Chapter IV: Design Heuristics

Data Computation vs. Perceptual Mapping

The goal of this thesis is to enhance the breadth and quality of designs generated by an automatic visualization system by adding data processing operations to the design process. The consideration of data transforms extends previous automatic systems, which only considered mapping transforms (i.e. mapping data to graphics). In the previous chapters we laid out a framework which divides the visualization design process into four primary steps: data transforms, mapping transforms, graphical transforms, and rendering transforms. In this chapter, we discuss integration of data transforms together with mapping transforms, the issues that arise, and the design improvements that may be realized with this expansion. In appendix F, we speculate about the integration of graphical and rendering transforms into the automatic design process.

We begin this chapter with an example that shows where previous work leaves off, and how consideration of data transforms will improve the designs that can be produced.

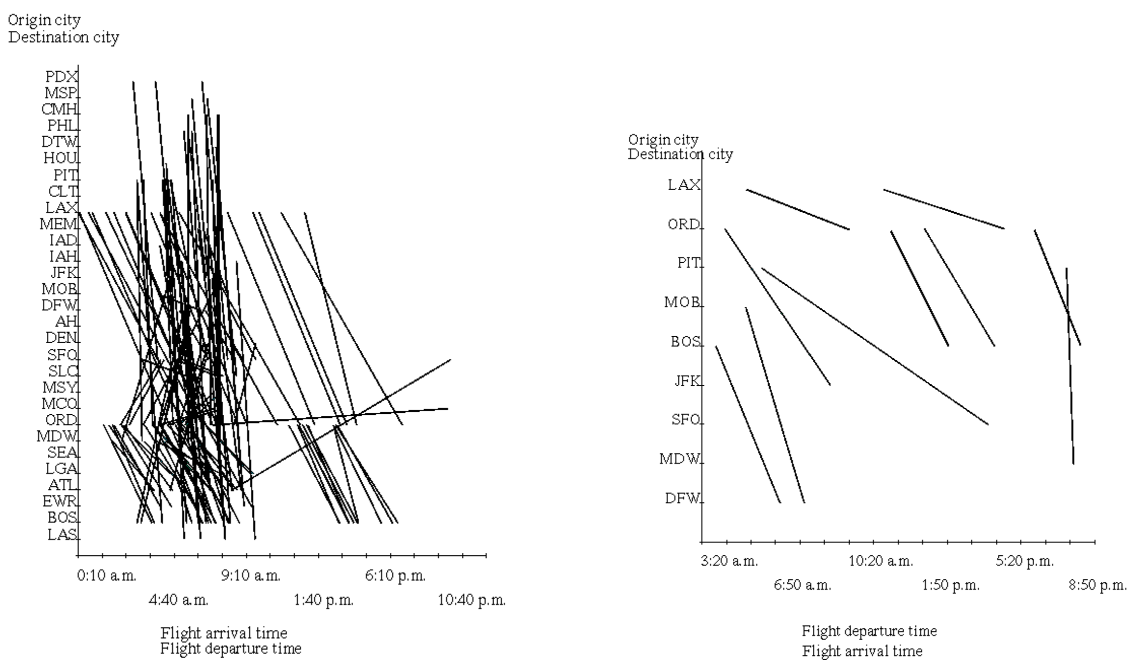
IV-1   An Airline-Scheduling Example Illustrating the Use of Data Transforms

This airline-scheduling example was used by Casner [*Casner, 1991*], to illustrate the importance of considering a user's task in mapping data to graphics (i.e. mapping transforms). We show that this "airline reservations" task can be better supported if we consider ways to transform or reorganize the data in addition to mapping it to graphics. The verbal description of the task that the visualizations must support is:

> *Given an origin and a destination city, the user "attempts to locate the two flights arriving in and departing from a layover city that offer the minimum amount of `down time' between the flight times and the beginning and ending time of a scheduled meeting (in the layover city)".*

Task IV-1: Airline-scheduling task. The user is trying to find flights to enable a meeting to be held in a layover airport en-route to a destination and to minimize time spent at the layover airport before and after the meeting.

In the following example designs, suppose that the origin and destination cities are *Los Angeles (LAX)* and *Boston (BOS)* and that the layover city is *Chicago (ORD)*. Further, suppose that the meeting time is from 2 p.m. to 4 p.m. Casner showed that this task can be achieved perceptually with the graphic design in Figure IV-1. In Figure IV-1, the *origin* and *destination* cities are encoded on the *y-axis* and the *departure* and *arrival* times are encoded on the *x-axis*. Each flight is represented by a line where the left-point of each line encodes the *origin city* and the *departure time* of a flight and the right-point of each line encodes the *destination city* and *arrival time* of a flight. For example in Figure IV-1b, the task solution is a flight leaving *Los Angeles* (LAX) at 5:30 a.m. noon and arriving at *Chicago* (ORD) at 10:20 a.m. and another leaving *Chicago* (ORD) at 7:00 p.m. and arriving at *Boston* (BOS) at 9:10 p.m.



(a) Full data set
This visualization shows all the elements in the data set (i.e. all 135 flights).

(b) Truncated data set. This example visualization shows the ideal case where there is little occlusion among the different flight lines. This data set was chosen so that it contains some flights that fulfill the task constraints as well as some other random flights that do not occlude one another.

Figure IV-1: Solving the airline-scheduling task fully perceptually (Casner's solution). Each line represents a flight with origin and destination city mapped onto the *y-axis* and arrival and departure time mapped onto the *x-axis*. This is the best design that gets generated when ONLY mapping operations are considered by the automatic system. I.e. this is the best possible design from current state of the art systems.

In Figure IV-2, we present Casner's analysis of the perceptual procedure a user must perform using the visualization in Figure IV-1 to achieve the airline-scheduling task (Task IV-1). Subsequently we show in Figure IV-3 how transforming the data makes this procedure simpler.

Subtask-1 :*<u>Find all the origin flights that fulfill the first leg of the flight schedule</u>*

Visually search for all flights whose origin is *Los Angeles (LAX)* and whose destination is *Chicago (ORD)*, the *layover-city*.

Search for *LAX* on the *y-axis* and then look over to the right for all flights that start from this *origin-city*.

For each of these flights, we find the end-point of the flight line and determine whether it goes to *ORD,* the *layover-city*.

For all flights that fulfill the origin (LAX) and destination (ORD) city constraints, check if they meet the meeting time constraints as well (arrives at ORD before 2 p.m.).

Look down on the *x-axis* to determine the *time of arrival* in *ORD*. If the time is after the scheduled meeting time, we discard the current flight as a possible candidate and continue looking for other relevant flights.

If the arrival time is before the scheduled meeting, we determine whether it has the smallest prior meeting downtime. If so we note the flight as the current most promising candidate and continue the process for all other flights.

Subtask-2 :*<u>Find all the destination flights that fulfill the second leg of the flight schedule</u>*

Find the earliest flight after the meeting using an analogous procedure to *subtask-1*.

At the end of *subtask-1* and *subtask-2* we would have determined the flights with the smallest downtimes before and after the meeting. To get the *total down time* we merely add the two downtimes

Figure IV-2: Casner's analysis of the perceptual procedure a user must perform with a visualization to achieve the airline-scheduling task (Task IV-1).

The pure perceptual procedure for the airline-scheduling task (Task IV-1), while relatively complex, is still more effective compared to a strictly *cognitive* procedure (i.e. looking at a spreadsheet table that contains the raw data). We performed a GOMS analysis for a tabular presentation of the data for the airline-scheduling task and estimated it to take approximately 4 minutes for task completion. In contrast, the perceptual solution represented in Figure IV-1only took 30 seconds. The GOMS analysis tables for both the cognitive and perceptual designs are presented in appendix C-1. However, we should point out that this analysis assumes that all the data fits within a single CRT screen, and there is no occlusion in the designs.

As can be seen from Figure IV-1a, when the data set size grows, the perceptual design quickly becomes unusable without interactive navigation of the display. For the cognitive solution, larger data sets would not fit within a single CRT screen thus interactive scrolling is required. Such navigation operations will add to the overall task time of both designs.

Figure IV-3 shows an alternative design for solving the same airline-scheduling task using the same data set as Figure IV-1b. The left chart shows all the flights that fulfill constraints for the first leg of the journey (*LAX* to *ORD* with *arrival time* before 2 p.m.) while the right chart shows all the flights that fulfill constraints for the second leg of the journey (*ORD* to *BOS* with *departure time* after 4 p.m.). The bar lengths in the left chart encode the computed total downtime before the meeting and the bar lengths in the right chart encode the computed total downtime after the meeting.

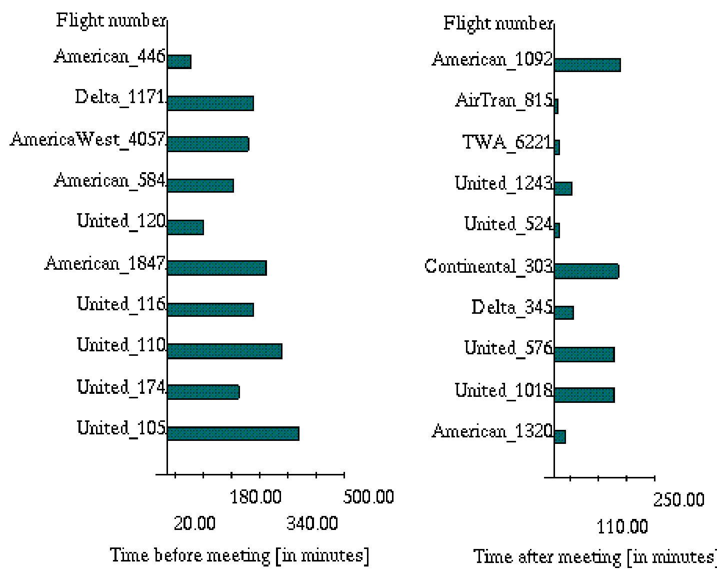


Figure IV-3: Our hybrid data transform and mapping transform design for solving airline-scheduling task. Here only the flights that fulfill the *city* and *meeting time* constraints are shown. Computation of the total downtime for the best flights is left to the user. *Time_before_meeting* is mapped to the *x-axis* of the left chart and *time_after_meeting* is mapped to the *y-axis* of the right chart. To perceptually compute the total downtime users add the shortest bar length in the left chart with the shortest bar length in the right chart.

In Figure IV-3, we are able to significantly simplify the perceptual complexity of the earlier designs as well as reduce visual clutter with **data transform techniques**. These techniques allow the automatic system to summarize the task results and filter out irrelevant flights thereby significantly improving the readability of the representation compared to Figure IV-1a. In addition, it also simplifies the perceptual procedure for solving the task because in this design users need not visually search for flights that fulfill the *city* and *time* constraints. Instead, all this information has been pre-calculated by the system with data transform techniques. To solve the airline-scheduling task (Task IV-1) using Figure IV-3 we only need to pick the shortest bar in the first chart (i.e. *American_446* which has the least downtime before our meeting) and the shortest bar in the second chart (i.e. *AirTran_815* which has the least downtime after our meeting). The total downtime can be estimated by perceptually adding the lengths of both these bars. The GOMS analysis for Figure IV-3 showed an estimated task time of only 3 seconds.

Figure IV-4 shows the GOMS time estimates for solving the airline-scheduling task using a purely cognitive procedure, a purely perceptual procedure (i.e. only mapping transforms, Figure IV-1), and a perceptual procedure (mapping transforms) combined with data transform techniques (Figure IV-3). We can clearly see that using both mapping and data transform techniques together is significantly more effective than using only mapping transform techniques which in turn is more effective than using only cognitive operators.

