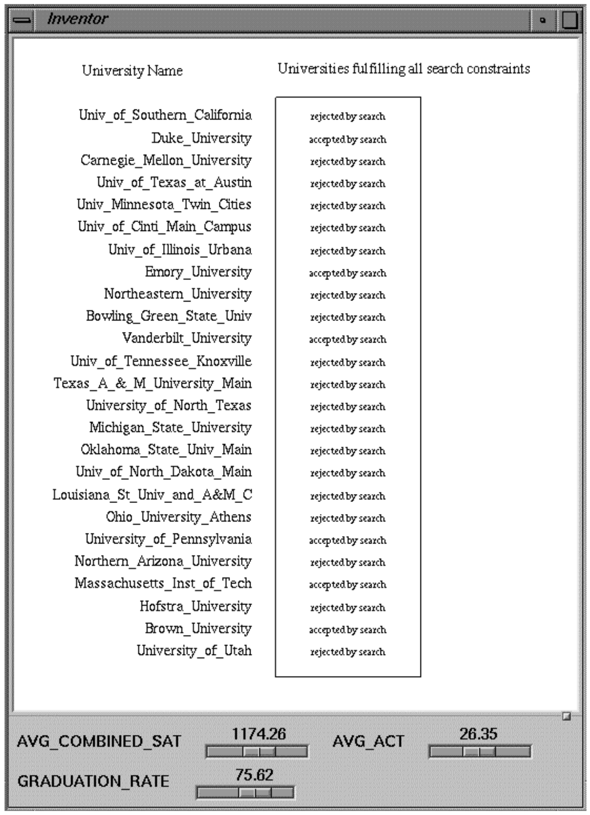
Total time for axes references	1560 + 1040 +1040			3640	
For each university row that fulfills all search conditions, scan left and read the relevant university name. There are 6 universities that pass all search conditions.					
Get university name	Attend get university name	50			
	Initiate eye movement	50			
	Eye movement	30			
	Read university name (3 * 290)	870			
	Verify name	50			
	Sub-total		1050		
Repeat for each university that passed all conditions	6 * 1050			6300	1. Duke, 2. Emory, 3. Vanderbilt, 4. University of Pennsylvania, 5. Massachusetts Institute of Technology, 6. Brown University.
Total time	1410 + 1410 +1410 + 6160 + 3640 + 6300			**20330**	

It is very interesting to note that the total time estimated for this design is very close to that of the previous design. This is because whatever additional time required for processing the additional chart space is made up for in terms of the added accuracy from having all the search attributes mapped to position. As a result we do not need to process the *University of Illinois* element which was wrongly added as a search candidate in all previous perceptual designs that mapped *avg_combined_SAT* to *saturation*.

As with the previous case however, the cost of changing search conditions is high because it is difficult to estimate, for each set of conditions, the number of universities that are added to or rejected from the search. Any change in condition requires new candidate rows processed and as was calculated in the GOMS table above, processing each row takes approximately 840 msec (without including time for axes reference). This is a higher figure than even in the previous design.

E-1.7 Design 7

This design is interesting because it is a purely pre-computed design, that is ordered after a number of different purely perceptual and hybrid perceptual + pre-processed designs. The interface has a set of sliders for setting the three search threshold values and the results of the search are shown as *text labels* next to the university names. Universities that pass the search have the phrase "*accepted in search*" while universities that do not pass the search have the phrase "*rejected in search*" next to them. The reason this design is ranked so low by our designer is because the search results are shown as labels. This may make processing the search results extremely ineffective because now the operation is not clearly pre-attentive and users may need to process each university separately to determine whether it passed the search or not.



The GOMS procedure for this design includes all the slider manipulation operations that are identical to those required in Design 1 and Design 2. Once the search conditions are properly entered in, users must attend to each label to see whether a particular university passed the search or not.

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Total time taken for slider manipulation	Time estimation taken from Design 1			**5370**	approx. 1180 for SAT approx. 26.5 for ACT approx. 76% for graduation rate
Now process each text entry:					
Scan to next entry and determine whether it is an "accepted" or "rejected" concept. Note that since the last two words of the two different phrases ("in search") are identical, it is only necessary to read the first word in the phrase.	Attend determine if next entry passed search	50			
	Initiate eye movement	50			
	Eye movement	30			
	Read first word in phrase	290			
	Verify results	50			
	Sub-total		470		
Repeat for all entries	(25 * 470)			**11750**	

If entry is “accepted”, scan to left and read university name	Attend get university name	50			1. Duke, 2. Emory, 3. Vanderbilt, 4. University of Pennsylvania, 5. Massachusetts Institute of Technology, 6. Brown University.
	Initiate eye movement	50			
	Eye movement	30			
	Read university name (3 * 290)	870			
	Verify name	50			
	Sub-total		1050		
Repeat for each “accepted” university	6 * 1050			**6300**	
Total time	5370 + 11750 + 6300			**23420**	

Note that depending on the phrase used to express the results of the search, it can be possible to view the results pre-attentively through *pattern/shape* matching or through *word length* matching.

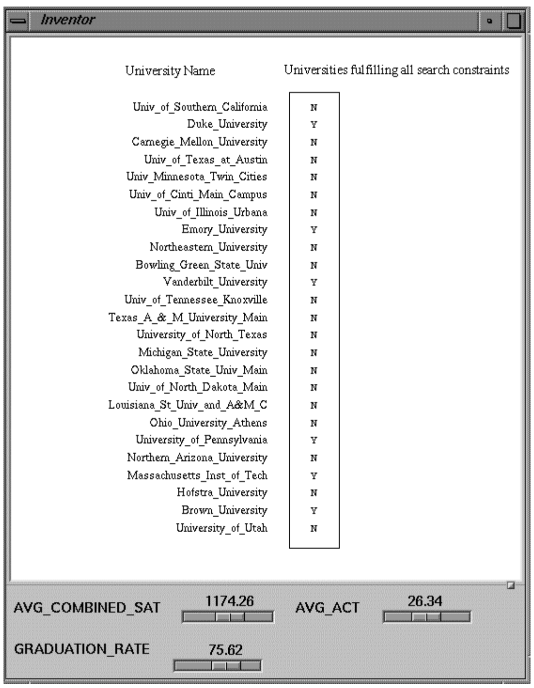


Figure E-1: Representing search results using text labels with “Y” or “N”. Such results can be pre-attentively searched on through pattern matching.

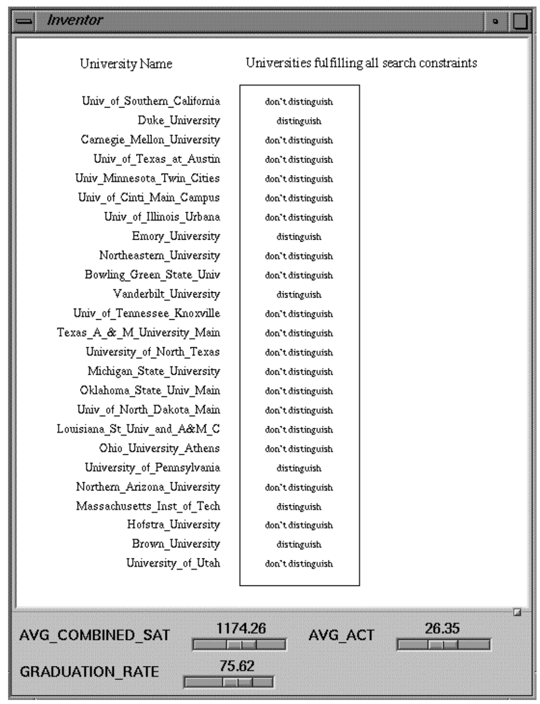


Figure E-2: Representing search results using text labels with “distinguish” or “don’t_distinguish”. Such results can be pre-attentively searched on through length matching.

For example in Figure E-1 the results of the search are shown with a simple “Y” or “N” symbol. In this case, even though technically the results are shown as labels, the labels used are so simple (a single letter) that it is similar to performing shape matching on glyphs which is a pre-attentive operation. Another possibility is to show the results using single words that have clearly different lengths (Figure E-2). In this way, users may also pre-attentively view the search results by filtering based on label length. As a result, this design can, with appropriate result phrases, be elevated to a much higher ranking. Currently however our automatic designer does not have this knowledge encoded within it. It is however not difficult for us to make this addition in the future. Without this additional knowledge, the processing time for changing search conditions is also very significant. For each change, we must peruse through the entire data set to determine the general size of the search concepts and this operation costs an estimated 11750 msec. This time does not include the slider manipulation time which adds another 1790 msec to the operation.

E-1.8 Summary

Figure E-3 shows the estimated GOMS total time for all the designs analyzed in this section. The designs are ordered on the *x-axis* according to the costs assigned by our automatic system and the *y-axis* encodes the estimated total time taken to complete the given search task in msec. From Figure E-3 we see that there is an increasing trend from left to right. Designs that are higher ranked by our designer (to the left) also have lower estimated total times and those that are ranked lower (to the right) have higher estimated total times. Thus the GOMS estimations conform to the orderings assigned by our automatic designer.

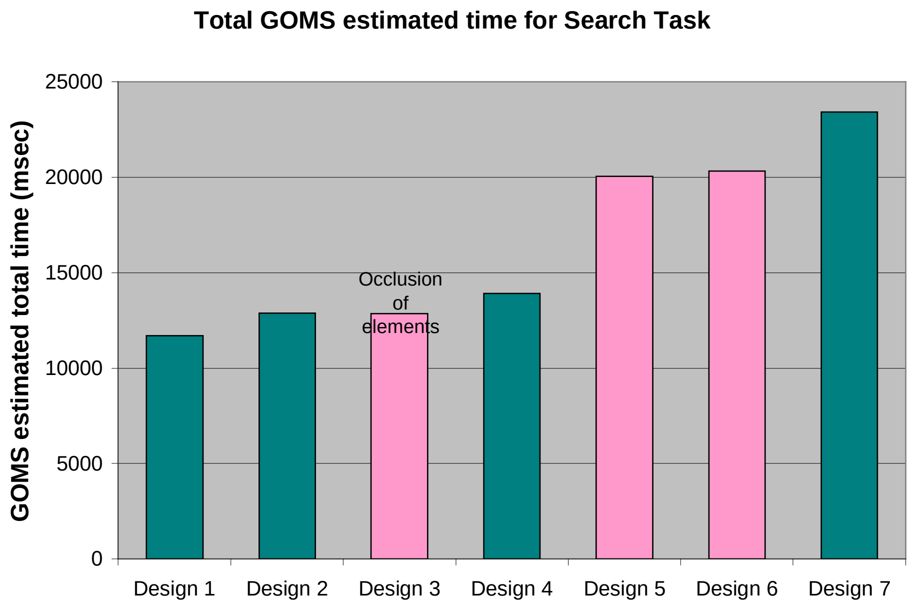


Figure E-3: GOMS estimated total time for Task 1 (search task). The designs are ordered based on increasing cost on the x-axis. The y-axis shows the GOMS estimated total time in msec. All pink bars indicate pure mapping designs (i.e. designs that can be generated with current state of the art automatic design research). All other bars are designs made possible by work outlined in this thesis.

The green bars in Figure E-3 are designs with data transforms techniques (i.e. they are designs that are made possible by work explored in this thesis) and the pink bars are designs that only contain mapping transform techniques (i.e. they are designs that can be generated with current state of the art automatic design technology). We see from Figure E-3 that a significant number of new designs are now possible that previously could not be generated (all of the green bars). More accurately, out of the twenty designs generated for this task, 4 were pure data transform designs, 6 were hybrid designs and 10 were pure mapping designs. Thus 50% of the designs in the top twenty are new designs from the expanded data transform design space. We will see in the next two tasks analyzed that the percentage of new designs made possible by our work is even higher. This is not surprising since search tasks lend themselves very well to perceptual processing (i.e. there are many effective perceptual operators for performing search) but this is not true of computation tasks that are commonly difficult to perform perceptually. While the time gains of the expanded design space is less significant in this example compared to the next computation task, our examples show that the variety added in addition to the interactive components provide very interesting design alternatives to the user.

The contributions of our work can be further seen by comparing Design 1 which is the best design generated by our expanded designer and Design 3 which is the best design that can be generated by previous automatic systems. Design 1 computes the search results for the user based on the three constraints placed on *avg_combined_SAT, avg_ACT,* and *graduation rate*. Only universities that fulfill all three constraints are shown. The advantage of this display is that the filtering significantly reduces clutter and the number of data concepts (i.e. universities) and attributes that need to be shown. As a result, less display space is required for the visualization and there are no readability problems such as overly high object density or occlusion.

Design 3 on the other hand shows all of the data to the user (i.e. *avg_combined_SAT* scores on *saturation*, *avg_ACT* scores on *y-axis*, *graduation rate* on *x-axis* and *university name* on *labels*). Four data attributes are shown here compared to the one (*university name*) in Design 1. To find a university that has high values on the three search properties we need to look for marks that are to the upper right of the display with high saturation. While this is not a difficult perceptual operation, a large problem with this design is the occlusion that occurs among some of the elements which makes it impossible to read some of the relevant university names. The problem worsens for a larger data set. In addition, *saturation*, which is used in Design 3 to encode *avg_combined_SAT* score, cannot show the values with very high accuracy and as a result errors may occur (as we saw in our analysis with *University of Illinois*). In contrast the search results in Design 1 are highly accurate because they are pre-computed by the system. For these reasons, our automatic designer ranked Design 1 over Design 3.

Another interesting aspect of our analysis concerns the estimated added cost for changing the conditions of a search. In most part, the time required is approximately in line with the ranking of our designer as well. This is because the time taken for each condition change is dependent on a user's ability to quickly scan the visualization for a general change in size of the search object set. This operation is more difficult for the later (lower ranked) designs because the search results cannot be viewed pre-attentively unlike the earlier (higher ranked) designs. And it is exactly this pre-attentiveness that cause the designs to perform better in the first place and also score better in our design system. An exception is Design 3, which as we noted can have negligible costs to changing search conditions because no input device manipulation is required and processing of the search results can be performed pre-attentively. Occlusion however is a problem and we discuss in appendix F how further additions to the automatic design process with graphical and rendering transforms may solve such readability problems and further expand the richness and quality of designs that can be automatically generated.

E-2 Task 2: Compute Task

Compute Total Non-salary Benefits Distributed by a Set of Universities

In this section we want to determine the results of a fairly complex computation. The task we chose is to determine the total benefits package given out by a set of universities to their faculty body. "Total benefits" refers to non-salary compensation. The size of the total benefits package given out by a university will indicate the general prosperity and quality of the university in terms of faculty size and general faculty incentives. The university data set used here was taken from *USNews*. This data set has three different faculty groups (*full*, *associate* and *assistant* professor) and two pieces of information for each group, namely the *total salary* and *total compensation* given out to each faculty type. To compute the total benefits given out by any particular university, we need to sum the differences between compensation and salary for each faculty group. The task specification entered into our automatic designer is shown below:

```
(compute   '(VALUE . ADD)
           (compute '(VALUE . SUBTRACT)
                        (lookup  '(OBJECT . nil)
                          '(VALUE . AVG_FULL_COMPENSATION))
                     (lookup '(OBJECT . nil) '(VALUE . AVG_FULL_SALARY))
                     )
            (compute '(VALUE . SUBTRACT)
                     (lookup '(OBJECT . nil)
                             '(VALUE . AVG_ASSOC_COMPENSATION))
                     (lookup '(OBJECT . nil)
                             '(VALUE . AVG_ASSOC_SALARY))
                     )
            (compute '(VALUE . SUBTRACT)
                     (lookup '(OBJECT . nil)
                             '(VALUE . AVG_ASST_COMPENSATION))
                     (lookup '(OBJECT . nil) '(VALUE . AVG_ASST_SALARY))
                     )
            )
```

Task E-2: View the total benefits given to faculty for a set of universities.

From analyzing the data set using the visualization displays generated by our system, we were able to group the universities into four categories based on total benefits: 1) large, well-known state schools; 2) relatively large second tier state schools & large private schools; 3) small, but very prestigious private schools; 4) smaller, less known state and private schools. The large, well-known state schools including *University of Austin Texas, University of Michigan,* and *University of Illinois* are the universities that have the largest total benefits package. This is because they have very large faculty bodies, in addition to providing good benefit incentives to their faculty members. The second group of universities are a mix between the large private schools (*Northeastern University, Southern University of California, MIT,* and *University of Pennsylvania*) and the mid-sized state schools (*Texas A&M University, University of Utah*). Universities in this group have relatively large faculty bodies and good faculty incentives. The third group of schools (*Brown, Duke, Emory, CMU*) are the very prestigious private universities that provide large incentives to their faculty members, but have a much smaller faculty size compared to the prior two groups. Finally, the last group of universities include the smaller private schools (*Hofstra University*) and state schools (*University of North Dakota Main,* and *Northern Arizona University*) with a small faculty and fewer benefits.

Since the task entered into our automatic designer is relatively general with respect to what can be done with the total benefit numbers, there are a set of different tasks that we could perform based on the output designs. For example we could try to categorize the universities based on total benefits as was done above, we could only find the top few universities giving out the most total benefits, we could compare the benefits from particular universities that we are interested in attending, etc. In the interest of simplicity, we chose a fairly small task for our GOMS evaluation:

Find the top four universities with the largest total benefits given to faculty.

As with the previous task we analyze a set of seven output designs, ordered from best (least cost) to worst (highest cost) based on the costs assigned to each design by our automatic system.

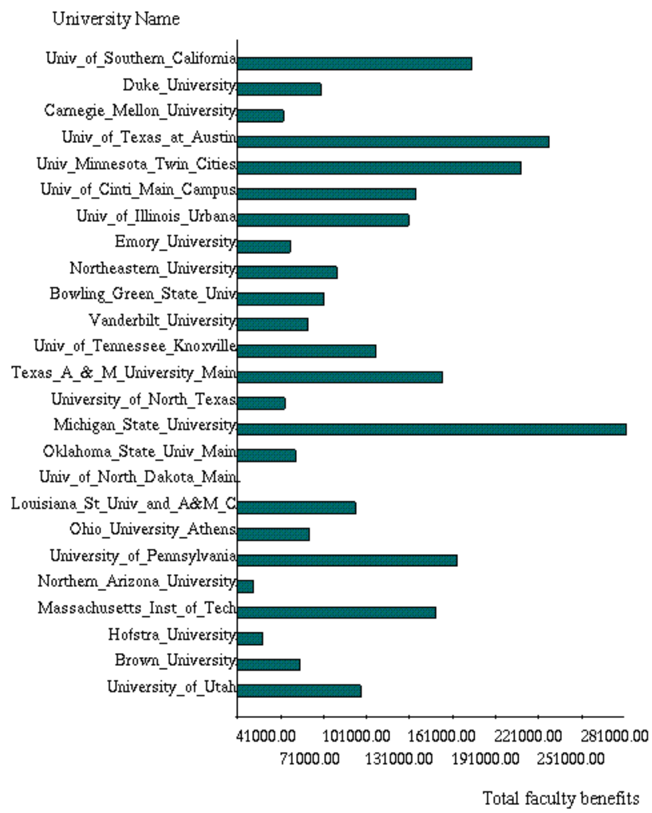
In all following GOMS procedures for this task we assume:

***Baseline Assumptions for Task 2 (Computation Task)*:**

1. Reading a university name is sufficient to commit it to memory until the end of the task, at which time the user is able to recall the top four university names read during the data analysis process. This may not be an accurate assumption because the ability of users to remember the university names depend on the length of the data analysis process. A longer analysis process would degrade short term recall and reduce the probability that users are able to remember the university names read. However, since we make this assumption for all designs, any recall failure can only have a negative bias towards the designs that already have a longer processing time (i.e. make processing time for a less efficient design even longer). This additional time therefore would not affect the initial time ordering/ranking of the evaluated designs.

E-2.1 Design 1

In this design, the entire *total benefits* computation is pre-processed with an internal data visualization technique. I.e. all the computation required is done internally by the system. The computed *total benefit* results for each university are then shown as *horizontal bar lengths* with the *university_names* on the *y-axis*.



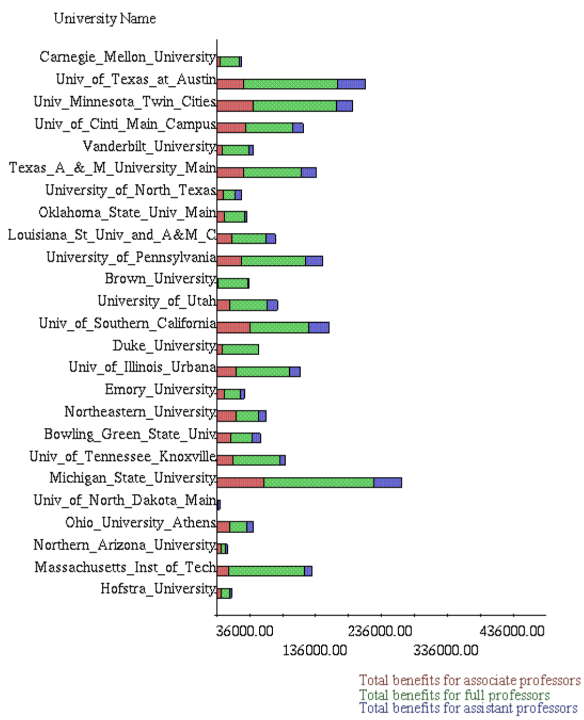
Using this design, we can easily pick the four universities with the highest *total benefits* package by choosing the four longest bars. This is a pre-attentive operation, which means that we do not need to attend to each bar to get the four longest. Instead we can simply scan from top to bottom and pick the bars that we see in order.

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub-total (msec)	Total-time (msec)	Target objects (data or graphics)
Pick next long bar	Attend picking next long bar	50			bar 15, bar 4, bar 5, bar 1
	Initiate eye movement	50			
	Eye movement	30			
	Perceive next longest bar	100			
	Sub-total		230		

Scan to the left of chosen bar to read the university name. (Assume that on average a university name has three words, which is the general case in our data set)	Attend get university name	50			1. Michigan State University, 2. University of Texas at Austin, 3. University of Minnesota Twin Cities, 4. University of Southern California
	Initiate eye movement	50			
	Eye movement	30			
	Read university name (3 word name) = 3 * 290	870			
	Verify name	50			
	Sub-total		1050		
Total time to process each bar	1050 + 230	1280			
Repeat *4* times for each of the top universities	4 * 1280			**5120**	
Total time				**5120**	

This design is very uncluttered and simple with most of the data summarized. We can perform our task very quickly and with high accuracy.

E-2.2 Design 2

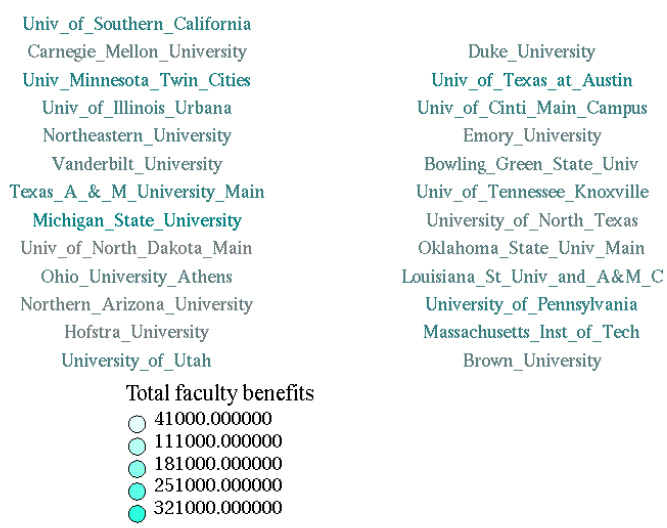


This visualization is a hybrid design. Part of the total benefits computation (i.e. the *subtraction* of *total_salary* from *total_compensation* for each faculty type) is performed internally, and the other part (i.e. the *summation* of *total_benefits* for each faculty type) is left to the user to perform perceptually.

Specifically, the *total_benefits* for each faculty type are shown using the *x-lengths* of three differently colored bars. The green bars represent *total_benefits* for *full professors*, the red bars represent *total_benefits* for *associate professors*, and the purple bars represent *total_benefits* for *assistant professors*. In addition, the bars are stacked to facilitate perceptual computation of the *summation* task. In this design, we can use the same procedure as Design 1. There is more perceptual complexity here compared to Design 1 (i.e. a separate bar is used to represent the *total_benefits* for each faculty type), but the stacking technique is so effective at grouping the objects and helping users perceptually perform the *summation* task, that the additional perceptual load for this design is negligible. For these reasons, our automatic system ranked this design second best. We would like to point out, however, that because of the greater design complexity compared to Design 1, it would initially take the user a longer time to understand this graphic. However, as was noted in our base assumptions, we do not take initial graphic understanding costs into account but instead assume that the user is an expert and already has great familiarity with the visualization designs generated. On the other hand, once understood this design includes more details that may be useful for subsequent tasks.

E-2.3 Design 3

In this design, the entire *total benefits* computation is pre-processed as in Design 1. However unlike Design 1, we map the results to *label saturation* instead of to *x-length* (position) as was done previously. The *labels* themselves show the *university_names*. *Saturation* while pre-attentive like *position*, is significantly less accurate for making quantitative value judgements. This is because our eye can only differentiate relatively **large** differences in *saturation* values. Because of the lower encoding accuracy, this design is ordered below both Design 1 and Design 2.



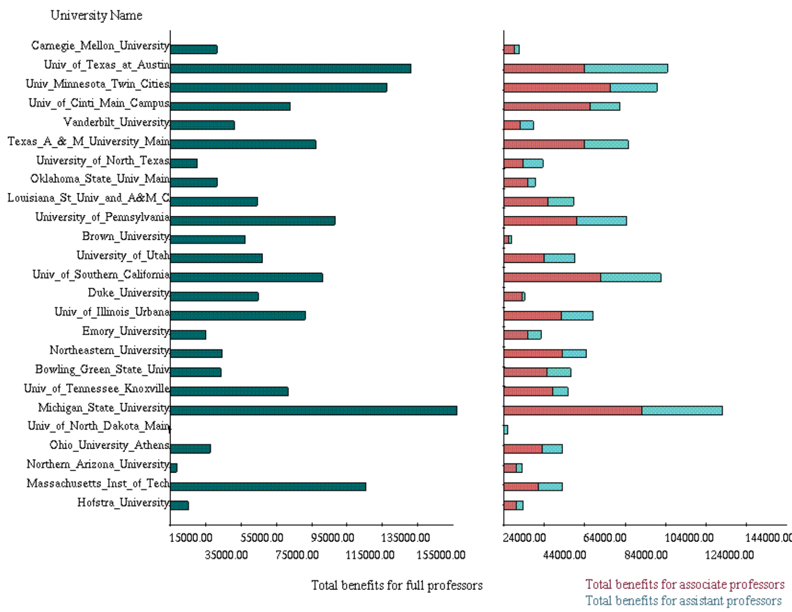
Using this design, we can find the four universities with the highest *total benefits* package by going to the four most highly saturated university names. This is a pre-attentive operation, thus we do not need to attend to each label. However, in the design above, we found ***seven*** universities with high *saturation* values and it is difficult to pick the four most saturated out of this set of seven because of the difficulty in making *saturation* judgements. As a result the GOMS estimated time is higher, because of the additional three university concepts that must be processed. In addition, the error rate is also higher, however that is not taken into account in the GOMS estimation.

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub-total (msec)	Total-time (msec)	Target objects (data or graphics)
Scan and pick next highly saturated label	Attend picking next saturated label	50			Label 1, 3, 7, 8, 15, 23, 24
	Initiate eye movement	50			
	Eye movement	30			
	Perceive saturated label	100			
	Sub-total		230		
Read university name	Attend read university name	50			1. University of Southern California, 2. University of Minnesota Twin Cities, 3. Texas A&M University Main, 4. Michigan State University, 5. University of Texas at Austin, 6. University of Pennsylvania, 7. Massachusetts Institute of Technology
	Read university name (3 word name)	870			
	Verify name	50			
	Sub-total		970		
Time taken to process each saturated label	(230 + 970)	1200			
Repeat process 7 times for each saturated label	7 * 1200			**8400**	
Total time				**8400**	

In the design above, there is an error rate of 43% (3/7). This error rate will increase the closer the *total benefit* numbers are to each other. As the error rate increases, so does the total time taken because there are more universities that need to be processed than necessary for the task. The one advantage of this design over the two previous examples is that no eye movement is necessary to get from the *total benefits* object to the university name because both *university_name* and *total_benefit* is encoded within the same *label* glyph.

E-2.4 Design 4

This design is similar to Design 2 in that it is a hybrid design with the *total_benefits* for each faculty type computed internally, and with the *summation* task left to be performed perceptually by the user. Unlike Design 2 however, this design splits the information over two chart spaces, the left space shows the *total_benefits* for *full professors* on the *x-length*s of the bars and the right chart shows the *total_benefits* for the other two faculty categories. Because there is less integration in the visualization (i.e. two spaces are used instead of just one as in all the previous designs) our designer recognizes that it would be more difficult to perform the *summation* task and as a result this design is given a higher cost (assigned a lower ordering).



To find the four universities with the highest *total benefits* package with this design, we need to scan across the two chart spaces and combine results from both charts. An important issue in determining the GOMS procedure here is the consistency of the axis scales in the two charts. Our system by default scales the axes of the charts according to the *min* and *max* values within each chart. This is so that the encoded values can be determined with the greatest amount of accuracy given the available space. If we were to scale all of the charts to a single consistent scale, some of the bar sets could get significantly dwarfed, thereby making it difficult for users to lookup their values if needed. However for the computation task here it is advantageous to make all of the *x-axis* intervals consistent because that can significantly facilitate the comparison of bar lengths across charts and this will allow for more effective task performance. Thus we start our GOMS procedure for this design by having the user determine the axes intervals in each chart and then re-scaling the axes through interactive operations if the intervals are not identical. In this design, the axis intervals are identical to begin with at 20k, thus no additional re-scaling is required.

Once we have scaled all charts to consistent axis intervals, we start processing the left-most chart based on the assumption that in Western convention, reading is from left to right, top to bottom. In the first chart we attend to, we scan for all the longer bars. Note that in this case, we may need to scan for more than just the four longest bars depending on the corresponding bar lengths on the related chart to the right. In this design we found 9 candidate universities with relatively long bars in the first chart. Out of these nine candidate bars, three are clear winners, *Michigan State University*, *University of Texas at Austin*, and *University of Minnesota Twin Cities*, all have the top longest bars in both charts. The fourth position however is less clear with several close possibilities including *University of Cincinnati Main Campus, Texas A&M university Main*, *University of Pennsylvania*, and *University of Southern California*. There are several ways in which we can process these four candidates. We describe two alternative methods here:

Case 1: *Comparison of bar length differences*

The most effective way, is to get the difference in bar lengths between pairs of candidate bars in both charts and then pick the university with the larger difference. For example, initially, we get the difference in length between the *Texas A&M University Main* bar with the *University of Cincinnati Main Campus* bar on the left chart. We then repeat this for the chart to the right. In this case, both *Texas A&M University Main* bars exceed the *University of Cincinnati Main Campus* bars in length, thus the *Texas A&M University Main* concept is chosen as the current fourth possibility. When processing the *University of Pennsylvania* concept however we find that the *University of Pennsylvania* bar is greater than the *Texas A&M University Main* bar in the left chart, but less in the right chart. In this case, we actually need to compare the two length differences and determine which is larger. For this design and data set the difference is larger in the left chart, thus *University of Pennsylvania* replaces *Texas A&M university Main* as the fourth choice. We then repeat this process for *University of Pennsylvania* and *University of Southern California*. Note that for this method to work, we first need to ensure that the axis intervals in both charts are identical, which we did at the start of the GOMS procedure.

Case 2: Comparison of total_benefit values

Another alternative method for processing the four candidate bars is to read-off their total benefit values from the *x-axis*, mentally combine the benefit values for both charts, and then compare these combined values, picking the highest one. Note that translating an *x-position* into a *total_benefit* value is a somewhat complex operation. This is because there are only labeled tick marks on the *x-axis* every so often and if an *x-position* falls in between two tick marks, we must estimate where it falls within the tick-mark interval before we can convert the *x-position* into a *total_benefit* value. In the GOMS sequence below, we estimate inter-interval positions by dividing up an interval into quarters and then estimating which quarter the *x-position* falls into. This method should be effective in most cases unless dwarfing occurs on the graphical property we are attempting to translate (refer to appendix F on details on dwarfing).

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Compute size of tick-interval on the left chart	Attend scan first x-axis label	50			20k interval
	Initiate eye movement	50			
	Eye movement	30			
	Read number	290			
	Verify number	50			
	Attend scan second x-axis label	50			
	Initiate movement	50			
	Eye movement	30			
	Read number	290			
	Verify number	50			
	Mental subtract number 1 from number 2 (assume 2 significant figures)	250			
	Sub total		1190	**1190**	
Compute size of each tick on the right chart.		1190		**1190**	20k interval
Since both charts have consistent axis intervals (20k) no additional load is required for re-scaling.					
Pick next long bar in left-most chart.	Attend picking next long bar	50			1. Michigan State University, 2. University of Texas at Austin, 3. University of Minnesota Twin Cities, 4. Massachusetts Institute of Technology, 5. University of Pennsylvania, 6. Texas A&M University Main, 7. University of Southern California, 8. University of Cincinnati Main Campus, 9. University of Illinois
	Initiate eye movement	50			
	Eye movement	30			
	Perceive longest bar	100			
	Sub-total		230		
Scan to same row on chart to the right	Attend scan to right chart	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive bar	100			
	Sub-total		230		
Determine whether current bar length is long wrt. other bars in chart. (note that comparison here is pre-attentive)	Attend compare bar length	50			
	Compare	50			
	Verify result	50			
	Sub-total		150		
If bars are long, store bar position in STM as a potential candidate.					
Time taken to process each bar pair	230 + 230 + 150	610			
Repeat for each long bar on first chart	9 * 610			**5490**	9 bars on left chart which are potential candidates.