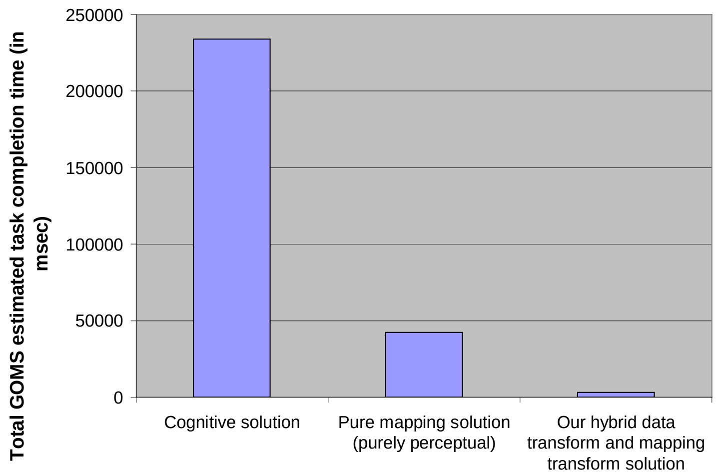


Figure IV-4: GOMS estimated total time for solving the airline-scheduling task using a pure cognitive, pure mapping, and a hybrid data + mapping design.

Earlier work on automatic visualization design [*Mackinlay, 1986a, 1986b; Casner, 1991, Roth, 1994*] centered purely on using mapping transform techniques (pure perceptual operators), preferring to address those tasks that cannot be easily accomplished through statistical computation (data transform techniques). This sentiment is well expressed by Tufte: "Why waste the power of data graphics on simple linear changes which can usually be better summarized in one or two numbers? Instead, graphics should be reserved for the richer, more complex, more difficult statistical material." Thus, previous work on automatic presentation systems have assumed that statistical processing has already occurred before the design process. These systems have instead focussed on developing design heuristics for making data-to-graphical mapping decisions based on both tasks and data. However, work on task analyses [*Springmeyer, 1992*] indicates quantitative processing (data transform techniques) is an integral part of graphic design. As was expressed by Springmeyer, "These results show that analyzing scientific data is a much more quantitative and active process than the passive viewing of images." Based on her analyses, Springmeyer was convinced that a large shortcoming of current visualization systems was their lack of integration with quantitative operations (i.e. data transforms). We believe that in order to design more effective visualizations for analyzing data, we must make data transform decisions together with mapping transform decisions. The brief GOMS comparison shown in Figure IV-4 supports this belief.

It is however erroneous to assume that data transforms will always be more effective than mapping transforms. Sometimes, over-computing a task creates more graphics and more work for the user (appendix C, Figure C-3). Other times, full pre-computation is just not possible. Commonly the most appropriate design for a task will consist of a blend of data and mapping transforms, as we will show in the next example. The most effective blend of data transforms and mapping transforms for a task sequence is dependent upon many factors including the task (e.g. whether the task requires simple or complex computation, whether the operation must be repeated many times, whether we know for sure what the task parameters are), the preferences of the user (e.g. whether they are comfortable with using input devices, whether they are familiar with particular input devices), and the availability of display and input resources (e.g. whether the elements will fit within the output media, whether physical devices are available for input). To make intelligent design decisions about how data computation and mapping techniques should be combined, we need to include data processing decisions as part of the automatic design process and not merely pre-compute the data beforehand. In the next example, we illustrate the weakness of over-computing the airline-scheduling task (Task IV-1).

One possible solution for Task IV-1 is to calculate the entire task with data transform techniques. In this case, the system would only present users with the two flights that produce the minimum *total downtime* (i.e *American_446* and *AirTran_815*). Having the system calculate the entire task with data computations, however, is only appropriate if we can fully and accurately define **all** our data analysis goals (e.g. meeting time constraint: >= 2 p.m. and <= 4 p.m.; origin = *LosAngeles;* layover = *Chicago;* and destination = *Boston*). Suppose that in addition to *total downtime* we were also concerned with *total cost* and *flight duration* (i.e. we want to choose flights with “generally low” *total downtime, total cost,* and *flight duration*). In this case, it is not possible for the system to calculate the entire task with data computation and only present users with one flight pair because we do not know what constitutes an acceptable balance between “low” *total downtime*, “low” *total cost* and “low” *flight duration*. The best balance between these three attributes can only be arrived at during the analysis process, after we have determined the number of flights that fulfill our *city* and *time* constraints and the data distributions of the acceptable flights with respect to our three attributes.

One way to solve this task is to use a design similar to the one in Figure IV-3 but augmented with *total cost* and *total duration* data (Figure IV-5). The left chart in Figure IV-5 represents flights that fulfill the first leg of our schedule and the right chart represents flights that fulfill the second leg of our schedule. In each chart, the labeled marks represent different flights. The *flight duration* is pre-computed, and encoded on the *y-axis*, *total downtime* is pre-computed and encoded on the *x-axis*, *total cost* is pre-computed and encoded with saturation, and *flight name* is encoded with *labels*. To find the most relevant flights, we look for marks on the left-bottom corner of each chart (low *downtime* and low *flight duration*) with low saturation values (low *total cost*). In the left chart (flight before meeting) the choice is clear. The best flight is *American_446* that has the lowest *flight duration* and *downtime_before_meeting*. Its cost (*saturation*) is also comparable with the other flights. In the right chart, some trade-offs must be made between *AirTran_815* that has the lowest *downtime_after_meeting* and *cost* and *United_576* that has the lowest *flight duration*. *AirTran_815* seems to be the better choice from Figure IV-5 because it has very low *downtime_after_meeting* and *cost*, as well as a *flight duration* that is not overly large whereas *United_576* has a very large *downtime_after_meeting*. Thus, this airline-scheduling task cannot be solved with a pure computation because the tradeoffs among the task attributes (*downtime*, *flight duration*, and *cost*) cannot be captured in a simple function and needs to be considered by the user. At the same time however, using a pure mapping design is also ineffective because of its high task complexity (in terms of both search and computation) as well as the many different data attributes that it combines. Thus, a hybrid data and mapping design (as in Figure IV-5) is the most appropriate here.

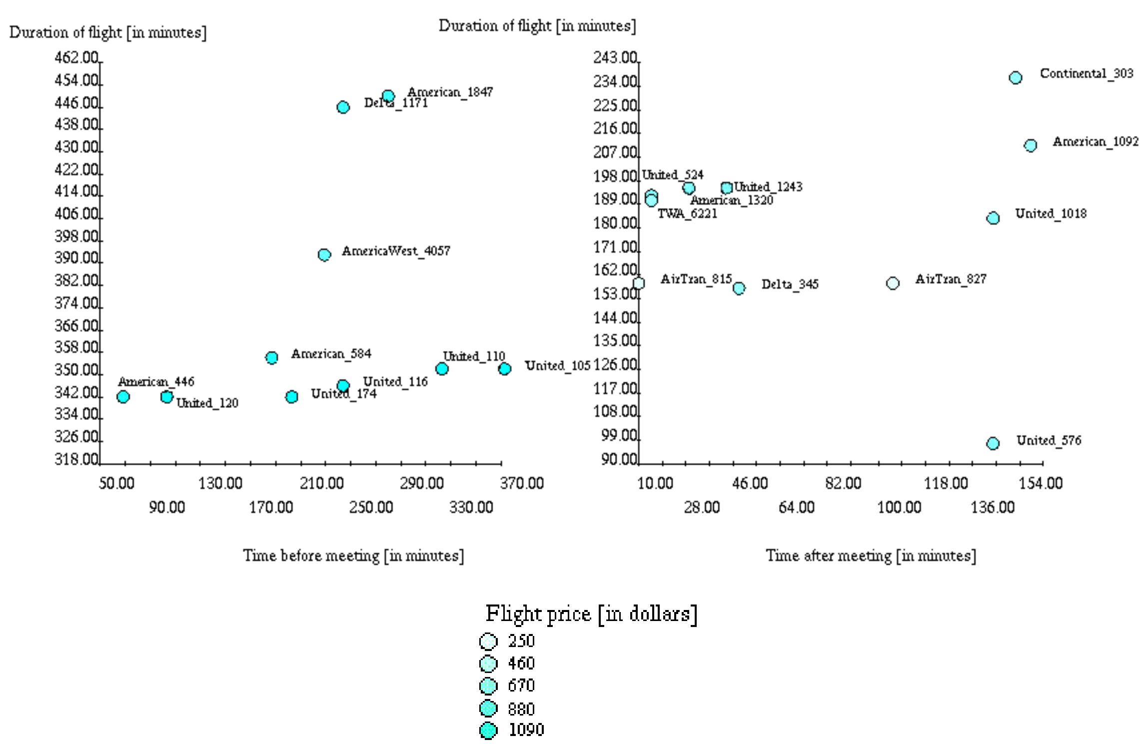


Figure IV-5: Visualization for finding flights with low total-downtime, low total-cost, and low duration. Because there are trade-offs that must be made among the three attributes, this task is best performed through perceptual perusal.

In appendix C-2 we present several other design alternatives for the airline scheduling task that have different blends of data and mapping transforms and discuss their strengths and weaknesses. The examples in this section and in appendix C-2 show us that there are many ways with which we can use data and mapping transforms to solve user goals. To make intelligent design decisions about how data computation and mapping techniques should be combined, we present a set of design dimensions for gauging the goodness of different visualizations (section IV-2). Based on these design dimensions, we develop higher level design rules for deciding when to use data transform techniques and when to use mapping techniques to solve tasks (section IV-3). In appendix C-4 and C-5 we introduce some task examples and show how our prototype automatic visualization designer addresses these tasks based our design dimensions and design rules.

IV-2 Visualization Design Dimensions

The airline-scheduling example presented in the previous section shows many different ways in which data and mapping techniques may be combined. To decide on which combination is most appropriate for a given task or set of tasks we need some standards of evaluation for the different designs (i.e. visualization design dimensions). In this section we present a set of design dimensions upon which to evaluate visualizations. Our dimensions are based on the interaction framework model presented by Abowd and Beale [*Abowd, 1991*] (Figure IV-6).

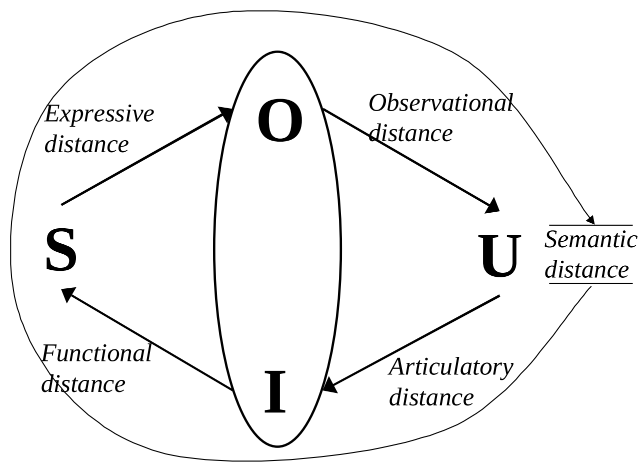


Figure IV-6: Interaction Framework model presented by Abowd and Beale [*Abowd, 1991*]. This framework is used to measure the effectiveness of various visualization interfaces in this work.