<table>
<tr><td colspan="6">There are then 4 bars that seem to have approximately the same combined lengths.
1. University of Cincinnati Main Campus,
2. Texas A&M university Main,
3. University of Pennsylvania,
4. University of Southern California
In the next steps we ascertain which of these candidates has the largest total benefit value by comparing bar length differences (case 1) or converting the bar lengths into total benefit values and then comparing those values.</td></tr>
<tr><td colspan="6">CASE 1: Comparing bar length differences</td></tr>
<tr><td rowspan="5">Scan to first unclear bar</td><td>Attend scan to right chart</td><td>50</td><td></td><td></td><td rowspan="5"></td></tr>
<tr><td>Initiate eye movement</td><td>50</td><td></td><td></td></tr>
<tr><td>Eye movement</td><td>30</td><td></td><td></td></tr>
<tr><td>Perceive right edge of bar</td><td>100</td><td></td><td></td></tr>
<tr><td>Sub-total</td><td></td><td>230</td><td></td></tr>
<tr><td rowspan="8">Compare current bar length with previous candidate bar length. Assume that the user can then store this length in STM.</td><td>Attend compare</td><td>50</td><td></td><td></td><td rowspan="8">1. Texas A&M University Main with University of Cincinnati Main Campus,
2. Texas A&M University Main with University of Pennsylvania,
3. University of Pennsylvania with University of Southern California</td></tr>
<tr><td>Attend process previous candidate bar</td><td>50</td><td></td><td></td></tr>
<tr><td>Initiate eye movement</td><td>50</td><td></td><td></td></tr>
<tr><td>Eye movement</td><td>30</td><td></td><td></td></tr>
<tr><td>Perceive right edge of bar</td><td>100</td><td></td><td></td></tr>
<tr><td>Compare</td><td>50</td><td></td><td></td></tr>
<tr><td>Verify results</td><td>50</td><td></td><td></td></tr>
<tr><td>Sub-total</td><td></td><td>380</td><td></td></tr>
<tr><td>Repeat this process in the chart to the right</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="4">Compare the two length differences when the longer bar concept is different for each of the two charts</td><td>Attend compare</td><td>50</td><td></td><td></td><td rowspan="4">1. Texas A&M University Main with University of Pennsylvania,
2. University of Pennsylvania with University of Southern California</td></tr>
<tr><td>Compare</td><td>50</td><td></td><td></td></tr>
<tr><td>Verify results</td><td>50</td><td></td><td></td></tr>
<tr><td>Sub-total</td><td></td><td>150</td><td></td></tr>
<tr><td colspan="6">Total time taken for CASE 1 includes:
1. Time taken to scan for the lengths of all the candidate bars. There are 4 candidates, and two bars per candidate thus total time = 4 * 2 * 230 msec. = 1840 msec.
2. Time taken to compare the current candidate bar with the previous candidate bar. We need to perform this comparison 3 times because there are four candidates (for the first candidate no comparison is required). Since there are two charts, the total time here = 3 * 2 * 380 msec. = 2280 msec.
3. Time taken to compare the two length differences. This operation has to be performed twice for the two cases where the longer bar concept is different in each of the two charts. Total time = 2 * 150 msec. = 300 msec.</td></tr>
<tr><td>Total time for CASE 1</td><td>1190 + 1190 + 5490 + 1840 + 2280 + 300</td><td></td><td></td><td>12290</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

CASE 2: *Comparing the converted total benefit values*					
Scan to first unclear bar	Attend scan to right chart	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive right edge of bar	100			
	Sub-total		230		
Scan to axis to read bar length	Attend determine benefit	50			
	Initiate eye movement	50			
	Eye movement down (assume there is no difficulty with length from actual axis)	30			
	Sub-total		130		
Estimate tick interval position of bar (i.e. where within a tick interval the end point of the bar is situated)	Note bar length pos. on current tick interval	100			
	Determine in quarters the position of the bar length within the current interval tick.	50			
	Sub-total		150		
Convert distance to benefit based on length of 1 axis step	Convert to total benefit value (** refer to division by halving description below). Conversion can be either on: no interval, quarter interval, half interval, or three-quarter internal.	375			
Add with last interval tick	Mental add (2 significant numbers)	250			
Total time taken to lookup the total benefits of the first bar from the axis.	(230 + 130 + 150 + 375 + 250)	1135			
Total time taken to lookup the total benefits of the bar on the right chart		1135			
Add benefits from both charts	Mental add (We assume that the numbers have 2 significant figures based on the axis labels in the design)	250			Texas A&M university Main = approx 100k + 89k = 189k University of Pennsylvania = approx 110k + 89k = 199k; University of Southern California = approx. 105k + 104k = 209k
Total time taken to process each unclear row	Subtotal for unclear bars (1135 + 1135 + 250)	2520			
Repeat for each unclear bar pair	4 * 2520	10080			
Total time for CASE 2	1190 + 1190 + 5490 + 10080			**17950**	

Not surprisingly *case 1* results in a much lower total estimated time than *case 2*. This is because in *case 1*, no mental computations are necessary, nor do we need to convert from *bar length* to *total benefit* values. However, we should point out that in cases where the bar length differences in both charts are very close, it may be difficult to perform accurate comparisons across different charts (i.e. it may not be possible to remember the bar length difference in the first chart with enough accuracy that we can correctly compare it in the second chart.). In this design the bar length differences are large enough that this does not become a problem. If such a problem does arise however, we may need to resort to the *case 2* method.

Note that the further processing required for the four universities in the GOMS algorithm above (as represented by the *case 1*or *case 2* methods) is a result of the lack of integration in the graphic. The additional steps taken show some of the problems associated with dividing up related computation data into multiple different spaces. Interestingly enough whichever method is used does not alter the time ranking of this design. All subsequent designs have higher estimated total time compared to both *case 1* and *case 2* estimation of Design 4.

<u>*Mental Add and Subtract*</u>

In the above GOMS sequence, there were several mental addition and subtraction operations. Each of these operations are performed on numbers with 2 significant figures. The detailed breakdown of these operations are as follows:

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Mental add or subtract	Attend computation	50			
	Add first digit	50			
	Carryover	50			
	Add second digit	50			
	Verify results	50			
	Sub-total		250		

Note that to perform mental computations on numbers with more significant figures, we just add in 100 msec. for each significant figure. Essentially the time taken for a mental add or subtract is 150 + (*n*-1) * 100 where *n* is the number of significant digits in the numbers to be computed.

<u>*Mental Division*</u>

In the GOMS sequence above there is also a mental division operation. Specifically, it occurs when we need to convert a position that falls in between two tick marks back into its *total_benefit* value. Since in this case we only ever divide inter tick mark positions into quarters (i.e. quarter of a tick, half of a tick, or three quarters of a tick), we may simplify the mental division operation by simply halving the interval value as is shown below:

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Mental division by halving	Attend computation	50			
	Halve first digit	50			
	Halve second digit	50			
	Add both numbers	50			
	Verify results	50			
	Sub-total		250		

This allows us to compute the required division when the *x-position* is exactly in the middle of a tick mark. To perform the other cases, we can combine both mental subtraction and "division by halving" as is shown below. The table below lists all the inter-interval cases and the total times required. Since each of the cases is equally likely, we compute the time of any one case by taking the average time of all the cases, i.e. (0 + 250 + 500 + 750) / 4 = 375.

Possible cases	How to convert to total benefit value	Total time (msec)
No interval	No operation	0
Half interval	division by halving	250
Quarter interval	perform division by halving twice	500
Three quarter interval	Perform quarter interval and subtract that from a tick interval	500 + 250

E-2.5 Design 5

This design is a pure internal design, where the computation has been processed internally and only the final results are shown. Unlike previous cases however, where *position* (Design 1) and *saturation* (Design 3) are used, this design uses *labels* to show the total benefit values. Labels are not very effective for this task because they cannot be searched on pre-attentively. As a result, we must attend to each label individually and mentally compare the numbers with each other. This is a cognitively intensive operation and as a result this design is ranked much lower by our automatic designer. In addition, there is also significant state that the user has to keep track of and it is difficult to keep them all in STM (it is difficult to remember numbers). As a result, in our GOMS procedure, the user tracks state information by using their fingers as pointers.

University Name	Total faculty benefits
Univ_of_Southern_California	209920.00
Duke_University	101040.00
Carnegie_Mellon_University	74240.00
Univ_of_Texas_at_Austin	265209.00
Univ_Minnesota_Twin_Cities	245474.00
Univ_of_Cinti_Main_Campus	169492.00
Univ_of_Illinois_Urbana	164436.00
Emory_University	79007.00
Northeastern_University	112416.00
Bowling_Green_State_Univ	103558.00
Vanderbilt_University	91711.00
Univ_of_Tennessee_Knoxville	140956.00
Texas_A_&_M_University_Main	188679.00
University_of_North_Texas	75046.00
Michigan_State_University	320788.00
Oklahoma_State_Univ_Main	82513.00
Univ_of_North_Dakota_Main	40730.00
Louisiana_St_Univ_and_A&M_C	126133.00
Ohio_University_Athens	92852.00
University_of_Pennsylvania	199225.00
Northern_Arizona_University	52341.00
Massachusetts_Inst_of_Tech	183356.00
Hofstra_University	59244.00
Brown_University	86175.00
University_of_Utah	129774.00

In our GOMS procedure the user utilizes four fingers to keep track of the current four highest total benefit numbers. At the start of the procedure, the user picks the first four numbers as the current four highest, thus fingers are placed next to each of these numbers. In addition, the user also determines which finger points to the lowest of these four numbers (i.e. the *lowest max*) and generally what its total benefit value is (e.g. approx 74k). Subsequently, the user processes the following numbers, comparing them to the *lowest max* value. When a total benefit figure is found that is higher than the current *lowest max*, the user updates the current state by changing a finger position from the current *lowest max* position to the new number position. Note that we assume this only requires a finger move. It may sometimes, however, be necessary to move several fingers on these updates to keep our fingers from getting tangled up. However in such cases, the different finger moves would usually happen in tandem (simultaneously) thus the total time for the operation will be approximately equal to that of a single finger move. In addition to changing finger positions, we must also compute a new *lowest max* figure and keep its position and value in short term memory. We continue this process until we have processed all numbers in the table. At the end of the process our four fingers should be pointing to the four universities with the four highest total benefit values.

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Place fingers on first four numbers indicating that they are currently the four largest numbers.	Attend point to first four numbers	50			209920, 101040, 74240, 265209
	Initiate hand move	50			
	Move hand from lap to first four entries on screen	350			
	Sub-total		450	**450**	
Scan next number	Attend read next number	50			
	Initiate eye movement	50			
	Eye movement	30			
	Read number	290			
	Verify number	50			
	Sub-total		470		
Compare if number is more than *lowest max* pointed to by fingers	Attend number > current *lowest max*	50			
	Compare number with *lowest max*	50			
	Verify result	50			
	Sub-total		150		
Minimum total time taken to attend to each number	470 + 150	620			
If number is greater than *lowest max*, point finger to current number and remove finger from the previous *lowest max*.	Attend change finger positions	50			245474 → 74240 (max = 1 step) 169492→101040 (max = 2 steps) 188679 → 169492 (max = 7 steps) 320788 → 188679 (max = 2 steps) *Total move = 12 steps* Average move = 12/4 = 3 steps
	Initiate finger lift	50			
	Lift finger	60			
	Initiate finger move	50			
	Move finger $100 * (\log_2 (1.5/0.5)+.5)$	181			
	Initiate finger drop	50			
	Drop finger	60			
			501		

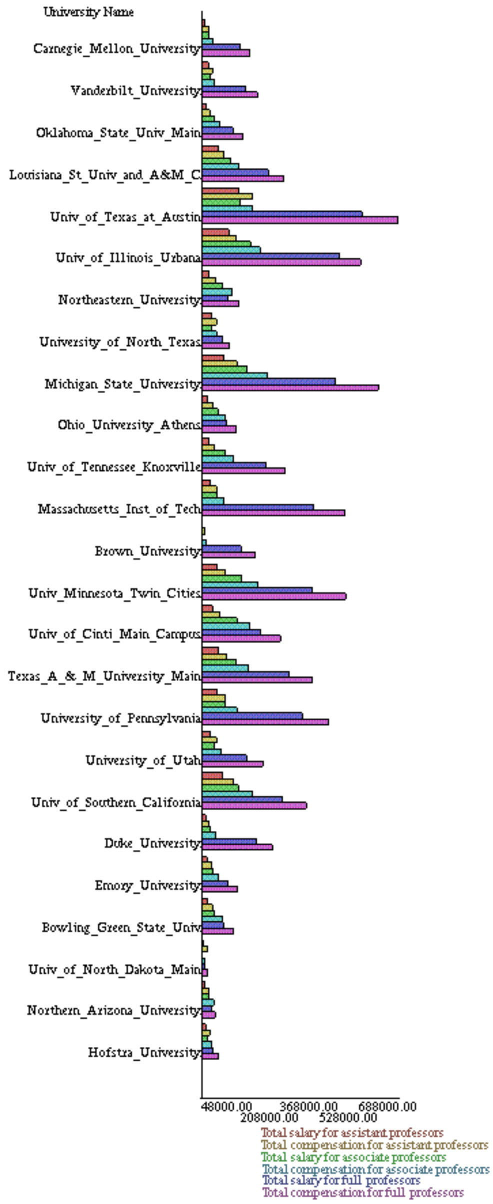
Time taken to move finger in the calculation above we estimate as follows:
We assume that the visualization is enlarged so that each label entry is at least the width of a finger. Or else, finger pointing would be very difficult. Thus,
Height of widest finger = Height of each entry = 0.5 inches
For this particular design and data set, we calculated 4 necessary finger moves, with an average total distance move of 3 steps = 3 * 0.5 inches = 1.5 inches
Thus using Fitts Law, the estimated time for movement is $100 * (\log_2 (1.5/0.5)+.5) = 181$ msec

In the general case, we can assume that any single move will be less than ¼ of the total distance because we are looking for four of the largest numbers, and on average each of these numbers should be spread out one in each quartile. Since there are 25 elements in the data set, each move should on average be 24/4 = 6.25 steps = 3.125 inches
Thus using Fitts Law, the estimated time for movement is $100 * (\log_2 (3.125/0.5)+.5) = 275$ msec

Compare numbers pointed to by four fingers to get the new *lowest max* value and store in STM.					
Compare a single pair of numbers	Compare result on finger 1 and finger 2	50			
	Attend read number on finger 1	50			
	Initiate eye movement	50			
	Eye movement	30			
	Read number 1	290			
	Verify number 1	50			
	Attend read number on finger 2	50			
	Initiate eye movement	50			
	Eye movement	30			
	Read number 2	290			
	Verify number 2	50			
	Attend compare	50			
	Compare #-1 and #-2	50			
	Verify result	50			
	Sub-total		1140		
Up to three pairs of numbers need to be compared to get the *lowest max* value. However note that only two numbers need to be read on the first comparison, on subsequent comparisons, only one number needs to be read because the other number is carried over from the previous computation.	1140 + (2 * 670)	2480			
Total time taken include time to process each number and the additional time of updating and computing new *lowest max* figures as necessary.					
Process all label entries in table. There are 25 numbers -4 of the first numbers = 21 numbers	21 * 620	13020		**13020**	
We need to perform four *lowest max* replacements in this particular visualization	4 * (501 + 2480)	11924		**11924**	
Total time	450 + 13020 + 11924			**25394**	

Note that this procedure requires significant cognitive and STM loads. The additional fatigue caused by these operations (compared to perceptual operations) are not taken into account in the GOMS procedure. This additional fatigue can potentially reduce the length of time a user is able to perform tasks effectively and with low error rates.

E-2.6 Design 6



This design uses only mapping techniques, i.e. all of the task data are mapped to graphics and it is up to the user to perform the entire computation task perceptually. There are six bars shown per university. Each pair of bars represent the *total_compensation* and *total_salary* for each of the three faculty types (*full_professor*, *associate_professor*, and *assistant_professor*). Under normal circumstances, this graphic would be very difficult to use because the computation involved in this task is very complex. Specifically, users would need to perceptually determine the difference in bar lengths between each pair of bars (of which there are 3 per university) and then sum up the bar length differences for each of the three bar pairs. This operation would require significant finger pointing etc, if not help from using measurement tools (i.e. a ruler). Once computed, the total benefit lengths would need to be marked on the paper so that at the end of all the processing, the user can compare all the perceptually computed lengths together. Because of the high computation load and the additional perceptual clutter from having to show six bars, the design is given a significantly high cost by our automatic design system.

Although normally such a design would be close to being unusable, in this example we are able to exploit patterns in the data distribution and realize significant perceptual shortcuts. Specifically, from looking at the bars, it becomes apparent that a significant portion of total benefits given out by a university goes to full professors (represented by the two purple bars). I.e. the difference in length between the two purple bars are comparably much higher than the difference in lengths of the other bar pairs. In addition, it is the universities with medium to long bars (greater total salaries and compensation) also have greater pair-wise bar length differences. In summary to perform our task we only need to process the universities with longer bars and for each of those, we only need to get the difference in lengths between the two purple bars. Since there are more than four possible candidate long bar sets, we must compare each perceptually determined bar length difference with all other previous bar length differences to choose the purple bars with the greatest length disparity. In this design, it is difficult to pre-attentively determine the cases where the length differences between the two purple bars are largest. Thus we process all university concepts with medium to long bars, where there are noticeable differences between the two purple bars (i.e. we need to process more university bars then in the previous designs).