In this framework, there are four components: the user (*U*), the system (*S*), input (*I*), output (*O*); as well as four translations between these components: *articulation, performance, presentation, observation*. Users through *articulation* generate inputs for a system detailing the requirements of their current tasks. The system *performs* a set of function operations on these inputs and generates a set of outputs. These outputs *present* a possible solution of the users' input queries. Users must finally *observe* and interpret these system outputs, updating their task model as necessary. This cycle is iterated over as many times as necessary until all task objectives have been satisfactorily met. Each translation step can be assessed for its effectiveness with respect to the overall interaction. Effectiveness of the four translation steps is measured by their *articulatory distance*, *functional distance*, *expressive distance*, and *observational distance* respectively (Figure IV-6). The summation of these four distances measures the effectiveness of the overall interaction or its "*semantic distance*". *Semantic distance* refers to the degree which user goals are fulfilled by the interaction. A large semantic distance means that the goals are not achieved well and a small semantic distance means that the goals have been satisfied acceptably.

Articulatory distance measures the ease with which users can specify their desires to the system. For visualization systems, articulatory distance measures the amount of input device manipulation required from users. A visualization technique that requires a great deal of user input has high articulatory distance and vice versa. *Functional distance* refers to whether the system possesses software functions or procedures capable of achieving user tasks. In our case, functional distance refers to whether the object definition functions (enumeration and functional description) and the transformation functions (data, mapping, graphical, and rendering) presented in the previous two chapters are sufficient to support basic data exploration tasks. *Expressive distance* determines whether sufficient feedback or information is provided to users to solve the input tasks. "*Sufficient feedback*" may mean whether sufficient data concepts and relationships are provided to solve the input tasks, whether the presentation reflects all facts contained within the data set, whether false information is introduced, and/or whether all information contained within the visualization is displayed at all times[1]. Finally, *observational distance* refers to the ease with which a user can interpret system feedback. Specifically, *observational* distance measures the effectiveness of the visual objects, visual properties, and visual compositions used to fulfill the input analysis tasks[2].

Our design dimensions measure either articulatory, expressive or observational distance. As for functional distance, we have supplied our system with all the necessary object definition (e.g. *enumeration, functional description, set operations*) and transformation functions (e.g. *addition, subtraction, assignment, grouping, mapping data to graphics*) needed to perform the basic set of data exploration tasks used in previous automatic system research and which we find interesting on our own work. Thus, the functional distance measure is not pertinent in our case. More generally, completeness of our object definition and transformation functions with respect to existing visualization systems was discussed in the previous two chapters. Completeness of our system in terms of task coverage is described in appendix C-3. We will now describe the various dimensions that may be used to estimate articulatory, expressive or observational distances as well as how these distances may be used to gauge the effectiveness of a design that favors data transform techniques versus one that favors mapping techniques.

---

1 The expressive distance described here is an expansion of Mackinlay's expressiveness criteria [*Mackinlay, 1986a, 1986b*].

2 Observational distance corresponds largely to Mackinlay's effectiveness criteria [*Mackinlay, 1986a, 1986b*]. In his dissertation Mackinlay presented a set of effectiveness heuristics that ranked different graphical properties (perceptual operations) based on their perceptual accuracy.

IV-2.1 Articulatory Distance

Articulatory distance increases with the amount of user input required by a visualization interface. When using visualizations to solve data analysis problems, user input is required for two primary purposes: 1) task clarification/alteration, and 2) data navigation as is shown in Figure IV-7.

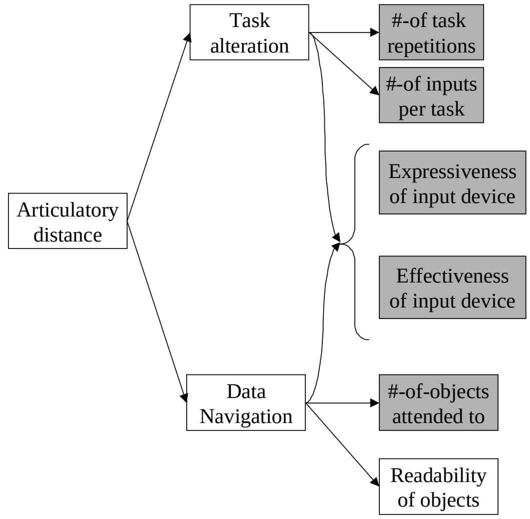


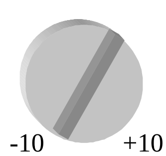
Figure IV-7: Breakdown of articulatory distance. Gray highlighted rectangles indicate the dimensions that are taken into account in our prototype automatic presentation system described in chapter V.

a) *Task alteration/clarification load*: To solve some data analysis tasks a user may need to provide task arguments to the computer system interactively. The amount of articulatory load required depends on the number of times a task needs to be repeated multiplied by the number of inputs per task. When a task is repeated many times, it becomes very important to reduce the articulation load of each iteration, even at the cost of losing flexibility.

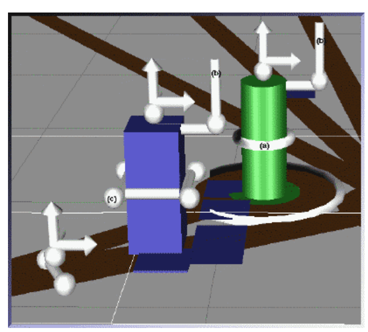
b) *Data navigation load*: Data navigation operations are commonly required for larger data sets where there is clutter and occlusion in the graphic design resulting in high observational distance. It is possible to lower observational load by limiting the amount of information that is shown to the user at any one time. The disadvantage, of course, is that users must navigate to different pertinent information slices through input devices. Data navigation depends on the number of objects that must be attended to and the readability of those objects (e.g. whether they are occluded, too small to interpret, or surrounded by high ink density. *Readability* issues are explored in detail in appendix F). The more objects we need to attend to, the greater the likelihood that we must perform more navigation. In addition, the less visible or readable the objects are, the more effort we must expend to get them to a readable state. In our prototype designer, we only estimate data navigation load by the number of objects attended to, leaving the more complex readability issues for future work.

c) *Expressiveness of an input device*: An input device is considered expressive of a particular visualization function if it can be used to generate all the inputs required by that function **and only** those inputs. A *menu* for example, is only expressive of discrete values, i.e. it can only be used to pick from a finite set of set of strings or numbers, but cannot be used to generate continuous input values. A *slider*, on the other hand, can be used to generate both continuous and discrete input arguments. *A text window* is very flexible and can be used to specify any input argument type. However, it is generally not very expressive because it does not indicate to users what the acceptable input arguments are unlike the *slider* and *menu* devices. I.e. users may very easily enter invalid input values when using *text-windows*. Our design system only allows the use of input devices that are expressive of the input data or arguments required. Unlike the previous two dimensions which are quantitative measures, expressiveness of input devices is implemented as a binary measure in our system.

d) *Effectiveness of an input device*: The effectiveness of an input device measures how easily users can manipulate an input device to generate the required task arguments. The effectiveness of an input device is most commonly measured by the amount of motoric energy expended by users for each input entry. Input devices such as buttons require very little motoric load because users only need to move the mouse over the button and click. Menus have a higher motoric load because we need to scroll down a list of choices in addition to choosing an entry with a mouse move and click. A text window has the highest motoric load because we need to move our hands over to the keyboard and type out our entries, which could take multiple keystrokes. Therefore, when the required input arguments can be expressed by a *button* input device (i.e. the input value range must be small and discrete), it should be chosen instead of *menus* and *text windows* because it is the most ***effective*** device.



(a) Dial



(b) SDM handles[*Chuah, 1995b*]

Figure IV-8: Input devices with different effectiveness properties

Input device effectiveness can also be measured by how easy they are to learn. Some input devices provide good *affordances* (or cues) to users indicating how they may be manipulated. For example, a *dial* or *knob* (Figure IV-8a) is an effective device for producing radial values because it provides good affordances for showing users that it should be rotated. In contrast, the virtual object handles provided in the SDM system (Figure IV-8b) are less effective because it is less clear how they should be manipulated and what they control. In our system we only account for effectiveness based on ease of use (level of effort) and not learnability. The input devices in our system are ordered in a list based on fewest manipulations to most manipulations. The system then picks the first expressive device in the list (i.e. the first expressive device that is most effective). Details of this process are described in chapter V.

In general, performing a task with data techniques requires more articulatory load because the task must be very explicitly stated (there can be no missing values). When there are unknown task arguments, the data technique designs require that users provide these missing arguments to the system. Thus, articulatory load is high because either we must provide several different initial task specifications, each containing a different task argument alternative or we must provide task clarification/alteration parameters during the data analysis process. The task specificity guideline in section IV-3.6 reflects this property of data techniques.

IV-2.2 Expressive Distance

Expressive distance measures whether sufficient data or information is shown to the user. What actually constitutes "sufficient data" may be interpreted in several ways: a) expressiveness of task, b) expressiveness of data, c) data correctness, and d) data presence, as is shown in Figure IV-9.

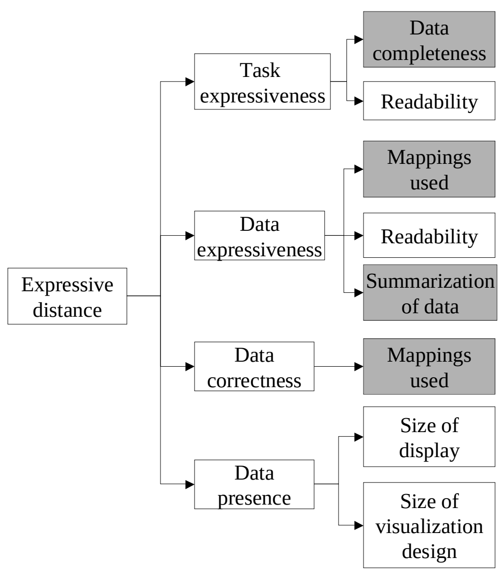


Figure IV-9: Breakdown of expressive distance. Gray highlighted rectangles indicate the dimensions that are taken into account in our prototype automatic presentation system described in chapter V.

a) *Task expressiveness*: In order for a visualization to be expressive of a data analysis task, there must be a sequence of cognitive, perceptual, and motoric actions that users may perform on the visualization design that will result in a solution to their task. These sequence of actions can only be generated if the visualization design contains all the data concepts necessary for solving the task (data completeness) and presents this information in a way that is accessible to users (readability). For example to solve the desired airline-scheduling task (Task IV-1), the visualization may contain the *origin-city, destination-city*, *arrival-time* and *departure-time* of all flights as in Figure IV-1. Alternatively, it may include only those flights that fulfill our *city* and *meeting time* constraints as in Figure IV-3. Although these two visualizations contain different data concepts and attributes, both contain enough information for solving the airline-scheduling task. Thus, Figure IV-1 and Figure IV-3 both have data completeness. On the other hand a visualization that only shows the *origin* and *destination* cities of all flights (leaving out their arrival and departure times) is insufficient for picking flights with the minimum *total-downtime* because no time information is provided to users. Such a design, therefore, is not expressive of the airline-scheduling task.

To achieve task expressiveness, a visualization must not only contain all the task information, but this information must be accessible to users. Sometimes, due to problems such as occlusion, dwarfed objects[3], or display density, some of the encoded information may not be visible or readable by users. For example, Figure IV-1b is data complete but not task expressive because some of the encoded information cannot be accessed due to object occlusion. We discuss readability issues in the appendix F as well as outline how they can be addressed using graphical and rendering transforms. In this chapter, we show how some readability problems may be avoided with appropriate combinations of data and mapping techniques as in Figure IV-3.

b) *Data expressiveness (Information loss wrt. original data set)*: A visualization is generated by processing and mapping a set of data concepts and attributes to graphics. We call the set of original data concepts and attributes the *original data set*. This data set is commonly attached to data characterizations that describe the concepts and attributes contained within the set, as well as the relationships among the data [*Mackinlay, 1986a, 1986b; Roth, 1990*]. These data characterizations help us structure the data so that we can generate better design solutions. Generally however, not all of the concepts or data characterizations contained within the original data set must be shown to solve a given set of analysis task(s). That is why data expressiveness is different from task expressiveness.