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Scan to first set of longer purple bars	Attend scan	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive bar set	100			
	Sub-total		230		
Get length difference between *full_professor total_compensation* bar and *full_professor total_salary* bar	Attend get length difference	50			
	Perceive difference	100			
	Verify results	50			
	Sub-total		200		
Compare bar length difference with all previous bar length differences and discard the one with the least difference. Here we assume that the user can remember the positions of the current set of four universities with the greatest bar length differences. We make this assumption here, and not in the previous case because here, the user can use the bar lengths as a general tag for the relevant bar objects. And since length searching is pre-attentive, it would not be difficult to get to the desired bars.					
Time taken to make a single comparison between the current bar length difference with one of the current largest bar length differences	Attend compare with current largest bar length difference	50			
	Attend scan to one of the largest difference bars	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive difference	100			
	Attend compare	50			
	Compare	50			
	Verify results	50			
	Sub-total		430		

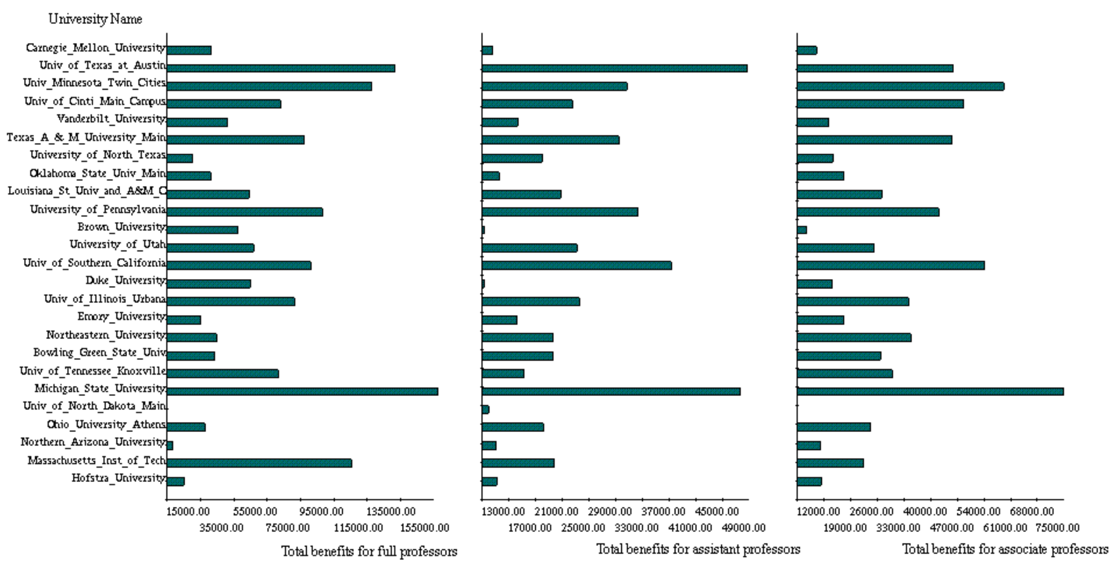
There are 4 comparisons that need to be performed	4 * 430		1720		
Total time for this task includes processing each of the long bar sets, of which there are 13, and performing the comparison with the current 4 largest differences, of which we need to perform (13 – 4) = 9 times. The 13 bars processed include: 1. Louisiana State University and A&M C. 2. University of Texas at Austin, 3. University of Illinois, 4. Michigan State University, 5. University of Tennessee Knoxville, 6. Massachusetts Institute of Technology, 7. University of Minnesota Twin Cities, 8. University of Cincinnati Main Campus, 9. Texas A&M University Main, 10. University of Pennsylvania, 11. University of Utah, 12. University of Southern California, 13. Duke University					
Repeat 13 times for 13 of the tallest bars	13 * (230 + 200)			5590	
Repeat 9 times	9 * 1720			15480	
Total time	5590 + 15480			**21070**	1. University of Texas at Austin, 2. Michigan State University, 3. Massachusetts Institute of Technology, 4. University of Minnesota Twin Cities

It is important to note that in the GOMS procedure above we **do not** take into account the time taken to notice the perceptual patterns in the display nor the time needed to come up with a plan on how to exploit those patterns. We should also recognize that because we are taking significant perceptual shortcuts, we may derive erroneous results in some cases. For example in the design above *Massachusetts Institute of Technology* is chosen instead of *University of Southern California* as one of the universities with the greatest total benefits even though that is incorrect. This is because *Massachusetts Institute of Technology* has very large total benefits given to full professors and very small benefits for all other groups whereas in *University of Southern California* the total benefits are more evenly distributed to all three faculty groups. Because we are only using total benefits for full professors as a yardstick for total benefits for all faculty, errors similar to this will arise for data concepts that do not fit the general perceptual pattern we used to reduce task complexity. Finally, the perceptual shortcuts used in the GOMS procedure here may not be utilized in all cases. We were able to exploit them here only because of the particular data distribution we are examining.

Our automatic designer placed this design below Design 5 even though the GOMS procedures show that this design is more time efficient than Design 5 because our system cannot take into account the specialized perceptual shortcuts that may be exploited in this particular case. Automatically identifying such patterns and taking them into account in the automatic design process, is a difficult but certainly interesting problem that is left for future work.

E-2.7 Design 7

This design is a hybrid data + mapping visualization. The total benefits for each faculty is computed internally, and their results are shown in three separate charts. Each chart represents the total benefit numbers for each faculty group. The GOMS procedure used here is similar to Design 4, except that in Design 4 the information is separated over two charts instead of three. Because the information is not well integrated, the design gets a higher cost compared to all previous visualizations.



The GOMS procedure used for this design is similar to the one used in Design 4, the only difference being that there is an additional chart space that we must process. As was the case previously, we begin by determining the axis intervals for the three charts. Unlike Design 4, the axis ranges for the three charts are different. The left-most chart has an axis interval of 20k, the middle chart has an axis interval of 4k and the right-most chart has an axis interval of 7k. As a result the user must re-scale the *x-axes* of the three charts to equalize their axis intervals. The most efficient and direct interactive method to achieve this would be for the user to type in the desired min-max values in each chart. First of all, the user must determine the min and max values to use. This is achieved by comparing and min and max *x-axis* values in each of the three charts. Once the proper values are determined, users need to select the two min and two max axis values that must be changed and enter in the newly computed min-max values so that all chart ranges are identical. We assume that the numeric keypad is used for the number entries so that once the user's hand is over the keypad no further hand movements are necessary. Between each number entry however users must move their hand between the keypad and the mouse because before an entry can be made the user must first choose which axis number they are altering by clicking on it.

Once the axis intervals are all consistent, we start processing the left-most chart and find all long bars within it. For each long bar we scan to the right and determine whether the respective bars in the other charts are relatively long as well. Many of the operations in the GOMS sequence below utilize operation times that have already been computed in Design 4. As was also the case in Design 4, three universities stand out as having very high total benefit values (*Michigan State University*, *University of Texas Austin*, *University of Minnesota Twin Cities*), however, there are four candidate universities (i.e. four rows) that could fill the fourth position. As in Design 4, we outline the two alternate methods here for determining which of these four candidate rows has the highest total benefit value.

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Compute size of tick interval on the left-most chart.	Total time taken from Design 4	1190		**1190**	20k interval
Compute size of tick interval on the middle chart.		1190		**1190**	4k interval
Compute size of tick interval on the right-most chart.		1190		**1190**	7k interval
Re-scale all axes so that they have similar ranges:					
Read single min value	Attend get min value	50			15000
	Initiate eye movement	50			
	Eye movement	30			
	Read number	290			
	Verify number	50			
	Sub-total		470		
Read next min value		470			min(15000, 13000)
Compare two mins and pick the smaller one. Since the numbers only have 2 or 3 significant figures we assume a simple mental comparison is sufficient here.	Attend compare	50			13000
	Compare numbers	50			
	Compare second digit	50			
	Verify results	50			
	Sub-total		200		
Read next min value		470			12000
Compare min values		200			min(12000, 13000)

Total time to get min value	(470 * 3) + (200 * 2)	1810			
Total time to get max value		1810			
Determine min-max values to use	Attend get min-max-values	50			
	Get min value	1810			12000
	Get max value	1810			155000
	Sub-total		3670	**3670**	

Now we determine the total time for interactively re-scaling the axes.

Click on first min or max value to change	Attend click	50			
	Initiate mouse move	50			
	Mouse move	100			
	Initiate click	50			
	Upstroke	60			
	Down-stroke	60			
	Sub-total		370		
Type in min/max value	Attend input in the min value	50			
	Initiate move hand to numeric keypad	50			
	Move from mouse to numeric keypad	132			
	Attend type value	50			
	Type in value: Number with move required takes 40 + 60 + 60 = 160 msec. (move, upstroke, downstroke) Number with no move required takes 60 + 60 = 120 msec. (upstroke, downstroke) Value 12 000 takes (3 * 160) + (2 * 120) = 720 msec. Value 155 000 takes (3 * 160) + (3 * 120) = 840 msec.	720/ 840			
	Verify results	50			
	Sub-total for min entry		1082		
	Sub-total for max entry		1202		
Move hand back to mouse	Attend move hand to mouse	50			
	Move from numeric keypad to mouse	132			
	Sub-total		182		

Total time to actually enter in the proper min-max values includes:

1. Time taken to enter in the min values = 2 * (370 + 1082 + 182) = 3268 msec.
2. Time taken to enter in the max values = 2 * (370 + 1202 + 182) = 3508 msec.

Total time for entry = 6776 msec.

Now that we have re-scaled the three charts, we proceed to processing the bars.

Pick next long/medium bar on leftmost chart	Attend picking next long bar	50			1. Michigan State University, 2. University of Texas Austin, 3. University of Minnesota Twin Cities, 4. MIT, 5. University of Pennsylvania, 6. Texas A&M University Main, 7. University of Southern California, 8. University of Illinois, 9. University of Cincinnati Main
	Initiate eye movement	50			
	Eye movement	30			
	Perceive next long bar	100			
	Sub-total		230		
Scan to bar on same row to the right chart	Attend scan	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive bar	100			
	Sub-total		230		
Compare bar length with other bars in chart to see if bar has comparatively long length	Attend compare	50			
	Compare	50			
	Verify results	50			
	Sub-total		150		
Scan to bar on the same row to the last chart and perform similar comparisons	230 + 150	380			
Total time for processing each row of bars	(230 + 230 + 150 +380)	990			
Repeat 9 times for 9 longest bars in leftmost chart	(9 * 1090)	8910		**8910**	

There are then 4 bars that seem to have approximately the same combined lengths.

5. University of Cincinnati Main Campus,
6. Texas A&M university Main,
7. University of Pennsylvania,
8. University of Southern California

In the next steps we ascertain which of these candidates has the largest total benefit value by comparing bar length differences (*case 1*) or converting the bar lengths into *total benefit values* and then comparing those values. Note that the following time estimates are all taken from Design 4 and adapted to suit this design which has 3 charts instead of 2.

Total time taken for CASE 1 includes:

1. Time taken to scan for the lengths of all the candidate bars. There are 4 candidates, and three bars per candidate thus total time = 4 * 3 * 230 msec. = 2760 msec.
2. Time taken to compare the current candidate bar with the previous candidate bar. We need to perform this comparison 3 times because there are four candidates (for the first candidate no comparison is required). Since there are three charts, the total time here = 3 * 3 * 380 msec. = 3420 msec.
3. Time taken to compare the three length differences across three charts. This operation has to be performed for all three comparison pairs and two comparisons are needed per pair. Total time = 2 * 3 * 150 msec. = 900 msec.

Total time for task assuming CASE 1 = Time to determine axes intervals + Time to determine min-max values + Time to re-scale axes + Time to process all 9 candidate elements + Time to perform CASE 1 processing = (3 * 1190) + 3670 + 6776 + 8910 + (2760 + 3420 + 900) = **30006 msec**
Total time taken for CASE 2 includes: 1. Time taken to convert the lengths to benefit values and then add those values = left chart conversion (1135) + middle chart conversion and addition (1135 + 250) + right chart conversion and addition (1135 + 250) = 3905 msec. We have to repeat this for each unclear bar, thus total time = 4 * 3905 = 15620 msec.
Total time for task assuming CASE 2 = Time to determine axes intervals + Time to determine min-max values + Time to re-scale axes + Time to process all 9 candidate elements + Time to perform CASE 2 processing = (3 * 1190) + 3670 + 6776 + 8910 + 15620 = **38546 msec**

We want to mention that *case 1* computation for this design can be difficult because unlike Design 4 there are three charts here, thus users must compare length differences between pairs of bars across three charts, and it may be difficult to accurately maintain the bar length differences across this many graphical regions. It is also interesting to note that the time taken to re-scale the axes is fairly large, taking about 10 seconds (10000 msec), and as a result the percentage difference in time between *case 1* and *case 2* is less pronounced here compared to Design 4.

E-2.8 Summary

As can be seen in Figure E-4 the estimated GOMS time for the various designs analyzed for this computation task conform to the orderings assigned by our automatic designer. Also note that a significant number of new designs are now possible that previously could not be generated (all of the green bars). In this compute task there are far fewer mapping designs that make it to the top design spaces compared to the search task. Given our design space limit size of 15000 nodes, we generated 15 designs for this task. Out of these 15 designs 4 were purely data designs, 10 were hybrid data + mapping designs, and only 1 was a pure mapping design. In addition as can be seen from the chart below, many of the most effective designs all contain data transformation techniques (i.e. they are designs that could not previously be generated). This is not surprising, since computation tasks are difficult to perform perceptually, thus there are naturally significant gains to pre-computing the task results. The estimated time taken to process the task using the best design generated by our system (Design 1) compared to previous systems (Design 6) show significant timesavings with the latter time exceeding the former by approximately 4 times. The results from this task show that there are very clear gains that can be realized from considering data transforms in the automatic design process.

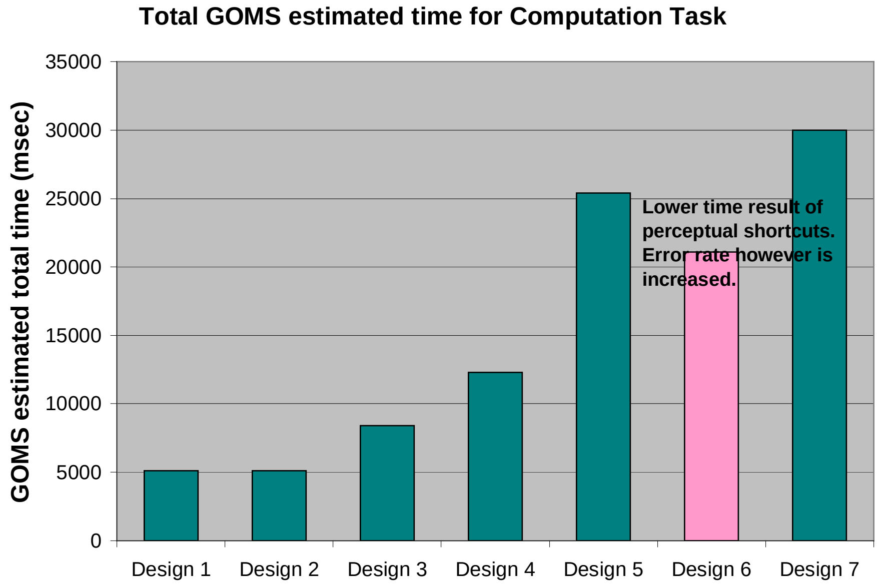


Figure E-4: GOMS estimated total time for Task 2 (computation task). The designs are ordered based on increasing cost on the *x-axis*. The *y-axis* shows the GOMS estimated total time in msec. All pink bars indicate pure mapping designs (i.e. designs that can be generated with current state of the art automatic design research). All other bars are designs made possible by work outlined in this thesis.

The advantages of data transform techniques can be further seen by comparing Design 1 which is the best design generated by our expanded designer and Design 6 which is the best design that can be generated by previous automatic designers. Design 1 is perceptually cleaner than Design 6, showing only the required computed results. Design 6 has six bars per university concept compared to the single bars per concept in Design 1. Our automatic design system chose Design 1 over Design 6 precisely because Design 1 utilizes fewer perceptual components (i.e. has less perceptual complexity), needs less display space, and requires no perceptual load to perform the task computation. On the other hand, in Design 6, the automatic system recognizes that significant perceptual processing is needed (to perform the subtraction and then the summation), in addition to the added complexity of showing 6 bars.

Other designs generated by our system for this compute task (not included in our GOMS analysis) internally process the task results to varying degrees of completeness. For example in Figure E-5, the total benefits are computed for the full and associate professor categories but not for the assistant professor category that must be computed perceptually. The summation task must also be performed perceptually.

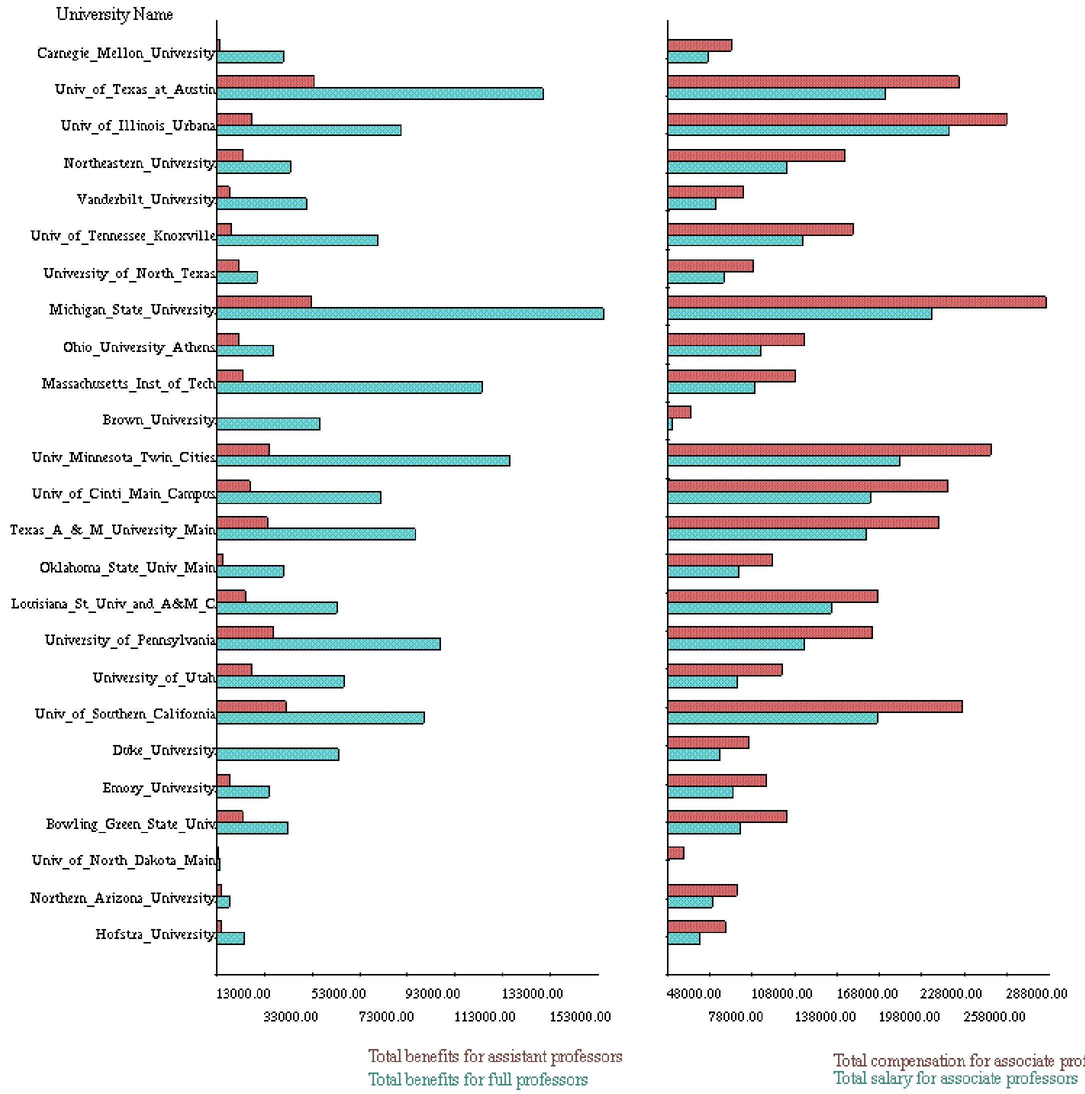


Figure E-5: Hybrid design where total benefits are pre-computed for full and associate professors (left chart) but not for assistance professors (right chart)

In Figure E-6, only the total benefits for full professors are pre-computed and all other operations must be performed perceptually. Note that to facilitate processing, there is a constraint in the designer that ensures all of the data attributes are mapped to the same graphical property class so that the subtraction and addition tasks can be performed more easily. In many of the examples here, the data attributes are all mapped to *x-position*. In chapter V we describe the constraint and cost structure in our automatic design system in greater detail.