For example, data transform techniques may cause information from the original data set to be lost through data summarization or culling. In Figure IV-3 much of the flight data from the original data set was filtered out, thus Figure IV-3 is *task expressive* but not *data expressive*. Data technique designs are usually much less data expressive compared to mapping technique designs because they work by simplifying or summarizing data and only showing the results of those simplifications.

c) *Correctness of the visualization (Information integrity)*: The expressive distance of a visualization also depends on its correctness[4]. Certain graphical languages may imply facts about the encoded data values

[3] Problems with scale that prevent some values from being differentiated.

[4] The concept of data correctness was first introduced by Mackinlay [*Mackinlay, 1986a, 1986b*].

that are untrue. For example using *saturation* to represent an unordered attribute (e.g. *house neighborhood*) suggests a perceptual ordering when actually there is none. Figure IV-10 shows a set of houses, represented as marks arranged in a grid representation. The *saturation* of the marks indicates the *house neighborhood* attribute, which is an unordered attribute. However, because *saturation* is an ordered perceptual property, the visualization falsely shows that the *Pt.Breeze* neighborhood (most saturated) is ordered above the *Squirrel Hill* neighborhood (less saturated) which is ordered above the *Shadyside* neighborhood (least saturated).

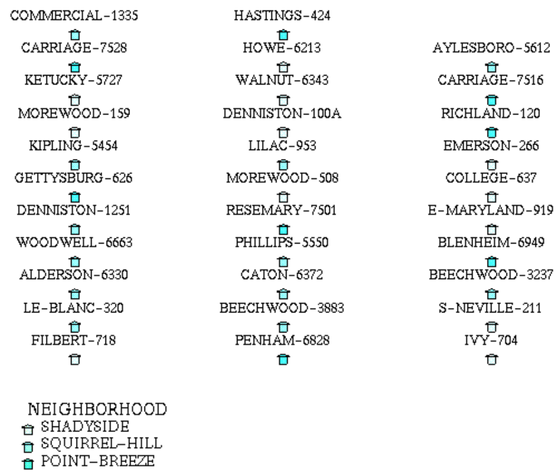


Figure IV-10: Encoding house neighborhood with saturation, This encoding has low data correctness because *saturation* is an ordered graphical property while *neighborhood* is not an ordered data attribute. By using *saturation* to encode *neighborhood* we are falsely implying an ordered set of *neighborhood* values when actually there is none.

d) Data Presence: Sometimes a visualization design is too large to fit within the CRT screen. When this occurs we must divide the visualization into segments and display sub-portions of it to users at different times. The visualization design is therefore only expressive of a piece of information for a limited time (i.e. temporary expressiveness). Data presence measures the ratio between the information shown per instance on the CRT screen with respect to the information within the entire visualization. Generally, a visualization with low data presence is also less expressive because only a small part of the total information can be seen at any one time. There are two ways to measure data presence: by calculating the ratio between average number of objects shown per instance and the total number of objects, or by calculating the ratio between visualization space per instance with respect to the entire visualization area. The lower the object or spatial ratios, the less data is shown and the greater the probability that users may miss some of the information and misinterpret the data contained within the visualization. When data presence is less than 1 (i.e. some information is hidden) users may find it necessary to store some information in short term memory to maintain context between the different information slices. This increases the cognitive load (observational distance) placed upon users.

Expressiveness criteria for visualizations was first introduced by Mackinlay [*Mackinlay, 1986a, 1986b*]. Mackinlay defined expressiveness as follows:

> *"A set of facts is expressible in a language if the language contains a sentence that encodes every fact in the set and does not encode any additional facts"*.

This definition covers *data expressiveness (b)* and *visualization correctness (c)*. The expressive distance dimensions presented in this section expand on Mackinlay's expressiveness criteria to include two other criteria: *task expressiveness*, and *data presence*. In our system, we account for all the expressiveness dimensions in Figure IV-9 except for *data presence*. Task expressiveness and correctness are implemented in our system as binary constraints (i.e. all designs generated by our system are task expressive and correct) and data expressiveness is implemented as a quantitative constraint.

IV-2.3 Observational Distance

Observational distance consists of cognitive and perceptual loads placed upon users when interpreting results from the visualization system. Perceptual load is determined by the number of perceptual operations that must be performed, and the difficulty of those perceptual operations. Similarly, cognitive load is determined by the number of cognitive operations that must be performed, and the difficulty of those cognitive operations, as is shown in Figure IV-11.

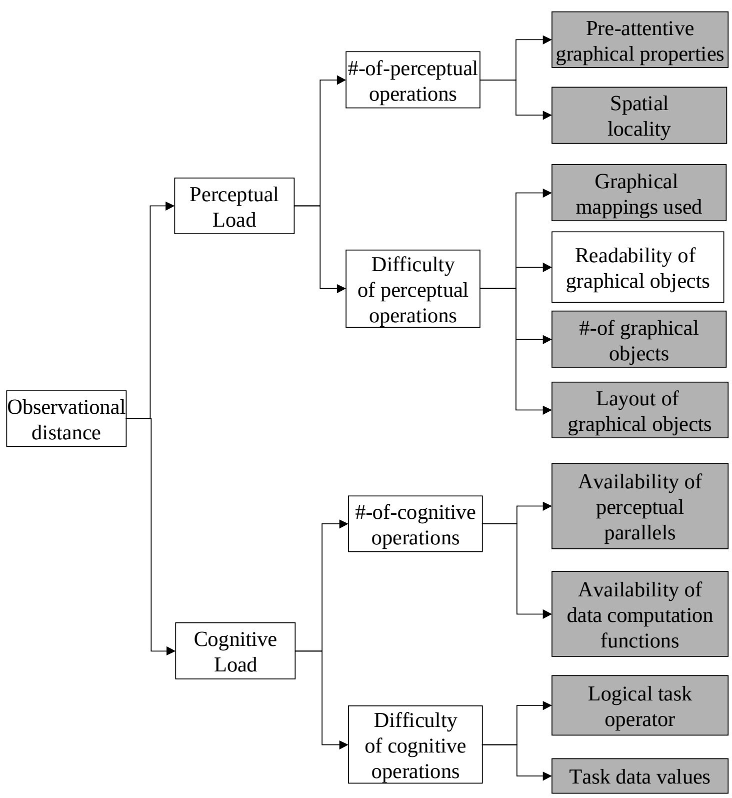


Figure IV-11: Breakdown of observational distance. Gray highlighted rectangles indicate the dimensions that are taken into account in our prototype automatic presentation system described in chapter V.

a) *Number of perceptual operations*: The number of perceptual operations required depends primarily on the effectiveness of the graphical mappings used to represent the data and the task. Certain graphical mappings (e.g. color) enable pre-attentive perception, which allows us to see certain common facts about a set of objects simultaneously (i.e. we need not attend to each object separately). Consequently, the number of perceptual operations required is significantly reduced. Pre-attentive vision is also very useful for quickly filtering out unrelated objects so that we only attend to the ones that are pertinent to our task. For example to find the first flight in the airline-scheduling task presented at the start of this chapter (Task IV-1) the user only needs to consider those flights whose arrival time is before the meeting (i.e. in Figure IV-1 we only consider lines which end before a certain distance to the right). All other flights may be perceptually filtered out. The number of perceptual operations can also be lowered by reducing the number of eye-movements that must be performed. This can be achieved by placing objects with related information together so we do not need to associate objects that are separated over large spatial distances. In our system we give preference to designs that effectively uses pre-attentive graphical properties and have good spatial locality (high level of graphical element integration).

b) *Difficulty of perceptual operations*: The difficulty of perceptual operations depends on the graphical representations and properties used to show the data as well as on the readability of those graphical representations. Different graphical mappings can result in simpler or more complex perceptual operations. For example, to solve an addition task, it is expeditious to map the task values to *stacked-bars* because judging *length* or *position* (e.g. looking up a bar *length*) is easier than computing sums of *size* or *length* (e.g. adding the *length* of two bars). Readability issues can also affect the difficulty of perceptual operations as is discussed in appendix F.

The perceptual complexity of a visualization is also dependent on the overall layout of the design and the number of graphical elements within it. To keep complexity low, we must ensure that the graphical elements and input device controls within the visualization are well integrated. In addition, we also want to ensure that there are not overly many graphical elements or controls, so that the visual interface does not appear too cluttered or confusing to the user.

c) *Difficulty of cognitive operations*: The difficulty of cognitive operations depends on the task. Tasks that require simpler mathematical operations, e.g. *addition* or *subtraction* or tasks that only require simple value comparisons can be solved with lower load cognitive operators. Other tasks such as computing *ratios*, *integrals* and *derivatives*, finding data trends, or identifying data relationships are harder to perform cognitively. The difficulty of cognitive operators may also depend on the data values involved in the operation. For example, it is more difficult to perform computation on numbers that have a higher number of significant figures, e.g. (200 + 300) vs. (273 + 329).

d) *Number of cognitive operations*: The number of cognitive operations required depends on how easily they can be offloaded onto our perceptual system. We usually want to keep the number of cognitive operations to a minimum, because they are usually much harder to perform and more taxing on users compared to perceptual operations. This is easy to accomplish when there are graphical objects capable of expressing the desired task data and relationships. The *addition* task, for example, has a close perceptual parallel - namely *stacked bars*. Thus, the cognitive load can be easily transferred onto the perceptual system. However, this is less true for more abstract computations like *log* and *exponent* which does not have a close perceptual parallel. However, because we consider data transform techniques in our automatic design process, we can offload these more complex tasks onto the computer system through data pre-processing operations. The advantage of data computation is that they offload the entire cognitive operation onto the computer system and only incur a small perceptual load from the user for interpreting the results. This is especially useful for cognitive tasks that cannot be easily mapped to perceptual operations.

IV-3 Data Techniques vs. Mapping Techniques Design Guidelines

The design dimensions given in the previous section provide useful guides for directing an automatic design system to more promising paths in the design space. However, the design dimensions alone are insufficient because some design dimensions are difficult to calculate or measure without additional perceptual and design knowledge. For example using color often reduces the number of perceptual operators and thus the observational distance of a design because it allows for pre-attentive perception. This information however cannot be deduced from the design dimensions alone. The fact that color allows for pre-attentive perception must be encoded into the designer as well. Thus in addition to the design dimensions, we present a set of higher level knowledge guidelines that capture how particular design decisions may affect the “goodness” dimensions of a design.

Previous work on automatic visualization design developed a set of *mapping technique guidelines*. Mapping technique guidelines capture knowledge on how data attributes should be mapped to graphical properties and objects. These guidelines describe the effectiveness of graphical properties for showing different types of data attributes. This could be based on whether the graphical property reduces the number of perceptual operations (e.g. because of pre-attentive perception) or the complexity of the perceptual operations. Mapping heuristics may also include structural heuristics that describe how data attributes should be mapped to objects and how new objects should be combined with existing ones. For example, integration of graphical properties within the same object or cluster of objects is preferred over spreading the properties over multiple regions, because integration reduces the number of eye movements that are required. Chapter V contains more details on how these heuristics can be translated into concrete constraints and design costs within an automatic design system. For more details refer to previous work on automatic visualization design [*Mackinlay, 1986a, 1986b; Roth, 1990*].

In this section, we focus on defining a set of design guidelines for making decisions between using data transform techniques versus mapping transform techniques to solve tasks. These guidelines were derived using the three distances (*articulatory, expressive,* and *observational*) described in the previous section. Each guideline helps reduce the semantic distance of a task by reducing one or more of these translation distances. Note that these guidelines are not meant to be a complete list of design principles, nor do we claim that they are applicable for all task situations. We do believe, however, that they are a reasonable set of rules for the data analysis tasks that we consider in this thesis. It is important to recognize that these design rules are not meant to replace the expertise of a graphic designer or an information specialist. However, by integrating such design knowledge into an automatic system we hope to enhance the computer system's ability to convey more complex information as well as reduce the more mundane and straight-forward design work that needs to be performed and free visualization designers to explore a much wider range of design alternatives.

IV-3.1 Accuracy

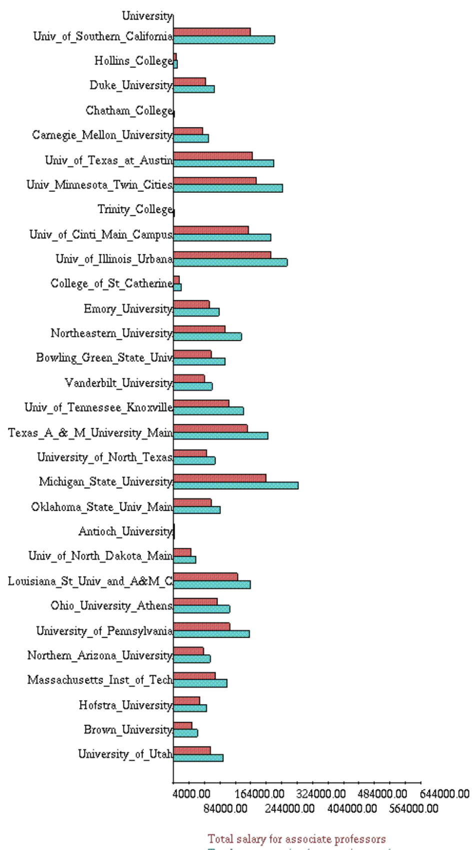


Figure IV-12: Graphic for determining the total benefits for associate professors by getting the difference between total compensation (blue bar) and total salary (red bar)

Different tasks require different degrees of accuracy. When tasks require a high degree of accuracy, there is a preference for using data transform techniques. This is because to get the same level of accuracy through mapping techniques (i.e. perceptual processing), we would need to encode the data as text and then offload the computation process to the user's cognitive system. For example, suppose we were considering a group of universities for possible associate-professorship positions and we want to determine the total benefits given out to associate professors in those universities. In order to get the total benefit values we must determine the difference between the *total_compensation_associate_professor* and *total_salary_associate_professor* attributes in our data set. One possible mapping design to achieve this task is to represent the *compensation* and *salary* values with two sets of bar lengths as in Figure IV-12. However, if we wanted to determine the *total benefit* figures with high accuracy Figure IV-12 is inappropriate because bar lengths can only show the results with a limited amount of precision.

University	Total benefits for associate professors
Univ_of_Southern_California	62658.00
Hollins_College	3078.00
Duke_University	21456.00
Chatham_College	1272.00
Carnegie_Mellon_University	17250.00
Univ_of_Texas_at_Austin	53997.00
Univ_Minnesota_Twin_Cities	67860.00
Trinity_College	876.00
Univ_of_Cinti_Main_Campus	56990.00
Univ_of_Illinois_Urbana	42120.00
College_of_St_Catherine	4187.00
Emory_University	24603.00
Northeastern_University	42640.00
Bowling_Green_State_Univ	34556.00
Vanderbilt_University	20640.00
Univ_of_Tennessee_Knoxville	37700.00
Texas_A_&_M_University_Main	53848.00
University_of_North_Texas	21721.00
Michigan_State_University	83980.00
Oklahoma_State_Univ_Main	24700.00
Antioch_University	1635.00
Univ_of_North_Dakota_Main	12152.00
Louisiana_St_Univ_and_A&M_C	34830.00
Ohio_University_Athens	31812.00
University_of_Pennsylvania	50160.00
Northern_Arizona_University	18432.00
Massachusetts_Inst_of_Tech	29913.00
Hofstra_University	18602.00
Brown_University	14400.00
University_of_Utah	32802.00

Figure IV-13: Data computation design for accurately computing total benefits for associate professors

University	Total compensation for associate professors	Total salary for associate professors
Univ_of_Southern_California	265854.00	203196.00
Hollins_College	13824.00	10746.00
Duke_University	108472.00	87016.00
Chatham_College	6036.00	4764.00
Carnegie_Mellon_University	95358.00	78108.00
Univ_of_Texas_at_Austin	262961.00	208964.00
Univ_Minnesota_Twin_Cities	286884.00	219024.00
Trinity_College	5220.00	4344.00
Univ_of_Cinti_Main_Campus	255020.00	198030.00
Univ_of_Illinois_Urbana	298480.00	256360.00
College_of_St_Catherine	23744.00	19557.00
Emory_University	121422.00	96819.00
Northeastern_University	179140.00	136500.00
Bowling_Green_State_Univ	136528.00	101972.00
Vanderbilt_University	104160.00	83520.00
Univ_of_Tennessee_Knoxville	185250.00	147550.00
Texas_A_&_M_University_Main	248888.00	195040.00
University_of_North_Texas	111650.00	89929.00
Michigan_State_University	327522.00	243542.00
Oklahoma_State_Univ_Main	125723.00	101023.00
Antioch_University	6855.00	5220.00
Univ_of_North_Dakota_Main	60636.00	48484.00
Louisiana_St_Univ_and_A&M_C	203949.00	169119.00
Ohio_University_Athens	148938.00	117126.00
University_of_Pennsylvania	199440.00	149280.00
Northern_Arizona_University	99456.00	81024.00
Massachusetts_Inst_of_Tech	142662.00	112749.00
Hofstra_University	90652.00	72050.00
Brown_University	65900.00	51500.00
University_of_Utah	132594.00	99792.00

Figure IV-14: Pure mapping design for accurately computing total benefits for associate professors

In order to perform the task more accurately with mapping techniques, we need to encode both value sets as text as in Figure IV-14. The observational distance for such a design is very high because we need to cognitively compute the *compensation* and *salary* differences for each university. It is much more effective, in this situation, to perform the task with data transform techniques and only present the computed differences to users as in Figure IV-13.

Table IV-1 compares the semantic distance between the pure mapping design (Figure IV-14) and the data computation design (Figure IV-13) based on the dimensions presented in the previous section. Table IV-1 shows that the mapping design has a much greater observational distance. This is because in the mapping design, the user needs to look up each of the text values, thereby resulting in *2n* perceptual lookups where *n* represents the number of universities. Given that the perceptual difficulty of a lookup is *p*, the perceptual load is *2np*. Apart from the perceptual load, there is also a cognitive load (*nc*) for mentally computing the difference between the *total compensation* and *total salary* values for each university (where *c* represents the difficulty of the mental difference operation). Also note that mentally computing differences is significantly more difficult than performing a perceptual lookup so $c >> p$. On the other hand the data computation design only requires a single perceptual lookup for each university, so the observational distance is *np*, which is substantially lower than the observational distance of the mapping design which is $2np + nc$. Thus when a task needs to be performed accurately, we assign a higher cost to the mapping solution and a lower cost to the data computation solution.

	Articulatory		**Expressive**			**Observational**	
	Task Clarification	Navigation	Task	Data	False	Perceptual	Cognitive
Data Design (Figure IV-13)				*2ne*		*np*	
Mapping Design (Figure IV-14)						*2np*	*nc*

Table IV-1: Semantic distance for computing the total benefits for associate professors

Both visualizations do not require any user input (i.e. no articulatory distance). In addition, they are expressive of the difference task and do not show any false information, so their task and correctness expressive distance is nil. However, the data computation design has a greater data expressive distance compared to the mapping design. In the data computation design, the original data has been summarized and it is no longer possible to extract the *total_compensation_associate_professor* and *total_salary_associate_professor* figures from each university. Thus, *2n* facts have been lost.

There are other tasks that require 'fuzzy' accuracy. For example, a person looking for houses in the *Shadyside* area may want to include some houses on the area boundaries even though they may technically fall within other neighborhoods. It is difficult to model such 'fuzzy' accuracy within the computer, and thus the articulatory distance for such tasks are large. Consequently, it is more appropriate to map the data to graphics so users can perceptually determine the appropriate level of 'fuzziness' for the task. This issue also relates to the task specificity issue, which we describe in section IV-3.6.

IV-3.2 Intermediate Tasks

When performing a complex task, we commonly need to break it down into several simpler tasks. For example, suppose we want to determine the total benefits given to full professors as well as associate professors for a set of universities. This operation can be decomposed into two *difference* operations between *total_compensation_full_professor* and *total_salary_full_professor,* as well as *total_compensation_associate_professor* and *total_salary_associate_professor* to get the *total benefits* for each faculty type. An *addition* operation is then applied to the two *total benefits* results as is shown below.

```
(Compute  Addition,
          (Compute  Difference,
                    total_compensation_full_professor,
                    total_salary_full_professor )
          (Compute  Difference,
                    total_compensation_associate_professor,
                    total_salary_associate_professor ) )
```

Task IV-2: Task for determining the total benefits given out to full professors and associate professors.

In Task IV-2, the *difference* operation produces intermediate results that are subsequently used by the *addition* task to produce the final result. As such, the *difference* tasks are not interesting in and of themselves. Tasks whose results are further processed by other tasks are called *intermediate tasks*. Intermediate tasks should be performed with data computation because they simplify the final graphic design by summarizing part of the data and hiding information that does not directly pertain to the main task. This reduces the amount of clutter within the graphic as well as the amount of perceptual interpretation that must be performed, without removing any of the information pertinent to our primary goal. Thus for intermediate tasks, the data solution is given a lower cost than the mapping solution which gets a higher cost in addition to the cost of the extra data attributes that need to be mapped.

For example, consider Figure IV-16 and Figure IV-15, which shows two visualization designs for solving the total benefits task (Task IV-2). In Figure IV-16 (pure mapping design) *total compensation* and *total salary* of each faculty type are mapped to the heights of four bars for each university. To solve Task IV-2 users must compare the lengths of the first two bars to get the *total benefits* for full professors and the lengths of the next two bars to get the *total benefits* for associate professors. This generates a perceptual load of $2p_2$, where p_2 indicates the perceptual cost of each difference comparison. Apart from perceptually estimating the length differences, users must also determine their combined lengths (i.e. the *total benefits* from both faculty types). This results in an additional load of p_3 where p_3 is the cost of estimating the combined length differences and then translating that back into a total benefit value. Thus the total observational load for each university, using the mapping design (Figure IV-15), is $2p_2+ p_3$. These perceptual operations need to be performed for each university so the total perceptual load is $n(2p_2 + p_3)$, where n is the number of universities.

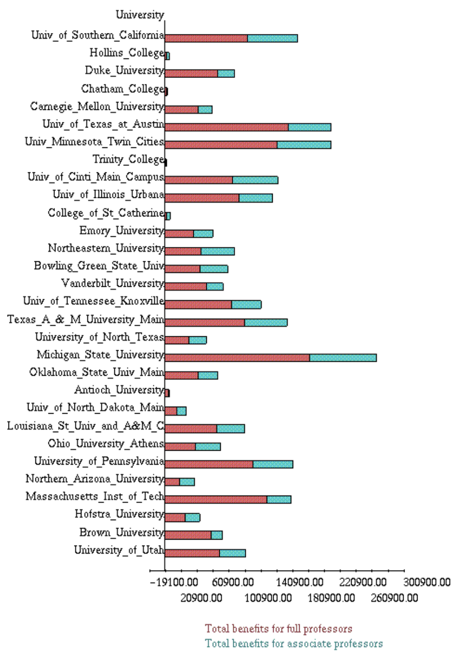


Figure IV-15: Data computation design for computing total benefits for associate professors and full professors. In this case both total benefits have been pre-computed and are shown as stacked bars.