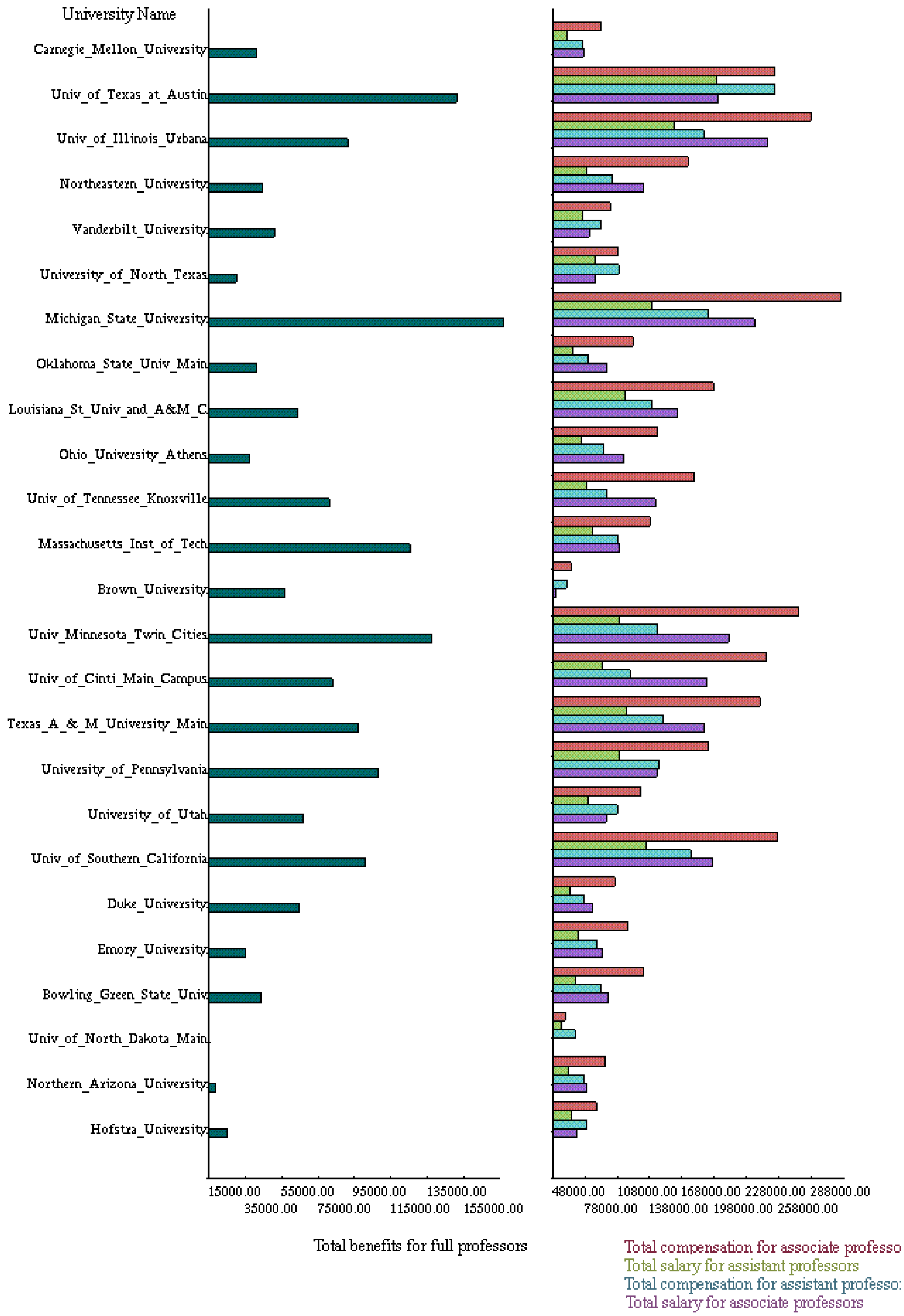


Figure E-6: Hybrid design where total benefits are pre-computed for only full professors (left chart) and the computation for associate and assistance professors (right chart) as well as the final summation task must be performed perceptually.

One may notice that if we had sorted or ordered the bars in the designs presented in this section based on *total_benefit* values, we would have been able to perform the given search task that we had set out for ourselves very easily. Specifically, we may augment our task specification with an additional *sort* task that encapsulates the *total_benefits* computation task as is shown below:

```
(compute   '(VALUE . SORT)
    (compute   '(VALUE . ADD)
            (compute '(VALUE . SUBTRACT)
                          (lookup  '(OBJECT . nil)
                            '(VALUE . AVG_FULL_COMPENSATION))
                      (lookup '(OBJECT . nil) '(VALUE . AVG_FULL_SALARY))
                      )
            (compute '(VALUE . SUBTRACT)
                      (lookup '(OBJECT . nil)
                               '(VALUE . AVG_ASSOC_COMPENSATION))
                      (lookup '(OBJECT . nil)
                               '(VALUE . AVG_ASSOC_SALARY))
                      )
            (compute '(VALUE . SUBTRACT)
                      (lookup '(OBJECT . nil)
                               '(VALUE . AVG_ASST_COMPENSATION))
                      (lookup '(OBJECT . nil) '(VALUE . AVG_ASST_SALARY))
                      )
            )
    )
```

Task E-3: Sort a set of universities by their total benefits

This addition causes some new designs to be generated, i.e. designs where the *sort* task is computed internally and only its results are shown (i.e. the *total_benefit* values are culled out). The designs that we analyzed and discussed above, however, will still all be candidates because they support the expanded task as well. It is important to note that this *sort* addition will only help us when we are searching for elements based on their ranking. If we were instead interested in comparing the relative total benefit values of two or more universities or in looking up the *total_benefit* values of particular universities, the *sort* task would be of no help. We expect that such additions or refinements to tasks will be a very common user operation. Users often do not have a clear enough idea of the tasks they want to perform that they are able to zero in on the correct and complete specification on first try. The same goes for communication. Users are rarely able to describe the exact information needed at the beginning of an information session. Realistically users will arrive at the information after having a series of conversations with the system where refinements and changes are made to the initial request. We discuss task refinement and its associated issues in section E-4.

E-3 Task 3: Comparison Task + Simple Computation

Evaluating the Relationship between State Size and Voting Results

In this final section we wanted to explore a comparison task. Upon further analysis, however, we found that a comparison task alone does not result in many more interesting designs or design issues over what has been previously discussed in the first two tasks. To make this example a bit more interesting and to start exploring task combinations, we included a simple computation task in addition to the comparison. In this example we compare the number of votes for three different political parties (*democratic, republican,* and *other*) in different states so that we may see how they rank with respect to each other. In addition to the ranking, we also want to see the total number of votes in each state so that we may determine the importance of a given victory. Larger states presumably carry a greater political gain.
The task specification entered into our automatic designer is shown below:

```
 (compute ’(VALUE . SORT)
         (lookup ’(OBJECT . NIL) ’(VALUE . DEMOCRATIC))
         (lookup ’(OBJECT . NIL) ’(VALUE . REPUBLICAN))
         (lookup ’(OBJECT . NIL) ’(VALUE . OTHER)))

(compute ’(VALUE . ADD)
         (lookup ’(OBJECT . NIL) ’(VALUE . DEMOCRATIC))
         (lookup ’(OBJECT . NIL) ’(VALUE . REPUBLICAN))
         (lookup ’(OBJECT . NIL) ’(VALUE . OTHER)))
```

Compare the number of votes received by each of three political parties and determine their ranking. Also determine the total number of votes per state to ascertain the significance of any particular political victory.

From analyzing the displays generated by our system we were able to categorize the states into three groups. The larger states are generally carried by the *democratic* party, the medium to small sized states are generally carried by the *republican party* and finally the smallest states have an about equal mix between *republican* and *democratic* victories. The *other* party did not gain a victory in any of the states and in most cases ranked last. There is only one exception in “AK” where the *other* party ranked above the *democratic* party, and the *republican* party came in first.

As with the previous cases there are several different types of tasks that we may perform based on the general directions entered into our designer. We may for example be interested in the number of *democratic* victories compared to *republican* victories, we may be interested in finding abnormalities in ranking such as in the case of “AK” described above, we may be interested in examining the ranking of particular states that are of interest, etc. In the GOMS analysis below we wanted to examine a task that utilized both the *ranking* and *total number of votes* information in tandem, so we chose the task of determining whether there is a relationship between *ranking* and *total number of votes*.

Ascertain whether there is a relationship between the party rankings and the total number of votes in a particular state.

The following assumptions are made for all the GOMS procedures of this task:

***Baseline Assumptions for Task 3 (Cmparison Task)*:**

1. In this task there are three parties and in our task specification we stated our interest to see their respective rankings. Thus in all the graphics generated, all three parties are represented. In all the graphics generated, it is necessary to differentiate each of the parties so that we may identify a particular rank with a party name. Sometimes the encoding scheme used for the parties may be different from one design to the next. This is because our designer assumes each output design is separate and does not try to use similar encoding schemes across multiple alternative designs. In all of the GOMS procedures for this task we assume that the user is familiar with any encoding scheme used for the three parties. This assumption is reasonable since we are modeling an expert user. In addition, adding such information into the GOMS estimation does not change the rankings of the different designs because it adds a commensurate amount of time to each and every design.

E-3.1 Design 1

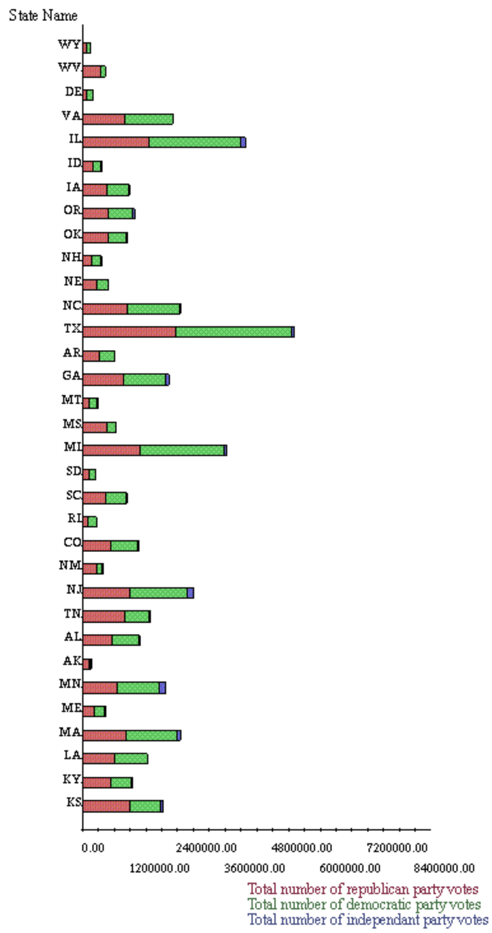
The first design generated is a purely computed design. Both *total number of votes* and *party rankings* are determined by the system and only the computed results are shown to the user. In the design below, each state is represented by a cluster of three marks. Each mark represents a different party, *red* represents the *democratic* party, *green* the *republican* party, and *purple* the *other* party. *Total number of votes* is mapped to the *y-axis* and the marks are ordered from left to right depending on the computed rankings of the different parties. The left-most mark is the party with the most votes and the right-most mark is the party with the least votes. The *x-axis* of this design does not encode any data. Clusters are however shifted to the right to improve layout and avoid occlusion problems. This graphic is especially efficient for performing the desired task because we can pre-attentively see the relationship between *ranking* and *total number of votes* within each *x-column* without having to attend to each *state* concept. Clusters high on the *y-axis* start off with a *red* dot (i.e. larger *states* are won by the *democratic* party), clusters in the mid to lower portion of the *y-axis* start off with a *green* dot (i.e. mid to small *states* are won by the *republican* party), and finally clusters at the bottom-most areas start off with either a *red* or *green* dot (i.e. the very small states are won by a mix of both *democratic* and *republican* parties).

The GOMS procedure for this design is simple, requiring three perceptual groupings at the top, mid-to-low, and very low portions of the *y-axis*. Since the patterns can be viewed pre-attentively users can deduce the relevant information with just three perceptual scans.

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Scan top of *y-axis* to see which color mark is to the right of the clusters there.	Attend scan top	50			*Red* or *democratic* party to the left.
	Initiate eye movement	50			
	Eye movement	30			
	Perceive information	100			
	Verify results	50			
	Sub-total		280		
This is pre-attentive for each *x-column* and there are 4 *x-columns* at the top so 4 * 280 msec are needed.	4 * 280			1120	
Scan to middle of *y-axis* to see which color mark is to the right of the clusters there.	This operation is similar to the one above, and there are 4 *x-columns* in the middle area so 4 * 280 msec are needed,			1120	*Green* or *republican* party to the left.
Scan to the lower y-axis to see which color mark is to the right of the clusters there.	This operation is similar to the one above, and there are 6 *x-columns* in the middle area so 6 * 280 msec are needed,			1680	*Purple* or *other* party to the left.
Total time				**3920**	

As can be seen from the GOMS sequence the design is simple and allows significant time savings because most of the data has been summarized due to the internal computation. In addition the visualization design allows the relationship between *total number of votes* and *party ranking* to be viewed pre-attentively for each *x-column*, requiring only 14 perceptual processing steps. Note however that the visual design generated for this graphic is not a standard representation and as such the learning curve of using the design may be relatively high. However, since we assume expert users in our GOMS evaluation this issue has no effect on the total time. Also note that ranking of the three different parties is from left to right. This is based on the Western convention of reading which is from left to right. Users from other cultures may misinterpret this encoding scheme.

E-3.2 Design 2



We found it quite interesting and somewhat surprising that the second design generated in this task is a purely perceptual design. This is because the task entered is fairly complex one including a computation task which usually does not fare well with perceptual designs. However because the two tasks here (i.e. *summation* and *sort*) operate on the same set of data attributes (i.e. *number of votes for democratic, republican* and *other* party) a perceptual design that is effectively executed, can show the required data in an integrated fashion that supports both tasks as is the case below. In this design, the *number of votes* for each party type is mapped to the *x-length* of a different colored bar. The *green* bars show the number of votes for the *democratic* party, the *red* bars show the number of votes for the *republican* party, and the *purple* bars show the number of votes for the *other* party. By mapping the *number of votes* information to the same graphical property class (i.e. *x-length*) we are able to facilitate both comparison and computation. By further stacking the bars, we greatly simplify performing the *summation* task perceptually. This design, however, is still only ordered second because more information must be perceptually mapped here including *state-name* and *number of votes* for each party (four attributes). This is in contrast to the two attributes (*total number of votes* and *party ranking*) that are mapped in the first design. In addition, despite the perceptual mapping choices used in the design that help facilitate the task, there is still some perceptual load that need to be expended to determine the task results especially for the comparison or ranking task which can be somewhat difficult because the bars are stacked.

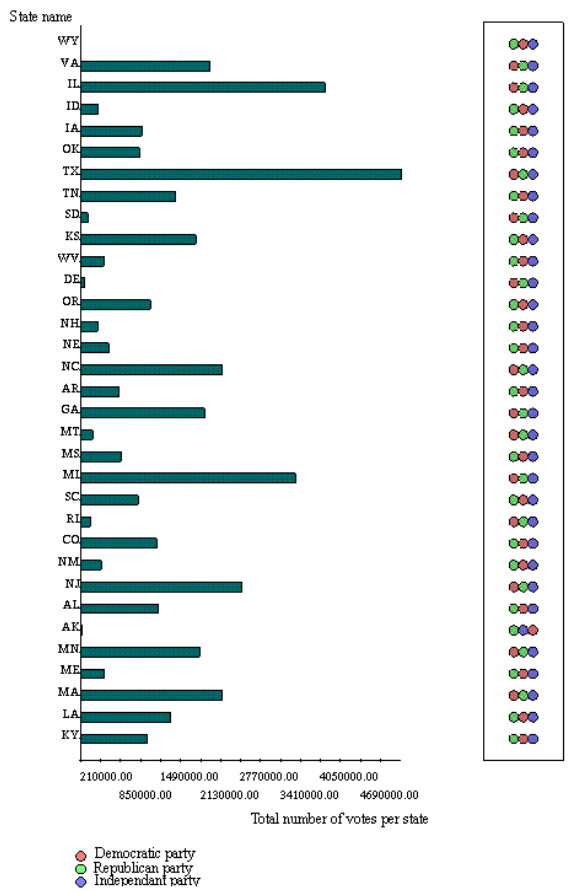
The GOMS procedure for this design begins with a scan for the longer bars in the design (i.e. the larger *states*) and for each long bar, we determine which color bar has the greatest length (i.e. which party is ranked first). Note that because of the data distribution, specifically the *number of votes* for the *democratic* and *republican* parties are usually very close in value resulting in almost equal bar lengths, it can sometimes be difficult to determine the ranking of the two parties. For better data distributions, it is possible to determine the rankings for groups of bars pre-attentively. However, in this case we at least need to attend to each bar and compare the *red* with the *green* lengths. In some states, where the values are especially close, we may not even be able to determine the ranking results accurately.