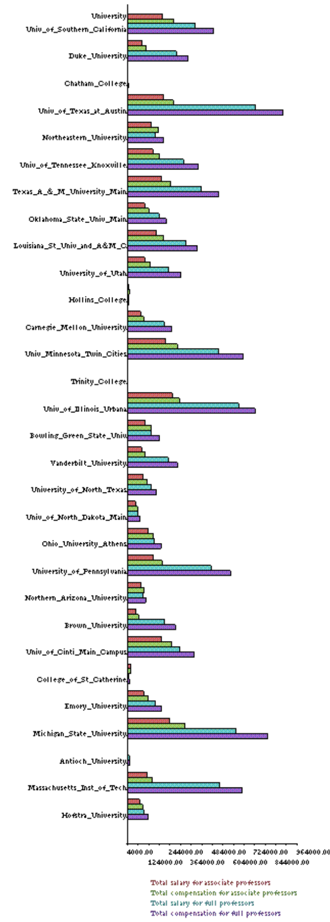


Figure IV-16: Pure mapping design for computing total benefits for associate professors and full professors. In this case, we need to perform the entire task perceptually. Initially we must get the bar differences of the first two bars (red and green) and the last two bars (blue and purple). We must then sum up these differences to get the total benefits.

In Figure IV-15, the *difference* intermediate tasks have been performed with data transform techniques, and the *total benefits* for each faculty type are represented as stacked bars. The total benefits from each university can be determined by simply looking at the height of each stacked bar. In this case, the perceptual load is only np_1, where p_1 measures the cost of a perceptual look up (i.e. looking up the bar length value from the *x-axis*). The cost of estimating total benefit values from two combined length differences in Figure IV-16 (p_3) is clearly more difficult compared to the axis value lookup (p_1), thus $p_3 >> p_1$. The observational load for the mapping design [$n(2p_2 + p_3)$] is therefore greater than that of the data computation design (np_1).

	Articulatory		Expressive			Observational	
	Task Clarification	Navigation	Task	Data	False	Perceptual	Cognitive
Data Design (Figure IV-15)				*4ne*		np_1	
Mapping Design (Figure IV-16)						$n(2p_2 + p_3)$	

Table IV-2: Semantic distances for total benefits task (Task IV-2)

By performing the intermediate tasks (i.e. difference tasks) with data computation, we reduce the number of values that need to be shown by at least half. Rather than having to show the *total_compensation_full_professor*, *total_salary_full_professor* , *total_compensation_associate_professor,* and *total_salary_associate_professor* data attributes (as was done in Figure IV-16), we only show *total_benefits_full_professor* and *total_benefits_associate_professor* in the data computation design in Figure IV-15. Therefore there is less clutter in the display and less output space is required. However, the data expressive distance is also higher because of data filtering. Nevertheless, since the *difference* tasks are intermediate tasks summarizing and hiding their origin data values is appropriate because the tasks are only important for the results they generate in service of the main *addition* task.

IV-3.3 Availability of Perceptual Operations

Certain tasks can be easily offloaded onto the perceptual system without adding much, if any, observational distance. Some examples are *addition* and *subtraction,* which can be mapped to *stacked bars* and *overlapping* or *interval bars* respectively. In each of these cases, the task results are perceptually summarized onto one graphical feature. The results of the addition task are summarized by the stacked bar *heights* and the results of the subtraction task are summarized by the interval bar *lengths*. Certain abstract mathematical tasks (e.g. logarithmic or exponential computation) do not have any perceptual parallels and cannot be offloaded onto the perceptual system. Such tasks also tend to have high cognitive loads, which results in large observational distances. For such tasks, data computation techniques can be used to offload the expensive cognitive computation onto the computer system.

Other tasks such as summarization tasks (e.g. *sums*, *mean*, and *median*), or getting the *minimum* and *maximum* values within a set, can be performed perceptually but require more perceptual effort from users compared to the *addition* and *subtraction* tasks. For example to find the maximum data value from a bar chart we would need to compare the *heights* of a set of bars and pick the tallest one. Unlike the *addition* and *subtraction* cases, the task result is not captured in a single perceptual value but rather has to be derived by considering a set of perceptual values.

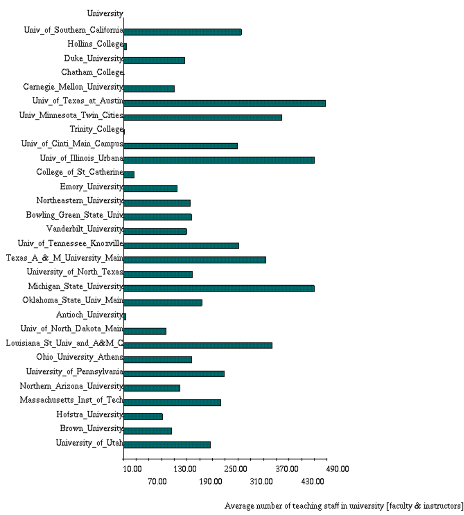


Figure IV-17: Data computation design for computing average number of teaching staff per university. In this case the average number of teaching staff has been pre-computed and the results are shown on the *x-axis*.

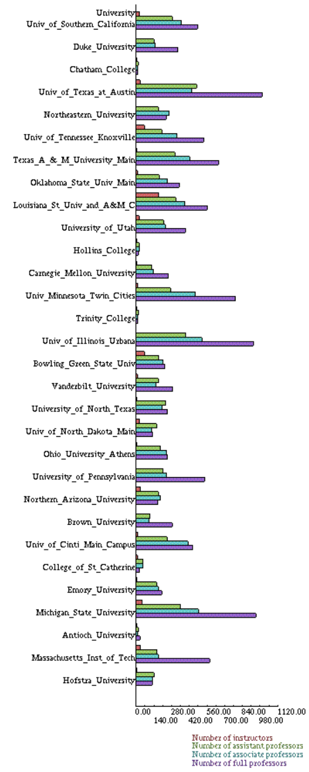


Figure IV-18: Pure mapping design for computing average number of teaching staff per university. In this case the average number of teaching staff must be perceptually estimated by finding an average line across each cluster of bars.

For example, suppose we want to determine the average number of teaching staff (including *full professors*, *associate professors*, *assistant professors*, and *instructors*) within a set of universities.

	Articulatory		**Expressive**			**Observational**	
	Task Clarification	Navigation	Task	Data	False	Perceptual	Cognitive
Data Design (Figure IV-17)				*4ne*		np_1	
Mapping Design (Figure IV-18)						$x(4np_2 + p_3) + np_1$	

Table IV-3: Semantic distance for finding average number of teaching staff for a set of universities

The pure mapping design (Figure IV-18) maps the number of each faculty type to a differently colored bar length. To estimate the average number of teaching staff, we pick a line that separates the four bars in such a way that the sum of bar lengths above the line is equal to the sum of lengths below it. This requires at least four bar difference estimations (to get the lengths above and below the average line) ($4p_2$)

and several comparisons between the lengths on top and below to determine their equality (p_3). When the lengths above and below are **not** equal we must re-estimate a new average line and repeat the process above (x). Once we get an acceptable average line we can look up the average value from the *y-axis* (p_1). The total perceptual load is therefore $x(4np_2 + p_3) + np_1$.

On the other hand, the data computation design (Figure IV-17) pre-processes the average number of teaching staff and only maps the results to bar lengths. Thus, users only need to perform n bar length lookups resulting in a perceptual load of np_1. The observational distance for Figure IV-18 (the mapping design) is much larger because there are more visual artifacts that must be attended to (4 bars instead of just 1) and because of the perceptual load needed for estimating and re-estimating the mean number of teaching staff (x). Thus for tasks that have good perceptual parallels (e.g. *addition*), the data and mapping designs are rated equally by our automatic design system. On the other hand, for tasks that **do not** have any perceptual parallels (e.g. *exp, log*) or for tasks that have high cost perceptual representations (e.g. summarization tasks), preference is given to the data computation solution.

IV-3.4 All to All Operations

Thus far, we have been considering tasks that compute or compare pairs of values, e.g. computing the difference between *total compensation* and *total salary* for each university faculty type. These pair-wise (*value-pair*) comparisons occur very commonly in data analysis, but do not represent the *only* task class. Another important class of tasks is all-to-all tasks. All-to-all tasks require each value in a set to be processed with all values in the second set, e.g. processing the *total compensation* values for each university with the *total salary* values of all other universities.

For a more realistic all-to-all task, consider an extended airline-scheduling task analogous to the one described in the airline-scheduling task in section IV-1 (Task IV-1), except here we take both *total-cost* and *total-downtime* into consideration.

> *"Given an origin and a destination city, the user "attempts to locate the two flights arriving in and departing from a layover city that offer the minimum amount of **cost** and `**down time**' between the flight times and the beginning and ending time of a scheduled meeting (in the layover city)".*

As before we assume that the origin city is *Los* Angeles, the layover city is *Chicago*, and the destination city is *Boston*. In addition, the meeting in *Chicago* is from 2 p.m. to 4 p.m.

This is an *all-to-all* task because we must compare all flights before the meeting with all flights after the meeting. To solve this task with data computation we pre-process the total downtime and total cost for all flight pairs as in Figure IV-19 (total downtime is encoded with *x-length* and total cost is encoded with *saturation*). If there are n_1 flights before the meeting and n_2 flights after the meeting, we must calculate and show values for $n_1 * n_2$ flights (i.e. $O(n^2)$ flights, where n is the total number of flights in the data set).

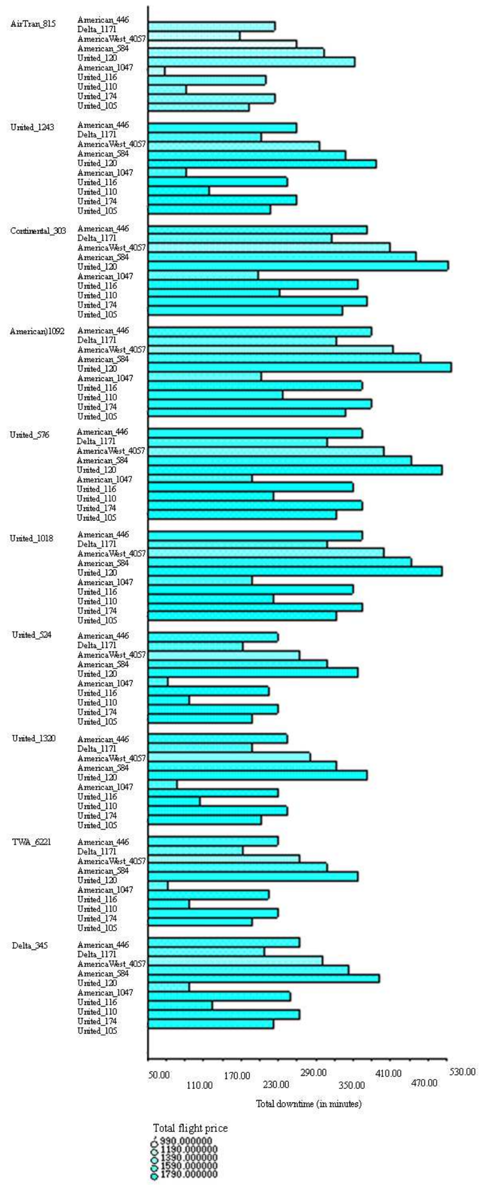


Figure IV-19: Data computation design for computing total downtime and total cost for all pairs of flights that fulfill our airline-scheduling criteria. Total downtime is pre-computed and encoded on the *x-axis* while total cost is pre-computed and encoded as *saturation*.