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Scan for longest bars (pre-attentive) (then subsequently mid, and then short bars)	Attend scan	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive results	100			
			230		
For each bar cluster, determine which color segment is the longest. This may be a pre-attentive operation but because the *republican* and *democratic* votes are so close in value, it is difficult to make pre-attentive comparisons. As a result we assume that each bar must be attended to individually here.					
Compare the length of one colored bar with another	Attend compare	50			
	Initiate eye movement	50			
	Eye movement	30			
	Compare bar lengths	50			
	Verify results	50			
			230		
Note that since the *other* party usually has a very low number of votes we assume that the user is able to pre-attentively note this pattern initially, and thus only one pair of bar comparison is needed for each state. Thus the comparison cost for each state is 230 msec.					

<table>
<tr><td colspan="6">In this procedure we need to scan for long, mid, and short bars, thus the estimated time for that is:
3 * 230 msec.
In each of these operations we need to attend each bar in the set and determine the ranking of the parties. Since there are a total of 33 bars, the total time for determine the rankings of all bars in all three long, mid, and short bar categories is:
33 * 230 msec</td></tr>
<tr><td>Total time</td><td>(3 * 230) + (33 * 230)</td><td></td><td></td><td>7820</td><td></td></tr>
</table>

We want to point out that in the ideal case, where the data distribution allows pre-attentive (simultaneously) ranking of a set of bars, the total estimated time taken for this design would be significantly reduced at: (3 * 280) msec for perceiving each long, mid, and short bar category, and then another (3 * 280) msec for pre-attentively ranking each set of bars. The total time in this case is therefore only 1680 msec., which is much closer to the time taken in the first design.

E-3.3 Design 3



This design is a pure pre-computed design similar to Design 1. However, the data is separated over two different spaces and aligned based on *state-name*. The *total number of votes* within a state is mapped to the *x-length*s of bars in the left chart and the *party rankings* are shown with a three mark cluster in the *table* to the right. As in Design 1 the marks are ordered from left to right (most votes on the left, least votes to the right) and each party is represented by a different colored mark (*red* = *democratic, green* = *republican, purple* = *other*).

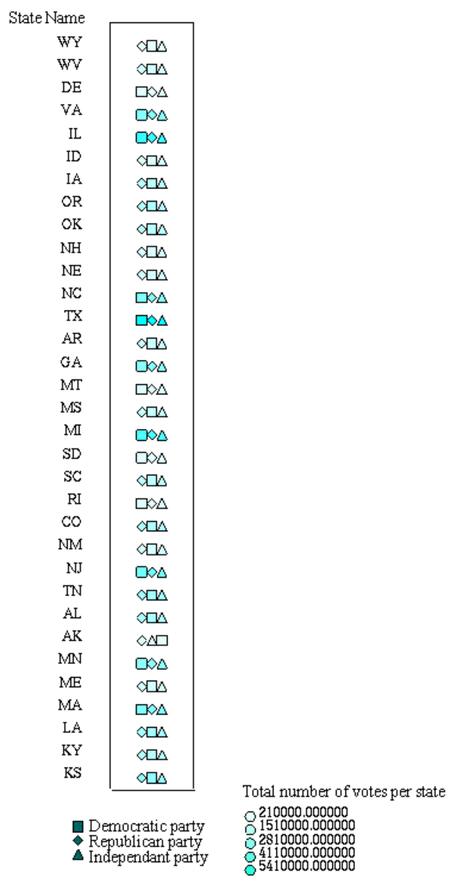
To solve the task using this design we first look for the states with a large number of votes on the left chart and then scan over to the right chart to lookup their rankings. We repeat these steps for the mid-sized and small states. Note that the ranking lookups are not pre-attentive and requires each of the relevant states in the large, mid, or small-sized categories to be visited.

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Scan to next longest bar	Attend scan to next bar	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive bar	100			
	Sub-total		230		
Scan to the right and lookup ranking information and verify which party is ranked first.	Attend scan to right	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive marks	100			
	Verify results	50			
	Sub-total		280		
Total time to process a single row	230 + 280	510			
Total time	33 * 510			**16830**	

Note that the time taken to process this design can be reduced very significantly if we were to sort the elements on the *y-axis* based on *total number of votes* in each state. In doing this the design becomes very similar to Design 1 where the *y-axis* is used to encode *total number of votes* and as a result we are able to view the ranking results pre-attentively unlike the individual attention required in the GOMS sequence above. In this case, this design would take the same amount of time to use compared to Design 1. To automatically generate such a design however, requires the automatic system to recognize that the *y-axis* is used to encode and un-ordered (nominal) attribute, and since *position* allows ordering information to be shown, it is possible to add more information into the display without increasing perceptual complexity. Thus the designer must automatically transform a *nominal* data attribute into an *ordinal* attribute by adding ranking information into the element values. Such data characterization altering operations are beyond the scope of our work. Secondly, as was pointed out in the *summary* section of the previous task, this sorting operation requires an expansion to the current task specification that may cloud the main task(s) and associated design issues we are interested in exploring. We will however discuss the sort alternative in greater detail in section E-4.

E-3.4 Design 4

This design pre-computes the two task results as in Design 3, however, less effective graphical properties are used and it is for this reason that the design is ranked lower by our system. Specifically, *saturation* is used to show *total number of votes*, *shape* is used to show the three different parties, and *x-ordering* in each cluster is used to show ranking. Even though the previous design is less integrated, the results can be viewed with much greater accuracy and the additional time needed for the scan between the two spaces is not too significant. As with the previous case, ordering the *y-axis* based on *total number of votes* (i.e. *saturation*) can improve our ability to perform the task. We discuss this sorting issue in greater detail in section E-4.



The GOMS procedure for this graphic is very similar to that of the previous design. The difference is that instead of choosing elements based on *bar lengths*, here we are choosing elements based on their *saturation*. Initially we find all the saturated clusters, we then examine each cluster to determine how the parties ranked. We then repeat this process for the mid and low saturated clusters.

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Scan to next most saturated cluster.	Attend scan	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive cluster	100			
	Sub-total		230		
Determine and verify shape of leftmost mark in cluster. (here we assume that the expert user does not need to refer to legend but are able to associate shape with party because of regular use of such graphics)	Attend get shape	50			
	Perceive shape	100			
	Verify results	50			
	Sub-total		200		
Total time	33 * (230 + 200)			**14190**	

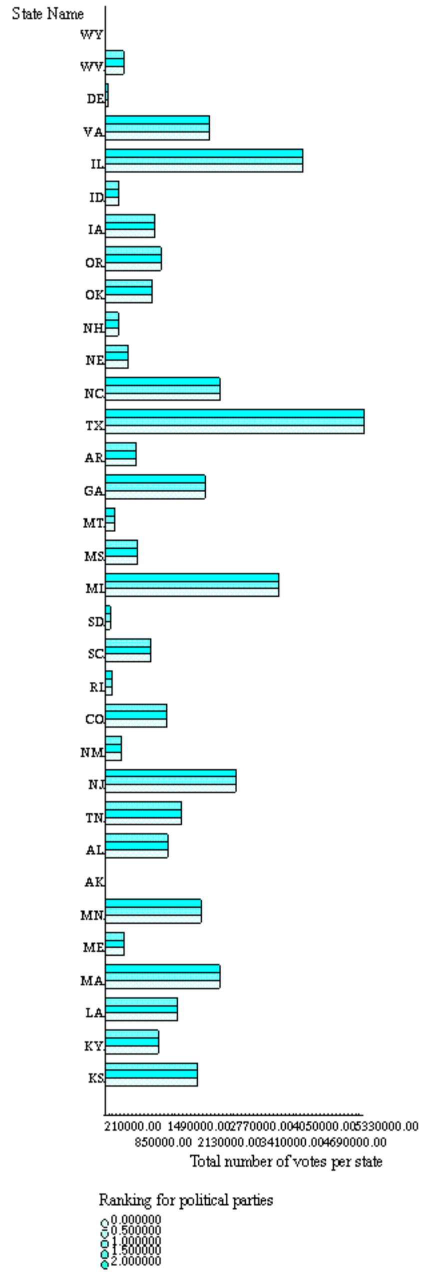
Note that the time taken for this task is slightly less than that of the previous design (2.5 seconds less). This is because the previous design requires additional eye-movements to scan from the left chart to the right table. However, the accuracy with which the task can be performed is not captured in the GOMS time estimation. However, since accuracy in grouping is less important to the task here, it has less of an effect than in the previous tasks where inaccuracies resulted in additional processing and inclusion of concepts that do not fulfill task requirements.

E-3.5 Design 5

This design is also fully pre-computed, it is interesting however, for the way in which it shows the computed results. *State* is shown on the *y-axis* and each *state* is associated with a cluster of three bars, one for each party starting with *democratic* on the top, followed by *republican* in the middle and *other* at the bottom. The *x-length* of all the bars encode the *total number of votes* within the state. The saturation of each bar represents the current ranking of the bar. Highly saturated bars are ranked higher compared to lower saturated bars. A primary weakness of this graphic is that we need to process each set of bars separately to get the most saturated bar (most highly ranked party) and then there is an additional step where we determine the position of that bar and associate its position to the relevant party. This is in contrast to the previous pre-computed designs where users can simply get to the highest ranked party by scanning to the left-most mark without having to even process any of the other marks in the cluster. Because of this the design is ranked lower by our system. Note that unlike the previous designs, even if we ordered the states on the *y-axis* based on *number of votes* we would still need to attend to each bar cluster individually.

In addition, unlike the *mark clusters* this graphic requires much more display area, due to the fact that a bar cluster expands the width needed to show each *state* concept three-fold. Also note that currently it is difficult to get the saturation values for bar clusters with small length (i.e. it is difficult to determine the party rankings for states with comparatively few total number of votes). For example consider “DE” and

“AK” where it is not possible to perceive the saturation values given the current *x-axis* scale. This problem however is not serious as it can be easily fixed by lowering the *minimum* range on the *x-axis*.



The GOMS procedure for this design begins with a search for all long bars (i.e. states with a large number of votes). For each of these bar clusters, we locate the most saturated object and determine whether it is on the top, middle, or lower position of the cluster. This will tell us which party was ranked first in the state. Subsequently we repeat this process for the mid-sized and short bar groups.

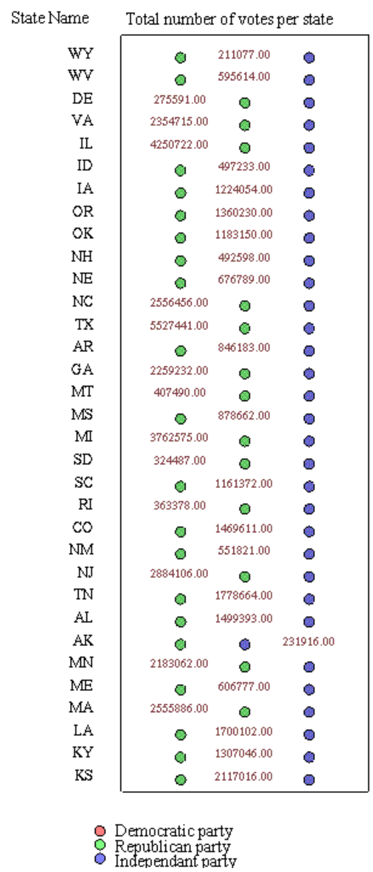
General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Scan to next longest bar	Attend scan to next longest	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive bar set	100			
	Sub-total		230		
Determine most saturated bar	Attend get most saturated	50			
	Compare all saturations and get most saturated bar (pre-attentive)	50			
	Verify results	50			
	Sub-total		150		
Get most saturated bar position within cluster	Attend get most saturated bar cluster position	50			
	Count Since the bar could be on the top, middle, or bottom positions the count could be anywhere from 1 to 3. On average we therefore assume a count of 2. Thus total count time is (2 * 50 msec) = 100 msec	100			
	Verify results	50			
	Sub-total		200		
Total time to process a single bar cluster	230 + 150 + 200	580			
Total time	33 * 580			**19140**	

As was expected, the time taken for this design is greater than other previous similar designs (i.e. Design 3 and Design 4). This is because additional steps are needed to compare the saturation values within each cluster and to subsequently identify the party associated with the most saturated bar. I.e. mapping *party ranking* to *saturation* is less efficient than mapping *party ranking* to *position*. The general structure of this design is also less traditional, thus learning time may also be greater.

E-3.6 Design 6

This design is also fully pre-computed, but it uses *labels* to represent the *total number of votes* in each state. In the design *state* is encoded on the *y-axis* and each state has a cluster of two marks and a label associated with it. The *red* label represents the *democratic* party, the *green* mark represents the *republican* party and the *purple* mark represents the *other party*. Ranking is shown based on the ordering of these three objects. The left-most object is the most highly ranked party, followed by the middle object, and finally the right-most object represents the party with the least ranking. While this label encoding provides accurate *total number of votes* figures, it is a very ineffective graphical property for grouping the states based on total votes because unlike all of the previous designs we need to attend to the label of each *state* concept.

Even worse yet, we need to first scan through all of the labels to get the range of values before we can begin the categorization process.



At the start of this GOMS procedure, we quickly scan through all of the labels to get the *min* and *max total number of votes*. During this process we assume that the working *min* and *max* numbers can be kept in STM (short term memory). Since we do not need to get exact *min* and *max* values here, it is reasonable to assume that users will round up or down the *total number of votes* figures to fewer significant digits to simplify comparisons as well as storage and recall from STM. Here we assume a general rounding to a single significant digit. Once we get the *min* and *max* numbers we go through the set of states again and only process those *total number of votes* entries that fall within the upper range of the *min-max* values extracted. When we are done, we repeat this process for the mid and lower range values. Thus we end up having to go through the table four times, once to determine the *min-max* values, and the next three times to process our three desired value groups namely high values, mid values and low values.

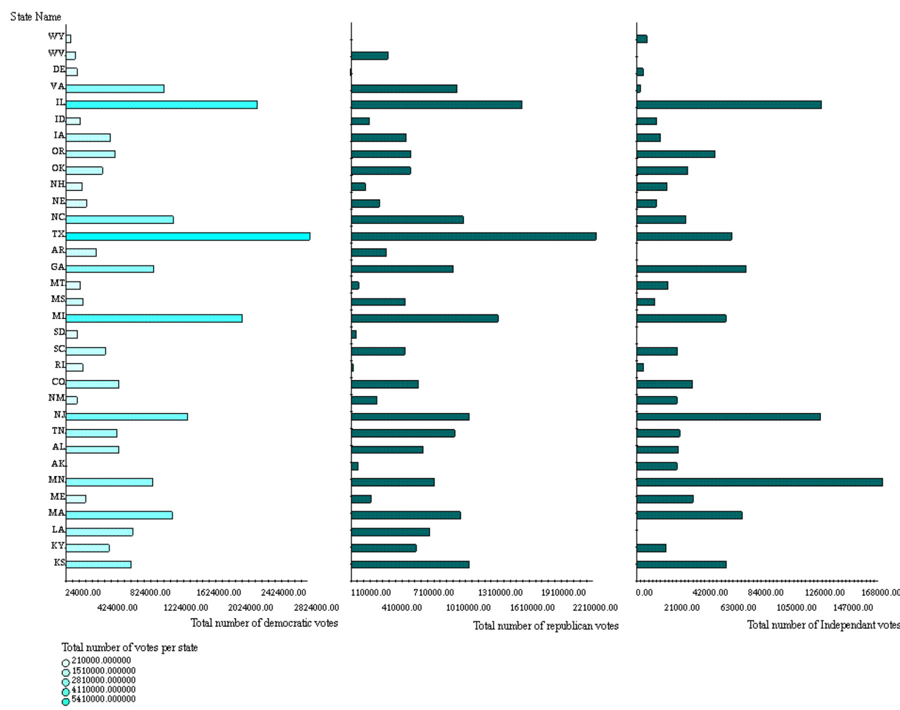
General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Scan down all numbers and get general *min* and *max* values to determine the range for the total benefit values.					
Time taken to process a single element. We assume that the *min* and *max* values can be stored in STM.	Attend check element to see if it is min or max candidate	50			*max =* *200k → 500k → 2mil → 4 mil → 5mil* *min = 200k*
	Initiate eye movement	50			
	Eye movement	30			
	Count digits in number in chunks of 3 digits to get general scale. Most of the numbers can be calculated in 2 chunks thus total time to count is: (2 * 50 msec) = 100 msec	100			
	Verify that digit chunks are consistent with scale of either current *min* or *max* (50 msec + 50 msec)	100			
	If chunks match, read left-most digit	290			
	Compare with *min* or *max* depending on digit chunks	50			
	Verify results and update *min* or *max* as necessary	50			
	Sub-total when chunks match		720		
Note that since all the numbers either fall within the *2 chunk (hundred thousand)* or *2+ chunk (million)* categories, we must always read the most significant digits to make the comparison with the current *min* and *max* values.					
Total time for processing all elements to get approx. min and max	33 * 720			23760	
Now we examine the time taken to process the ranking for each *total number of votes* category. Three categories: 1. States with > 1.5 mil. votes. 2. States with high hundred thousands of votes to 1.5 mil. 3. States with low hundred thousands of votes. Note that we do not charge any time for determining the bounds of these three categories because the time taken is difficult to estimate and we do not believe that it is significant because only approximate bounds are needed and no computation is necessary.					
Step 1: Process bars in each state size category beginning with the largest states followed by the mid and then small states.	Attend process states that are in current category	50			

Step 2: Scan next number and determine whether it falls within the current category.	Attend determine next state that falls into current total benefits category	50			
	Initiate eye movement	50			
	Eye movement	30			
	Count chunks	100			
	Compare if number is within current desired category range based on chunk count	50			
	Sub-total		280		
	If so, read one or two most significant figures	290			
	Compare if number is within current desired category range	50			
	Verify results	50			
	Sub-total for entire step 2		670		
Step 3: If number is within current category bounds, then process ranking information. Specifically get the color of the left-most mark or label. The three objects are close enough in the display that an eye movement is not needed to get to the left-most mark.	Attend left-most mark color	50			
	Perceive color	100			
	Verify results	50			
	Sub-total		200		
Total time computation: There are three categories thus *Step 1* has to be performed 3 times = 3 * 50 msec. = 150 msec. *Step 2* has to be performed on all elements in the table three times (once for each *total number of votes* category). Some elements require reading of the most significant figure while others do not as they can be discarded with just the chunk count. In this particular data set, there are 13 numbers with 2 chunks and 20 numbers with 2+ chunks. Thus for category 1, we can discard 13 numbers based on just chunk comparison, and we have to fully process the other 20 numbers. Total time for category 1 = (13 * 280) + (20 * 670) = 17040 msec. For the second category we need to full process all numbers but we **do not** need to perform the chunk comparisons. Thus total time = 33 * 620 = 20460 msec. For the third category we discard 20 numbers and only fully process 13. Thus total time = (20 * 280) + (13 * 670) msec = 14310 msec. Total time to verify whether a state falls within its category for all three categories = 17040 + 20460 + 14310 msec = 51810 msec. *Step 3* is only performed once for each state because we assume that a state only ever belongs to at most one of the categories. Thus time taken = 33 * 200 msec. = 6600 msec.					
Total time	23760 + 150 + 51810 + 6600			**82320**	

Note that the time taken using this design is much higher than that of previous designs. This is because it is difficult to perform groupings of elements based on the *total number of votes* data attribute when it is mapped to *labels*. The same situation arose in Task 2, Design 5, where the estimated total time was higher than a subsequent lower ranked design (Task 2, Design 6). Higher costs are not assigned to *labels* because our automatic designer tries to balance the accuracy goal with the perceptual goal of being able to quickly identify patterns. While *labels* are not very effective for the latter goal, it does provide very accurate results. Since there is no preference in the current task specification one way or the other, our designer balances both conflicting goals, with a stronger preference for the pattern identification goal. Refining the task with more complete accuracy preferences will help the designer adjust the weights between these two goals as necessary.

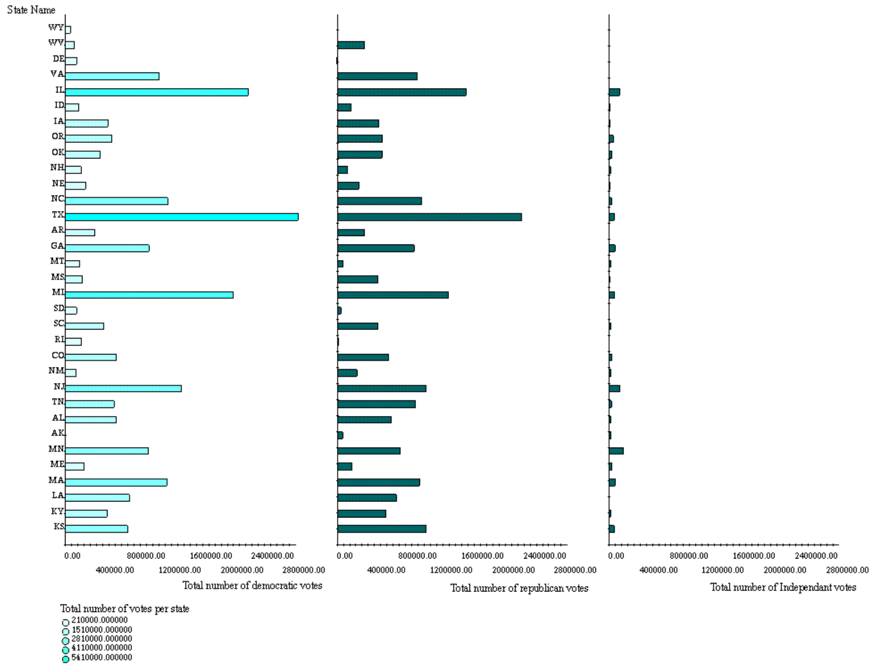
Another weakness of the design presented here with respect to the task is that it is difficult to determine where to set the category bounds based on *state* size. This is because the category bounds to some degree is based on the pattern in the *party ranking* data and we cannot process that information until after we have set up some temporary category bound values for grouping the elements. Thus we may have to readjust the category bounds as necessary mid-way in the analysis, and this will further lengthen the required time.

E-3.7 Design 7



This design is a hybrid design, with *total number of votes* pre-computed and mapped to *saturation*. The ranking is left to be performed perceptually by the user. Specifically *number of democratic votes* is mapped on the *x-lengths* of the left-most chart, *number of republican votes* is mapped on the *x-lengths* of the middle chart, and *number of other votes* is mapped on the *x-lengths* of the right-most chart. This design is ranked low for several reasons. The primary reason is because the information is spread out over many different spaces and this lack of integration as we have shown previously increases perceptual load and forces users to perform comparisons of values across different charts which can be difficult. Secondly, more data attributes are shown here compared to most of the previous designs. In this design five data attributes are mapped including *total number of votes*, *number of democratic votes*, *number of republican* *votes, number of other votes,* and *state-name*. Finally, the computed *total number of votes* is mapped to *saturation*, and *saturation* is not a very accurate graphical property for showing continuous values.

As was the case in Task 2, Design 4 and Design 7, having consistent axis scales here for the three charts can facilitate the *comparison* task. Thus we start our GOMS procedure with getting the combined *min* and *max* values for the three charts (0.0, and 300000) and then altering the required axis ranges so that all three charts share identical *min* and *max* values. This altered chart design is shown below.



Before we begin the task, we note that all of the bars in the rightmost chart are very short compared to the two other charts and as a result they can be discounted from our subsequent comparison operations. To perform this task, we start processing the left-most chart and search for bars that are highly saturated. For each highly saturated bar, we scan to its rightmost edge and note its length. We then scan to the right and compare its length with the corresponding bar on the middle chart. Note that for bars that have similar lengths, it may be difficult to accurately compared their lengths especially across different graphical spaces. For such cases, we can estimate the length of a bar using two fingers, and then move our hand over to the related chart and compare our finger interval with the corresponding bar in the chart. This allows us to more accurately maintain perceptual state (i.e. *bar length*) across charts and as a result helps us perform more accurate length comparisons. Another alternative would be to lookup the actual number of votes of the bars from their respective axes in order to get an accurate comparison result. This technique is even more accurate than the finger interval processing method, but it is cognitively taxing and requires much more processing time. Below we estimate the time taken for both these cases.

General goal	Cognitive, perceptual, or articulatory step taken by user	Time taken (msec)	Sub total (msec)	Total time (msec)	Target objects (data or graphics)
Total time taken to get consistent axis min-max values for the three charts.	Time taken from Task 2, Design 7	3670			0.0, 300000
Click on first min or max value to change	Time taken from Task 2, Design 7	370			
Type in min/max value	Attend input in the min value	50			
	Initiate move hand to numeric keypad	50			
	Move from mouse to numeric keypad	132			
	Attend type value	50			
	Type in value: Number with move required takes 40 + 60 + 60 = 160 msec. (move, upstroke, downstroke) Number with no move required takes 60 + 60 = 120 msec. (upstroke, downstroke) Value 0 takes (1 * 160) = 160 msec. Value 300 000 takes (2 * 160) + (4 * 120) = 800 msec.	160/ 800			
	Verify results	50			
	Sub-total for min entry		492		
	Sub-total for max entry		1132		
Move hand back to mouse	Time taken from Task 2, Design 7	182			
Total time to actually enter in the proper min-max values includes: 3. Time taken to enter in the min values = 2 * (370 + 492 + 182) = 2088 msec. 4. Time taken to enter in the max values = 2 * (370 + 1132 + 182) = 3368 msec. Total time for entry = 5456 msec.					
Now that we have re-scaled the three charts, we proceed to processing the bars.					
Process bars based on decreasing saturation	Attend get next most saturated bar	50			
	Initiate eye movement	50			
	Eye movement	30			
	Attend get bar length	50			
	Perceive bar length	100			
	Sub-total		280		
Scan to same row over to the right and get length of bar there	Attend get length of bar to the right	50			
	Initiate eye movement	50			
	Eye movement	30			
	Perceive bar length	100			
	Sub-total		230		

Compare the two bar lengths	Attend compare	50			
	Compare	50			
	Verify results	50			
	Sub-total		150		
Time taken to process one bar row	280 + 230 + 150	660			

As was noted previously, in some cases the bar lengths may be too close in value, and it is difficult to compare their lengths accurately across different graphic spaces. In this case, we have two alternatives:

Case 1: We use our fingers to mark off the length of the first bar on the left chart and then move our hand over to the bar on the right, comparing the length between our fingers with the bar length.

Case 2: We lookup the number of votes corresponding to the given lengths from the *x-axis* and then compare these values.

CASE 1:

Time taken to process a single row of bars for Case 1:

In this case we assume that the user increases accuracy of comparison across charts by using their fingers to approximately capture the bar lengths and to transfer this state across charts. To facilitate finger pointing etc, we assume that the chart height takes up approximately an entire 21 inch display screen, with a height of approximately 11 inches. Since there are 33 bars, each bar height is approximately 0.33 inches. We also observe that the data distribution results in an average move of about 6 steps from one bar to another which approximates to 6 * 0.33 = 2 inches.

We will also assume that the width of the visualization takes up the entire width of the display screen, measuring at approximately 14 inches. Each chart is therefore approximately 4.5 inches. Average length of a bar is approximately a third of the chart width thus average bar length = 4.5 /3 = 1.5 inches.

Move hand to appropriate bar. Total time to move hand based on Fitts Law = $100 * (\log_2 (2.0/0.33)+.5)$ = 271 msec.	Attend move hand to current bar being processed	50			
	Initiate hand move	50			
	Hand move	271			
	Sub-total		371		
Move fingers simultaneously to the beginning and end of bar. Assume that the range of area at bar end-points = 0.05 inches Total time to move fingers using Fitts Law = $100 * (\log_2 (0.75/0.05)+.5)$ = 395 msec. Note that perceiving the end points of the bar is taken into account in Fitts Law.	Attend fingers to begin and end of bar	50			
	Initiate move	50			
	Move fingers	395			
	Sub-total		495		
Move hand to left chart exactly over the relevant bar. Total time to move hand using Fitts Law = $100 * (\log_2 (4.5/0.33)+.5)$ = 382 msec.	Attend hand move	50			
	Initiate hand move	50			
	Hand move	382			
	Sub-total		482		
Compare width indicated by finger with width of current bar	Attend compare	50			
	Compare	50			
	Verify results	50			
	Sub-total		150		
Total time taken to process a single row for CASE 1	371 + 495 + 482 + 150	1498			

CASE 1: Total time includes: Time taken to initially process all bar pairs = 33 * 660 msec. = 21780 Time taken to process all the bar pairs with similar lengths. In the design above we found 17 such bars thus the total time for this operation = 17 * 1498 msec. = 25466 msec.					
Total time taken for entire task assuming CASE 1 = Time taken to determine min-max values for 3 charts (3670) + Time taken to re-scale all chart axes (5456) + Time taken to process bars (21780 + 25466)					
Total time taken for task assuming CASE 1	3670 + 5456 + (21780 + 25466)			**56372**	
CASE 2: Time taken to process a single row of bars for Case 2:					
Lookup number of democratic votes for current bar row	This time value was taken from the computed time to lookup values on axis in Design 4, Task 1	1135			
Lookup number of republican votes for current bar row.	This time value was taken from the computed time to lookup values on axis in Design 4, Task 1	1135			
Compare the two vote numbers to see which is higher. Here we assume that in the previous lookups the numbers are rounded up to two significant figures. This is because it is difficult to keep longer numbers in STM and it is difficult to compare such numbers as well. In this example it is sufficient in all cases to process the numbers up to two significant figures.	Attend compare	50			
	Compare first digit	50			
	Compare second digit	50			
	Verify results	50			
			200		
Total time taken to process a single row	1135 + 1135 + 200	2470			
CASE 2: Total time includes: Time taken to initially process all bar pairs = 33 * 660 msec. = 21780 Time taken to process all the bar pairs with similar lengths. In the design above we found 17 such bars thus the total time for this operation = 17 * 2470 msec. = 41990 msec.					
Total time taken for entire task assuming CASE 2 = Time taken to determine min-max values for 3 charts (3670) + Time taken to re-scale all chart axes (5456) + Time taken to process bars (21780 + 41990)					
Total time taken for task assuming CASE 2	3670 + 5456 + 21780 + 41990			**72896**	

Not surprisingly the time taken for case 2, which only requires perceptual and motoric loads is lower than the time taken for case 1, which requires cognitive computation and comparison. It is also interesting to note that the time estimated here for both cases is less than the estimated time for the previous design. This is made possible by the nature of the data distribution and the specific nature of the task we chose for our GOMS evaluation. Specifically, significant time savings were realized here due to the fact that we did not need to process information from the third chart at all.

Note that in the general case where all the information must be processed, the total time taken for both cases becomes significantly higher and exceeds that of Design 6. We show the modified computations for complete processing of all chart data below. The time taken to initially process the bars are increased by 230 msec + 150 msec per row from having to scan to and make comparisons with bars in the third chart. In addition, the time taken to process bar pairs with similar lengths is also increased based on additional processing with the right-most chart. We assume that the data distribution is similar to that of the previous charts, thereby requiring double the number of similar length comparisons.

<table>
<tr><td colspan="6">CASE 1:
Total time includes:
Time taken to initially process all bar pairs = 33 * (660 + 230 + 150) msec. = 34320 msec.
Time taken to process all the bar pairs with similar lengths. In the design above we found 17 such bars thus the total time for this operation = 17 * (1498 + 495 + 482 + 150) msec. = 44625 msec.</td></tr>
<tr><td colspan="6">Total time taken for entire task assuming CASE 1 =
Time taken to determine min-max values for 3 charts (3670) + Time taken to re-scale all chart axes (5456) + Time taken to process bars (34320 + 44625)</td></tr>
<tr><td>Total time taken for task assuming CASE 1 and full processing of ALL data</td><td>3670 + 5456 + (34320 + 44625)</td><td></td><td></td><td>88071</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

<table>
<tr><td colspan="6">CASE 2:
Total time includes:
Time taken to initially process all bar pairs = 33 * (660 + 230 + 150) msec. = 34320 msec.
Time taken to process all the bar pairs with similar lengths. In the design above we found 17 such bars thus the total time for this operation = 17 * (2470 + 1135 + 200) msec. = 64685 msec.</td></tr>
<tr><td colspan="6">Total time taken for entire task assuming CASE 2 =
Time taken to determine min-max values for 3 charts (3670) + Time taken to re-scale all chart axes (5456) + Time taken to process bars (21780 + 43690)</td></tr>
<tr><td>Total time taken for task assuming CASE 2 and full processing of ALL data</td><td>3670 + 5456 + (34320 + 64685)</td><td></td><td></td><td>108131</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

E-3.8 Summary

As with the previous two tasks, Figure E-7 shows that in most part the ranking assigned by our automatic design system is consistent with the GOMS estimated total times. In Figure E-7 there are two bars shown for Design 7, and one of these bars (purple bar) has a lower estimated total time compared to Design 6. However as was discussed previously this was only made possible because in Design 7 we were able to solve the task without considering any of the information in the right-most chart. In this example the data distribution is such that the information in the right-most chart has no effect on the results of the party-ranking task. When we estimate the time taken to process all of the information, the total time for Design 7 exceeds that of Design 6 as is shown by the second (right, green) bar. It would be very challenging and interesting to characterize this class of perceptual shortcuts and encode that information as heuristics into our automatic design system. However, this issue is complex and we leave it for future work.

Like the previous computation task, most of the top designs for this comparison + simple compute task are pure data technique designs or hybrid data + mapping designs. Out of the 19 visualization alternatives generated, 5 were pure data technique designs, 12 were hybrid data + mapping designs, and only 2 were pure mapping designs. Thus most of the top designs for this task (89.5 %) are only made possible because of the design space expansion from adding data techniques into the automatic design process.