Although Figure IV-19 only contains information on 20 different flights, (10 * 10) = 100 values must be shown because of the *all-to-all* comparison and it is difficult to display all the information together clearly on the CRT screen. Therefore, we may need to navigate around the visualization space during the analysis session. Given that we have enough space on the CRT screen to show x elements, we need to scroll the visualization $O(n^2)/x$ times in order to get to all the information (i.e. there are $O(n^2)/x$ information slices). Each scroll requires moving the mouse over to the scroll bar (m), a mouse click on the scroll bar control (k), moving the scroll control (m) and a mouse release (k). The articulatory load is therefore $O(n^2)/x * 2(m+k)$ where m is the cost of a mouse move and k is the cost of a mouse click or release. To find the pair of flights with the best balance between total downtime and total cost, we choose the shortest bar with the lowest saturation in each information slice and compare these bars across slices to get the best one. This results in a perceptual load of $(O(n^2)/x) * p_1$ where p_1 indicates the perceptual cost of each bar search and comparison and n represents the number of flights.

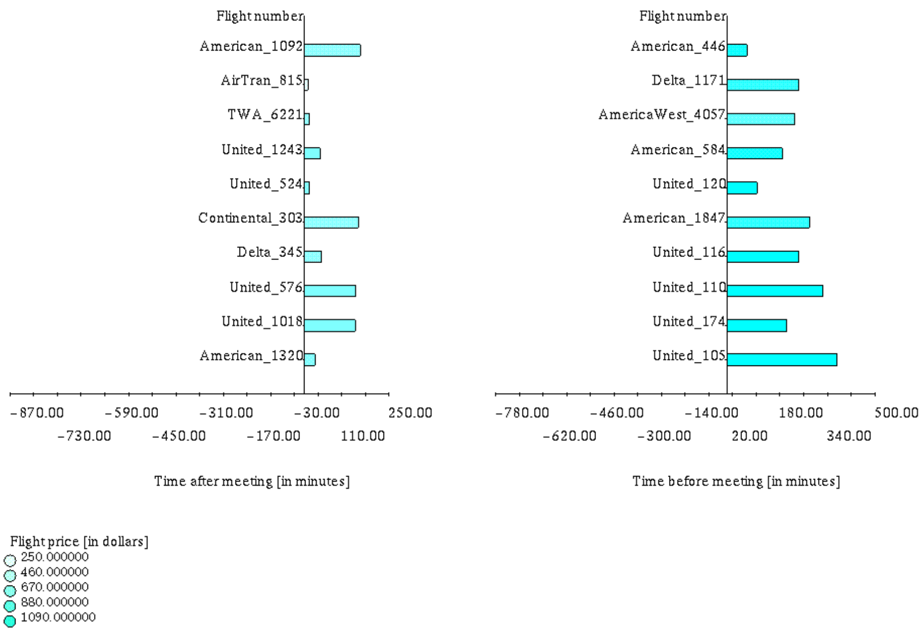


Figure IV-20: Mapping design for computing total downtime and total cost for all pairs of flights that fulfill our airline-scheduling criteria. *Time_after_meeting* is mapped on the *x-axis* of the left chart, *time_before_meeting* is mapped on the *x-axis* of the right chart, and *flight_price* is mapped to *saturation* in both charts.

Figure IV-20 shows the mapping techniques solution for solving the same task. In this design, the total cost and total downtime computations are **not** performed with data techniques. Instead, separate cost and downtime information are shown on both legs of the flight. The downtime and cost for the first leg of the flight is shown as *x-length* and *saturation* on the left chart in Figure IV-20 and similar information on the second leg is shown on the right chart. In order to find flights with low cost and low total downtime, we look for shortest, least saturated bar in each chart. Assuming that we can display x elements in the given amount of space, the navigation load would only be $(O(n)/x) * 2(m+k)$ because we only need to show at most $n_1 + n_2$ flights compared to the $O(n^2)$ flights in the data computation design shown in Figure IV-19. To find the best flights using Figure IV-20 we must look for two of the shortest and least saturated bars in each information slice, thereby resulting in a load of $2p_1$ for each slice. Total perceptual load therefore, comes to $O(n)/x * 2p_1$ where n is the number of flights, x is the number of elements that can be displayed on the CRT screen, and p_1 measures the difficulty of locating the shortest, least saturated bar in each information slice and comparing that bars across information slices

	Articulatory		Expressive			Observational	
	Task Clarification	Navigation	Task	Data	False	Perceptual	Cognitive
Data Design (Figure IV-19)		$O(n^2)/x * 2(m+k)$				$O(n^2)/x\ p_1$	
Mapping Design (Figure IV-20)		$O(n)/x * 2(m+k)$				$O(n)/x\ p_1 + 3p_2$	

Table IV-4: Semantic distance for an airline-scheduling task which balances total downtime and total cost

Based on Table IV-4, we see that the mapping design is far superior because when we try to solve an all-to-all task with data computation, we are forced to show many more data values, and this results in greater navigation and perceptual loads. Although in the mapping design there is the additional cost of having to process two charts, this cost is far outweighed by the processing needed for the large number of objects in the data computation design. I.e. for larger *n* the $O(n^2)/x$ factor in the data computation design far outstrips the $O(n)/x * 2$ factor in the mapping design. Although we have assumed that cognitive load is negligible, it can be fairly significant here, because we must compare data sets across several separate screens and as a result, we may need to maintain some context in short term memory across different information slices. Since there are more data slices that we need to traverse in the data computation solution, the related cognitive costs will probably be greater as well. Thus, for all-to-all tasks, our designer assigns a lower cost to the mapping solution particularly if the data set is large and if there are effective graphical representations for showing the task.

IV-3.5 Task Variation on Attribute

A big disadvantage of using data computation techniques to solve tasks is that they are limiting, i.e., they serve very specific purposes and cannot be adapted for a wide range of different goals. When performing a task through data computation we only show the results of the computation and hide the initial and intermediate values from users. Consequently, the resulting visualization design can only be used to solve its original, intended goal. For example, if we used data computation to perform an *addition* task we cannot also perform a *difference* task based on the computed results because the original values have already been summarized. Data simplification comes at the cost of inflexibility.

People, however, are much more versatile, and by mapping the source data onto graphics, we give end users greater flexibility in being able to solve a wider range of tasks with the same graphic. Thus when we need to solve a set of different tasks that operate on the same data attributes, we often end up having a higher observational distance if we use data computation operations. This is because a new set of values must be computed and visualized for each task variation. When the data complexity added by the data computation is greater than the cognitive load it subtracts, we should address the task by using mapping techniques.

Suppose we are studying election data for three different political groups over multiple states. Our task is to view the total number of votes in each state to determine its importance as well as to rank the political groups based on their individual number of votes. To fulfill this task with data computation (Figure IV-21) we process and represent the total number of votes by using the length of horizontal bars and align each bar with a pre-computed ordered list of the three political parties. This ordered list is represented by a series of dots, ordered from left to right, with each dot representing a different party. Color is used to encode the party type. In order to solve the task we look up the total votes from the *x-axis* (*np_1*) and the ranking information from the series of aligned dots (*np_2*).

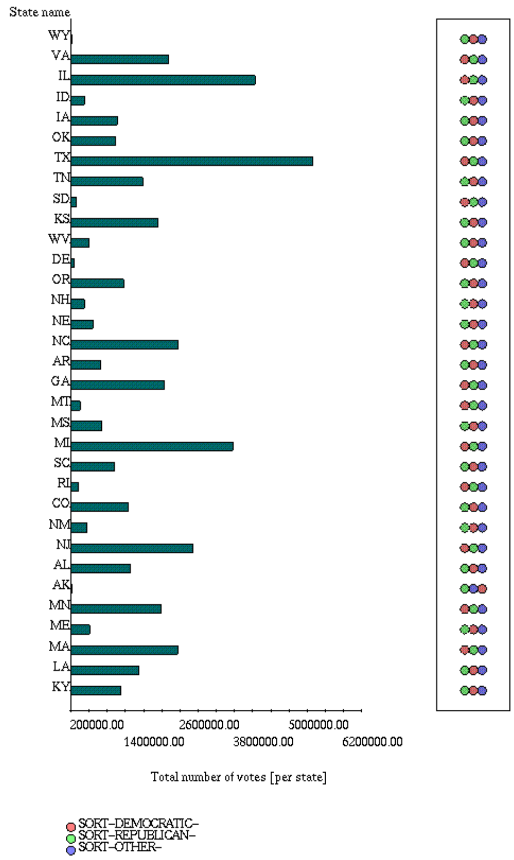


Figure IV-21: Data computation design for computing total number of votes in each state and ranking the three political parties based on the number of votes received. *Total_number_of_votes* has been pre-computed and is shown on the *x-axis* of the left chart. *Party_ranking* has also been pre-computed and is shown in the right table.

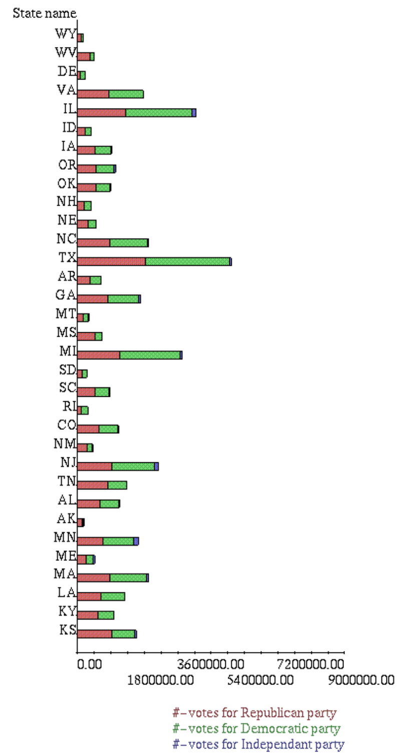


Figure IV-22: Mapping design for computing total number of votes in each state and ranking the three political parties based on the number of votes received. The *#_votes_for_Republican_party* is mapped to the *x-length* of the *red* bar, the *#_votes_for_Democratic_party* is mapped to the *x-length* of the *green* bar, and the *#_votes_for_Independant_party* is mapped to the *x-length* of the *purple* bar. Total votes can be derived by looking at the combined length of the stacked bar and party ranking can be derived by comparing the three differently colored bar lengths for each state.

Alternatively we could represent the task with mapping techniques by combining the number of votes received by each political party as a stacked bar (Figure IV-22). We can deduce the total number of votes as well as the group ranking from the same graphical representation (i.e. both tasks can be performed using the same graphical objects). We can look up the total number of votes from the combined height of the stacked bar (np_1) and determine the party ranking by comparing the lengths of the different divisions within each stack (np_3).

In this example, the mapping design (Figure IV-22) is preferable. Both designs have comparable observational distances; however, the mapping visualization is much more expressive. As is shown in Table IV-5, the data computation design has a *data expressive* distance of *3ne* (where *e* represents the expressive distance for each inaccessible number of votes figure) because we can no longer derive the original number of votes for each political group from the summarized results.

	Articulatory		**Expressive**			**Observational**	
	Task Clarification	Navigation	Task	Data	False	Perceptual	Cognitive
Data Design (Figure IV-21)				*3ne*		$np_1 + np_2$	
Mapping Design (Figure IV-22)						$np_1 + np_3$	

Table IV-5: Semantic distance for finding the total and individual sales

The data computation design has more clutter and shows less information compared to the mapping design without cutting down the observational distance of the task. This is because we were able to achieve two different tasks using the same graphical objects in the latter case while in the former case we had to encode the results of each task using two different sets of objects. Generally, when there are good perceptual parallels and significant task variation over the same data attributes, our automatic design system favors a mapping solution over a data computation design. When no effective perceptual parallels are available however, then our system weighs the cost of having more clutter (i.e. more graphical objects) and lower data expressiveness in the data computation design with the added cost to cognition and perception from having to perform the task perceptually with mapping techniques.