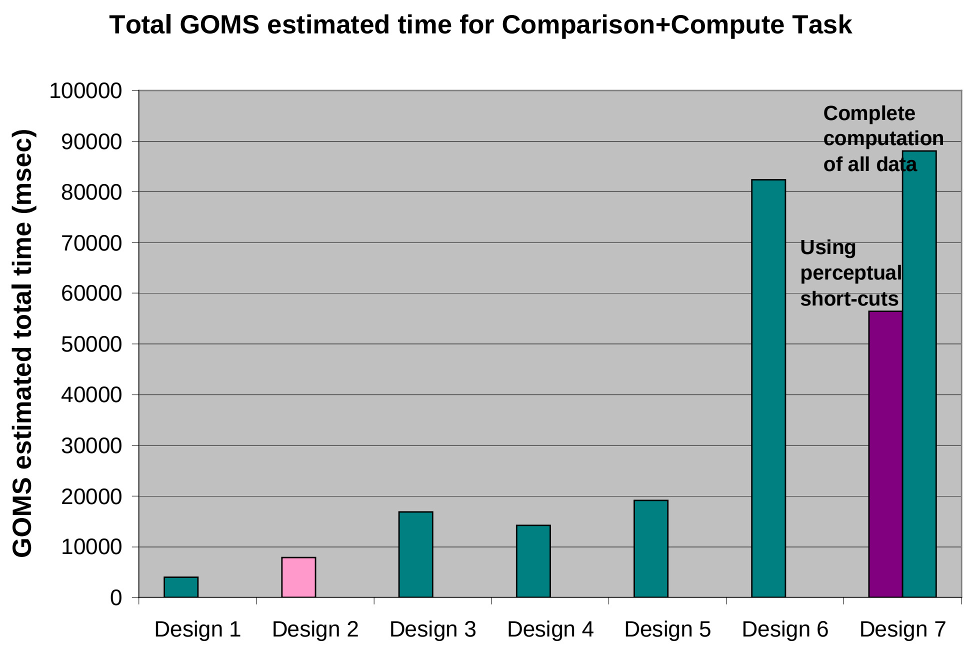


Figure E-7: GOMS estimated total time for Task 3 (comparison task). The designs are ordered based on increasing cost on the *x-axis*. The *y-axis* shows the GOMS estimated total time in msec. All pink bars indicate pure mapping designs (i.e. designs that can be generated with current state of the art automatic design research). All other bars are designs made possible by work outlined in this thesis.

An interesting feature in Figure E-7 is that the design ranked second is a pure mapping visualization. This is in contrast with the previous computation task where the best pure mapping design was ranked 6^{th}. There are several reasons why the pure mapping design (Design 2) performs well here. First of all, the two sub-tasks in the specification, i.e. the *summation* and *sort* tasks, both operate on exactly the same data attributes (i.e. number of votes for each of the three parties). In addition, both tasks are facilitated by similar mapping constraints (e.g. mapping all the data attributes to the same graphical properties, in this case, *x-position*). Because of their shared data attributes and task constraints we are able to use the same graphical objects to solve both tasks as in Design 2. In addition, the computation task in this example is just a simple *summation* that can be effectively represented by using stacked bars. Thus the nature of the two tasks allows for an effective perceptual mapping to be achieved. However, these perceptual requirements are so tight that there are few designs that are able to fulfill all of them. Hence other than Design 2, there are no other pure mapping designs in the top spots.

Despite the ability of Design 2 to combine the perceptual goals of both the tasks so well, the top design in this example is still a data transform design with both *total number of votes* and *party-ranking* pre-computed. While both Design 1and Design 2 have the same number of objects, the former requires fewer data attributes to be mapped (i.e. only 3, namely *total number of votes, party ranking* and *party type*) compared to the five attributes in Design 2 (*number of votes for democratic party, number of votes for republican party, number of votes for other party, state-name* and *party-type*). In addition the comparison task may sometimes be difficult to perform on Design 2 because of the bar stacking and the narrow differences in values between the number of *democratic and republican* votes. In contrast the ranking results in Design 1 require very little perceptual effort and are accurate. For these reasons our automatic designer chose Design 1 over Design 2.

As with the previous task, some of the designs here can be made more effective by refining the task so that we sort the *states* based on their *total number of votes*. This sorting makes it unnecessary or very simple to perform search and categorization tasks. We discuss task refinements in the next section and how it relates to the automatic visualization design process.

E-4 Task Refinement and Sorting

As was discussed previously it is rarely the case that a task is completely and accurately specified at the start of a data analysis or communication session. Users often refine their task or communication goals through an iterative process where they consider output from the computer system, integrate that output with their current task model, change that model as necessary, and finally convey those changes to the computer system. At the end of this cycle, the iterative process begins anew. For example, consider Task 2 (computation of total benefits) and Task 3 (comparison of party votes and computation of total votes). In both these cases, once we analyzed the initial output designs, we recognize that our ability to solve the tasks can be enhanced by refining our original specification to include a *sort* operation. This is because in both these cases we were interested in finding or grouping objects based on a single attribute (*total benefits* attribute in Task 2 and *total number of votes* attribute in Task 3). By sorting the objects in the display based on the relevant attribute we can quickly perform the task with less perceptual load.

In the original tasks (before the *sort* addition) it was not made clear to the design system what the user intended to do with the *total benefit* values in Task 2 or the *total number of votes* values in Task 3. The user's intention could have been to compare the *total benefit* ratios among several universities, find universities with particular *total benefit* values, lookup the *total benefits* for given universities, group universities based on *total benefits,* etc. According to Bertin [*Bertin, 1981*], tasks can be divided into three levels of reading: elementary, intermediate, and global. For elementary tasks it is more important to be able lookup single values clearly and accurately, while for the intermediate and global tasks it is more important to be able to get a sense for general gestalt patterns or trends in the data set. When no information is given with respect to the level of reading desired (e.g. what to do with the *total benefit* values in Task 2 once computed), our designer tries to make design decisions that balance between the three different levels, as was done in previous systems. Essentially the higher levels of reading (e.g. finding data patterns) are given preference over elementary readings, unless the user explicitly specifies a desire for the latter.

In these Task 2 and Task 3, the big win with expanding the specification with a sort operation is not necessarily the pre-computation of the sort but rather the ability to integrate the sort results into the graphic effectively while reducing perceptual complexity. This can be achieved by recognizing that when a nominal attribute (e.g. *university name* or *state name*) is mapped to *position* we are not utilizing all the expressive power of the *positional* graphical property. Specifically *positionals* can capture order or ranking information as well. Thus it is possible to show the ranking results by integrating it into an originally *nominal, unique* attribute (e.g. *university name* or *state name*) and turning that *nominal* attribute into an *ordinal*. In this way, additional information is included into the graphic without increasing the number of graphical properties or objects in the visualization.

University Name

Michigan_State_University
Univ_of_Texas_at_Austin
Univ_Minnesota_Twin_Cities
Univ_of_Southern_California
University_of_Pennsylvania
Texas_A_&_M_University_Main
Massachusetts_Inst_of_Tech
Univ_of_Cinti_Main_Campus
Univ_of_Illinois_Urbana
Univ_of_Tennessee_Knoxville
University_of_Utah
Louisiana_St_Univ_and_A&M_C
Northeastern_University
Bowling_Green_State_Univ
Duke_University
Ohio_University_Athens
Vanderbilt_University
Brown_University
Oklahoma_State_Univ_Main
Emory_University
University_of_North_Texas
Carnegie_Mellon_University
Hofstra_University
Northern_Arizona_University
Univ_of_North_Dakota_Main

Figure E-8: List of universities ranked based on their computed total-benefit values

For example Figure E-8 shows a visualization design which lists the university names based on their *total-benefit* values. This design fully pre-computes the *sort* augmented, total benefits *computation* task and adds the ranking information to the university names, which are mapped to *y-position*.

This design is more effective than the best design in Task 2 because no perceptual load is required to identify the four longest bars as was the case in Design 1, Task 2. Instead users can just read off the first four names on the sorted list. Note that mapping the sort results in any way other than the *nominal-to-ordinal* method described above does not result in any clear processing benefits. This is because the sort results are a quantitative attribute very similar to the total benefits attribute. By computing and showing the *sort* results instead of the *total benefit* results we are essentially replacing a quantitative attribute with another. For example in Figure E-9 ranking information is mapped to *mark x-position* while in Figure E-10 *total benefit* values are mapped to bar *x-lengths* as in Design 1, Task 2. Both the designs take approximately the same amount of time to process and their GOMS procedures are almost identical. The only time when Figure E-9 is possibly superior is when there is data dwarfing.

Figure E-9: Visualization design where the total benefit values are sorted and the sorted rankings are mapped to *x-position*

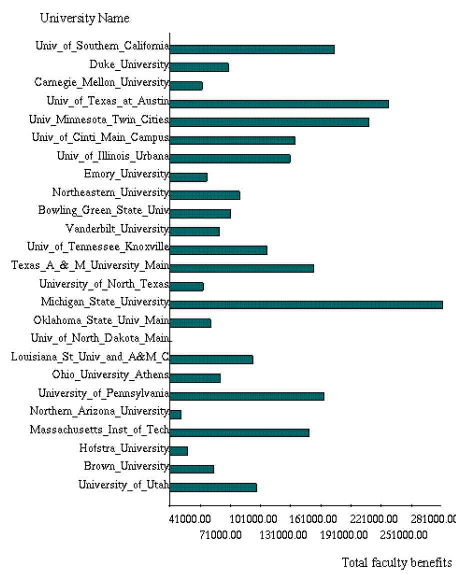


Figure E-10: Visualization where the total benefit values are mapped to bar *x-lengths*

Another interesting aspect of this example is that sorting can help highlight certain types of data trends and patterns. I.e. sorting can support certain tasks with intermediate and global reading level requirements. In such cases it becomes less important to encode the sorted attribute values with graphical properties that facilitate these higher levels of reading. Instead we can give a greater preference to the elementary reading levels and encode the attribute to facilitate lookup accuracy. For example, rather than ordering the universities top down based on benefit values and at the same time showing the *total benefit* attribute using *position* (as in Figure E-11), it may be more beneficial to map *total benefits* to *text* instead (as in Figure E-12). This is because sorting supports some of the higher level reading tasks and text labels are a good complement for providing support to the lower reading levels.

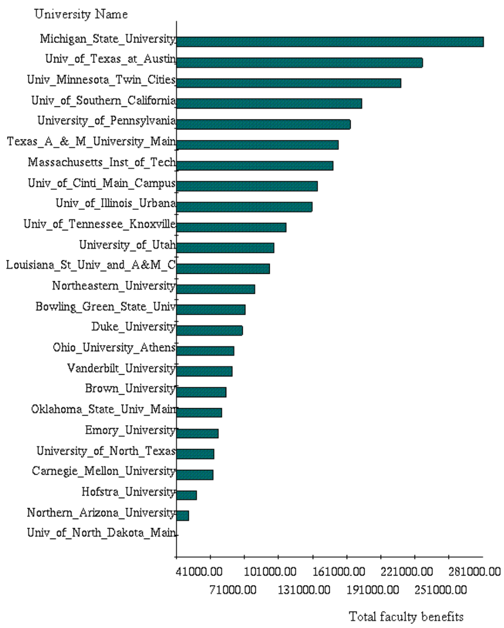


Figure E-11: Visualization where the universities are ordered on the *y-axis* based on *total-benefits* and in addition the *total-benefit* values are mapped to *bar-lengths*.

University Name	Total faculty benefits
Michigan_State_University	320788.00
Univ_of_Texas_at_Austin	265209.00
Univ_Minnesota_Twin_Cities	245474.00
Univ_of_Southern_California	209920.00
University_of_Pennsylvania	199225.00
Texas_A_&_M_University_Main	188679.00
Massachusetts_Inst_of_Tech	183356.00
Univ_of_Cinti_Main_Campus	169492.00
Univ_of_Illinois_Urbana	164436.00
Univ_of_Tennessee_Knoxville	140956.00
University_of_Utah	129774.00
Louisiana_St_Univ_and_A&M_C	126133.00
Northeastern_University	112416.00
Bowling_Green_State_Univ	103558.00
Duke_University	101040.00
Ohio_University_Athens	92852.00
Vanderbilt_University	91711.00
Brown_University	86175.00
Oklahoma_State_Univ_Main	82513.00
Emory_University	79007.00
University_of_North_Texas	75046.00
Carnegie_Mellon_University	74240.00
Hofstra_University	59244.00
Northern_Arizona_University	52341.00
Univ_of_North_Dakota_Main	40730.00

Figure E-12: Visualization where the universities are ordered on the *y-axis* based on *total-benefits* and in addition the *total-benefit* values are mapped to *text labels*.

We want to point out that the sorted designs (e.g. Figure E-8, Figure E-11, and Figure E-12) cannot be generated with current state of the art automatic systems because no pre-processing (i.e. data transformation) operators are considered in their design process. Thus, this example further underscores the importance of integrating data operations into automatic design because as we saw in Figure E-8, Figure E-11, and Figure E-12 pre-processed sorts can improve design effectiveness in interesting ways. In our system we are able to pre-compute the sort task but we currently **do not** allow operations that can alter the basic characterization of data attributes. I.e. we do not allow the ranking information to be added into *university name* by converting it into an *ordinal*. We suspect that there is a whole class of data alteration opportunities and thus we leave fuller treatment of this issue for future work.

In the *sort* example above, we present a simple task refinement operation, where a new task is added, encapsulating one of the previous tasks. Recall that in appendix C-5, we presented a *car purchasing* example where more extensive and complex task refinements were made. Currently our designer does not explicitly support the task refinement process. Each design requested is assumed to be unrelated and thus the designer does not strive to maintain any visual consistency between one request to the next. This visual consistency issue is an important one and it is currently being explored by other researchers in the field. We hope to integrate such work into our designer in the future. Of course the automatic design process implicitly supports the task refinement process by presenting users with a set of alternative visual designs for solving the same tasks. By analyzing and comparing these designs, the important elements of the user's task may become clearer and this would in turn help in subsequent task refinements.

E-5 Why GOMS?

The primary reason why we chose GOMS as an evaluation method for our work is because it would take a significantly longer amount of time to perform the same tests with users. In addition by mapping out the minute cognitive, perceptual and motoric steps required in a GOMS analysis we are able to better analyze the particular strengths and weaknesses of visualization designs and identify where most of the processing time is being spent. Some of the weaknesses of the GOMS analysis method used here, however, is that we only model the performance of experts and it is difficult in many cases to account for task accuracy problems, readability problems (such as occlusion, graphic density, and dwarfing) as well as more complex usability issues such as user experience, user fatigue, number of errors performed by users, task solution formulation (time taken to come up with a solution on how to solve the task with the graphic at hand), etc. While it would be interesting and useful to collect this type of information, they are **not required** for testing the three major issues we want to evaluate in our system. Specifically we want to determine the following:

- Our theories can be implemented and they perform as expected. I.e. the design metrics and heuristics used in the designer result in output designs that are ordered according to complexity of use (by "complexity of use" we refer to cognitive, perceptual, and motoric complexity).
- Our work increases the breadth of designs that can be generated by automatic systems. I.e. our automatic system should be able to produce designs that cannot be previously generated.
- Our work improves the effectiveness of visualizations generated. I.e. the expanded design space contains visualizations that allow certain task classes to be solved more effectively.

As was previously specified these goals can be adequately evaluated using GOMS because first of all we are only interested in testing the quality of our design with designs that can be generated by current state of the art automatic systems. We do not make any claims with respect to the quality of designs generated by our system and those that are generated by a human designer. Secondly all of the GOMS evaluations are generated based on the same set of assumptions and estimate time measurements. All procedures are also modeled based on expert performance and this consistency in the evaluation method makes the time comparisons among the different GOMS analyzed designs more equitable because any change in assumptions will affect all of the GOMS procedures in consistent ways. Finally and most importantly we do not require accurate time estimates for this evaluation. To ensure the correctness of our system it is only necessary for us to identify groups of designs that have similar time estimates and ensure that these groups are ordered properly by our design system. I.e. the absolute time estimates are less important, what is more important are the large time differences between the different design groups and our analysis of what are causing these differences.

We also stress that while it is possible to evaluate designs in our automatic system by generating GOMS sequences, this is a very expensive process and can cause many complex issues to arise when applied to partial designs. Single small changes to partial designs may cause a significant change in the GOMS sequence used and may invalidate a previously forecasted sequence. This would in turn result in great swings in the time estimations. In addition as we have discussed in this section GOMS does not take into account certain important issues such as accuracy and readability problems. Thus we have opted to use a higher level more abstract cost system in our designer based on the heuristics and metrics presented in chapter IV. Details of this cost structure are presented in chapter V. In this appendix however, we proved the correctness and feasibility of our higher level cost structure by showing that our designer orders its output designs consistently with GOMS time estimations.

E-6 Conclusion

In summary the GOMS analyses in this section validates the three evaluation goals we set out to test. Specifically, we showed that:

- The output order of our automatic designer (i.e. our design heuristics used) does indeed conform to cognitive, perceptual, and motoric complexity as computed by GOMS. I.e. the theoretical concepts developed here for characterizing and expanding the visualization techniques design space for automatic visualization generation can be implemented and the results are meaningful (i.e. conforms to GOMS computed times).
- Adding data transform techniques into the automatic design process expands the visualization design space and enables whole new sets of interactive and non-interactive visualizations to be generated. Many of these designs are shown in the following sections.
- Some of the new designs generated as a result of work developed in this thesis (i.e. pure data transform techniques and hybrid data + mapping designs) perform much better than the designs that can be generated with current state of the art technology (i.e. pure mapping designs) for the task classes we considered (*computation, search, comparison*); with the highest gain in computation tasks.

Demonstrating these three goals with GOMS validates the thesis statement set out in this document. This GOMS evaluation also attests to the generality and usefulness of the rules employed in our automatic design system and places our cost structure on a concrete, proven, empirical basis of cognitive, perceptual, and motoric steps. The analyses and design examples presented here also underscore the importance and richness introduced by data transform techniques to the visualization design process.

Throughout this appendix we have also highlighted important issues and analyzed why our design system ranks certain visualizations over others. More details are given on our designer ranking algorithms in the chapter V where we describe the implementation details of our system. Some of the important issues brought forth in this evaluation appendix help scope out the areas that we deal with in our work and the areas that we don't. Some of the interesting challenges that we leave for future work include the problem of ensuring visual consistency for task refinements, readability issues (appendix F), and operations that alter the general structure and type of data attributes (e.g. in the sorting example where *university name* is converted from a *nominal* to an *ordinal*).