IV-3.6 Task Specificity

Tasks can be stated at many different levels of specificity. The higher the level of specificity, the cheaper it is to accomplish the task with data computation. When tasks cannot be fully specified at the outset, users must supply the missing task arguments to the data computation functions during the analysis process. Consequently, users are required to learn and use a set of input devices and interface artifacts, which increases the articulatory distance of the design. For example, in the airline-scheduling task presented earlier, we must know the *origin, destination,* and *layover* cities as well as the meeting time, in order to use data computation to solve the task. If we are unsure of these task parameters, we must supply then during analysis with input devices, causing a higher articulatory distance.

Sometimes, tasks cannot be described with high specificity because it is difficult to capture the task requirements or constraints. For example, suppose we want to find a “good” university to attend. We would like the university to have relatively low tuition cost, but a good record of accomplishment for graduating its students, and a good student/faculty ratio (i.e. low ratio). We might be willing to pay more tuition however if the university has an exceptionally high graduation rate or low student/faulty ratio. In general, we want to pick a university based on a balance of all three factors. Note that for this task it is difficult to specify the input parameters fully because there is no “correct” set of parameter values.

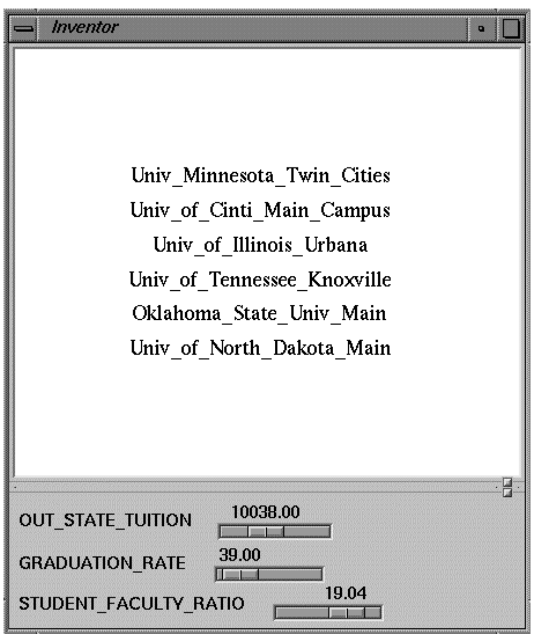


Figure IV-23: Data computation design for finding the best university based on *out-of-state-tuition, graduation-rate,* and *student-faculty-ratio.* Thresholds for each condition can be entered through the three sliders and those universities that fulfill the threshold conditions are pre-computed and shown.

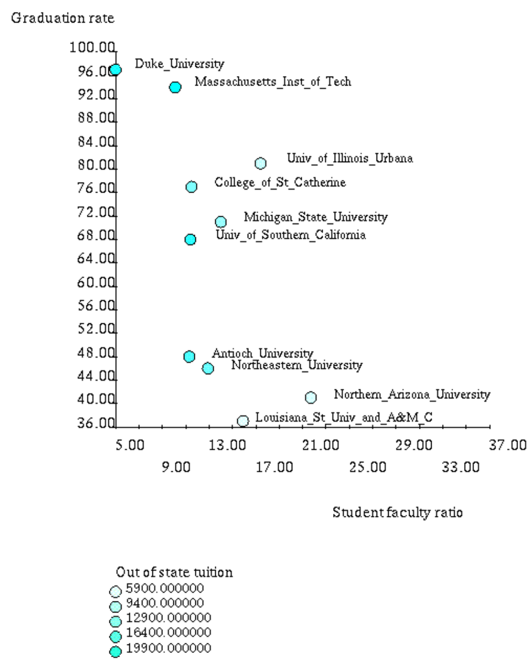


Figure IV-24: Mapping design for finding the best university based on *out-of-state-tuition, graduation-rate,* and *student-faculty-ratio*. *Student-faculty-ratio* is mapped to the *x-axis, graduation_rate* is mapped to the *y-axis,* and *out_of_state_tuition* is mapped to *saturation.* The best universities are those in the upper-left corner of the display, with low *saturation*.

In order to solve this task with data computation, we need to try out different parameter value combinations by entering them into the system using input devices. Figure IV-23 shows such an interface. It has three sliders for indicating the acceptable *tuition, graduation rate*, and *student/faculty* thresholds. The articulatory distance for this design depends on the number of parameter entries we must make. At the very least, we must perform three input device manipulations to feed in the initial threshold values. Each entry requires the user to place the mouse over the slider, click on the slider controller, drag the controller to the correct position, and release the mouse. This produces a load of $2(m + k)$ for each entry where m indicates the load incurred for a mouse move and k indicates the load for a mouse click or release. The minimum articulatory load for Figure IV-23 is then $6(m + k)$. Once we have entered these threshold values, all the universities that fulfill our constraints are shown as labels. Suppose that for each category half the universities pass our query, this would result in $n/8$ universities, thus the perceptual load for reading the resulting university names is $(n/8)\ p_1$.

Figure IV-24 shows how we can solve the same task with mapping techniques. In Figure IV-24, each university is represented as a *labeled mark*. *Student/faculty ratio* is mapped to the *x-axis*, *graduation rate* is mapped to the *y-axis*, *tuition cost* is mapped to the *saturation* of the *marks*, and the *university name* is mapped to *labels* next to each *mark*. Universities that fulfill our task criteria can be found by looking to the top-left corner of the chart (high *graduation rate*, low *student/faculty ratio*). We may slightly relax our constraints and consider the adjacent areas which indicate universities that have either lower *graduation rates* but good *student/faculty ratios* or high *graduation rates* but weaker *student/faculty ratios*. Within each of these areas, we are only interested in the less saturated *marks*, which indicate universities with lower tuition cost. Both the location and saturation lookups are perceptually pre-attentive and thus only two perceptual operations are required to find the appropriate universities (perceptual load of $2p_2$). Once we have identified the universities of interest, we lookup their names from the labels next to each mark. Assuming a uniform distribution, a quarter of the objects will be in the area that we are considering (i.e. $n/4$). In addition, we are only interested in the universities with lower tuition cost, i.e. the less saturated marks. Assuming that half of the universities in our area of interest are less saturated, we must attend to $n/8$ university labels. Thus, the total perceptual load for this design is $(n/8)\ p_1 + 2p_2$.

	Articulatory		**Expressive**			**Observational**	
	Task Clarification	Navigation	Task	Data	False	Perceptual	Cognitive
Data Design (Figure IV-23)		$6(m+k)$		$3ne$		$n/8p_1$	
Mapping Design (Figure IV-24)						$n/8\ p_1 + 2p_2$	

Table IV-6: Semantic distance for finding a house based on price, size, and distance to workplace

Based on Table IV-6, the mapping design is superior to the data computation design because the mapping design has no articulatory load and the additional two perceptual lookups required do not add much to the observational distance. For larger data sets, however, the mapping design could become cluttered and objects may be occluded (Figure IV-24). In this case, we would need to use input devices for navigation purposes, and semantic distance may end up being higher compared to the data computation design.

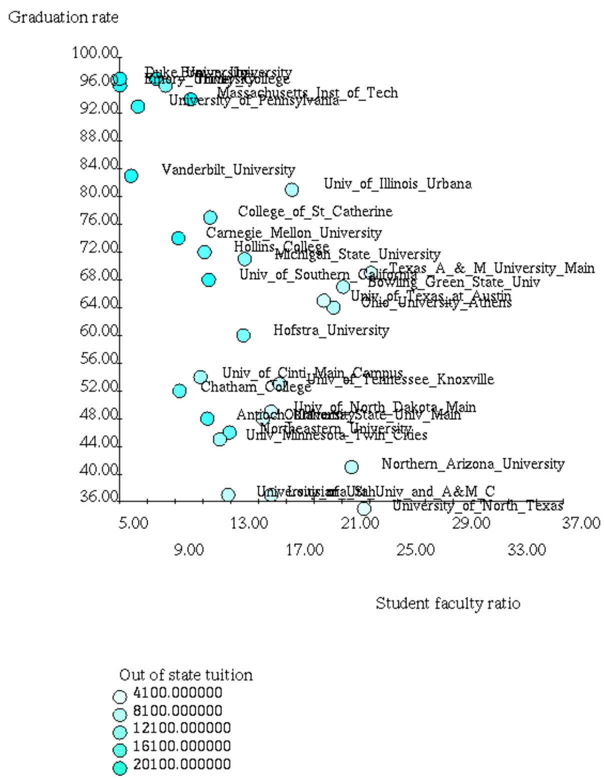


Figure IV-25: Identical design as Figure IV-24 but applied to a larger data set. As a result there is significantly more occlusion making it difficult for us to accurately view the *saturation* values on the *marks* as well as read the university names.

IV-3.7 Summary

In this section, we presented a set of high-level design guidelines that can help us determine whether to solve a task by mapping its data to graphics or by pre-processing it using data transforms. Details on how these more abstract design rules can be translated into concrete heuristics for an automatic visualization design system (AVID) are described in chapter V. Based on the guidelines presented here, we show how these data and mapping designs may be ordered according to their effectiveness at solving current user goals. In appendix C-4 we systematically explore the range of tasks available in our framework and describe how changes to the task will affect the design choices made by our automatic design system based on the design dimensions and guidelines presented in this chapter. In appendix C-5 we explore the possible space of data and mapping designs for a car purchasing task. For a complete description of our task language refer to appendix C-3.

IV-4 Conclusion

In this chapter, we described a set of design guidelines (section IV-3) that can be applied in an automatic design system for making decisions between using data transforms and mapping transforms. These design guidelines reduce the semantic distance for solving a task by either reducing articulatory distance, expressive distance, or observational distance. Designs that have many data computations usually have a smaller observational distance because some of the perceptual and cognitive load of interpreting the graphic has been offloaded onto the computer system and only the summarized task results are shown. On the other hand, data computation designs require complete task specifications (i.e. no task unknowns), which usually results in a greater articulatory distance. In addition, data computation designs also reduce the expressive capabilities of a visual design by filtering out all data that is not absolutely pertinent to the task. As a result, the range of perceptual tasks enabled by the design is reduced. Thus, when there is significant task variability over the same data attributes a mapping design is preferred.

High level user analysis goals generally consist of a mix of well specified subtasks (where it is clear what the goals of the task are and what the task parameters are) and non specific subtasks. Thus an effective design will most likely consist of a combination of data and mapping transforms. The blend of data and mapping transforms that is most appropriate is based on the interaction among the input task, data set, available graphical representations, as well as input and output hardware. Because the design decisions are based on a wide range of factors, it can sometimes be difficult to decide which sections of a task are more suitable for data computation and which are more suitable for mapping transforms. In this chapter we present a set of design guidelines that can help guide designers in making these decisions. These guidelines can also be translated into design heuristics and included into an automatic design system. We showed in section IV-1 that including data computation operations into the automatic design process significantly expands the visualization design space and the effectiveness of the system in being able to deal with data analysis problems. In appendix D, we analyze three more example tasks and systematically show the new set of designs that our work enables over previous research in this area. We also show that the rankings made by our system based on the design guidelines presented in this chapter conform to GOMS estimated performance time. Specifics on the architecture of our automatic design system are presented in chapter V.