Automating the Presentation of Information

Steven F. Roth Joe Mattis

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract In this paper, we explore the problem of creating effective computer displays of data contained in large information systems. We describe SAGE, an intelligent system which assumes presentation responsibilities for other systems by automatically creating graphical displays which express the results they generate. We describe SAGE's architecture and the methods by which it creates presentations. Implementing SAGE required identifying and representing a larger set of task and data characteristics than had been explored previously. We also describe several prototypical decision support and management systems which use SAGE in different ways to support the presentation needs of end users and system developers.[1]

AI Topic: Intelligent Interfaces, General Tools
Domain Area: Graphics Presentation
Language/Tool: Common Lisp and KnowledgeCraft
Status: A research prototype which has been integrated with several data-intensive research systems, and is under continuous development to handle more complex data and displays.
Effort: Four person years.
Impact: Theoretical value in representations of graphical knowledge. Practical value in ability to provide automatic presentation capabilities to developers and users.

1. The Need for Automatic Presentation

Decision support, database, statistical analysis tools and other forms of information management systems typically consider and generate large amounts of data which must be examined by managers and technicians. Because of the diversity of information and users' problem solving goals, it is difficult and costly to create interfaces which anticipate the display needs of each new application. As a result, displays are typically narrow in purpose, difficult to transfer to new applications, and inflexible in the way they interact with users. Displays are also poorly designed when system developers lack graphics design knowledge.

These difficulties are significant because the ultimate success of new systems depends on the ease with which users can access and evaluate their results. Application users are not unique in this need. Developers also need tools that flexibly display inputs and intermediate results to debug and verify the behavior of their systems. Some expert system development environments provide simple displays of production traces leading to a solution or tree diagrams of semantic relations. However, these environments do not provide graphical capabilities for simultaneously displaying quantitative and symbolic information representing different phases in the operation of a program. Even simulation packages, which are better suited to displaying quantitative data, require use of low level graphical languages or provide inflexible, predetermined interfaces.

Therefore, there is a need for tools to reduce the task of designing and programming displays to support data presentation in systems with quantitative and symbolic information. These tools should enable both developers and users to express and change their informational goals and have them realized automatically in effective displays.

This paper presents our work on SAGE, a system for automatic graphics and explanation, which was designed to meet these needs. SAGE is concerned with presenting quantitative and relational data using numerous variations and syntheses of 2-D static displays found in management and statistical graphics packages (e.g. bar and plot charts, node-link graphs, gauges, techniques using shape, color or size, tables). The next sections present a system overview and description of the concepts underlying its design. The last sections describe several experimental information systems which use SAGE's automatic presentation capabilities to support program development and communication of results to users.

2. Approach

The goal of our work was to develop a domain-independent, automatic graphics design and presentation system. Therefore, it was necessary to acquire graphical expertise and represent it in a form that is computationally useful. On the surface, graphics design appears to be an artistic, subjective process, perhaps not open to the level of description needed for automation. Although there have been excellent recent collections and discussions of

[1]This research has been supported by DARPA contract #F30602-88-C-0001 and by a grant from Digital Equipment Corporation.

graphics design [1], these have been enumerations of common errors made by designers and enlightening suggestions for improvements. In contrast, an appropriate AI perspective on graphics design would be one that forces us to think about representing the essential characteristics of users' data and information-seeking goals and representing the structural and functional characteristics of pictures. With explicit representations as a foundation, we can develop mechanisms for mapping from data and goals to graphical designs. Analogously, McDermott [2] has argued that AI can teach us much about software design once we represent the essential characteristics of application tasks, the functional mechanisms of computer programs, and methods by which we can map one to the other.

There was very little previous work on the problem of automatic presentation, and most of it addressed difficult interface problems and domain-specific solutions. Our work was based partly on APT [3], which was the first system to incorporate general graphical design capabilities from this perspective. SAGE went beyond APT in terms of the number and complexity of graphical displays and data types it considers and with its concern for users' information-seeking goals.

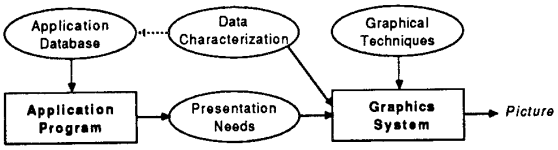

Figure 2-1: Relation Between SAGE & Applications

SAGE's architecture is illustrated in Figure 2-1. The application program in this architecture might be a query interface to a database or a program that retrieves information from any data-intensive system. Our emphasis on maintaining SAGE's generality and independence is conveyed by the relation between the application and the graphics system. SAGE's graphical reasoning cannot depend on knowledge of application data. Therefore, each program must communicate in a general language both its *presentation needs* (goals for viewing specific data), as well as a *characterization* of the data itself. SAGE selects and integrates *graphical techniques* which are expressive of the application's data characteristics and which satisfy the presentation needs.

3. Representing Characteristics of Data

A detailed discussion of data characterization is provided in [4, 5] and we only briefly outline it here. Our goal was to abstract a concise but complete set of characteristics to be used to distinguish the kinds of data each graphical technique can express (an approach initiated by [3]).

Data characteristics fall into several categories: features of database objects (e.g. resources, activities, dates, people, parts), features of relations among objects (e.g. the Start-date relation maps from activities to dates, the Has-part relation expresses the breakdown of departments), and definitions of relationships among relations (e.g. Start-date, End-date and Duration relations map to the beginning, end and size of an *interval*).

Characteristics of Database Objects include (1) the way they are ordered (i.e. quantitatively, ordinally or unordered), (2) whether they are measures of amounts vs coordinates, and (3) their domain (time, space, temperature, mass). All data types input by SAGE are characterized along these dimensions. All graphical styles in its library are also represented in terms of the values along each dimension which it can express. For example, the start-date relation maps from activities to dates. A date is characterized as a quantitatively ordered, coordinate in time. Both plot-charts and bar-charts are defined in SAGE as expressive of quantitatively ordered data, with a preference for expressing time along a horizontal axis. However, while plot-charts express either amounts or coordinates, bar-charts are only expressive of amounts. Notice that without the coordinate-amount distinction, SAGE would inappropriately present activity start-dates using a bar-chart, as in Figure 3-1.

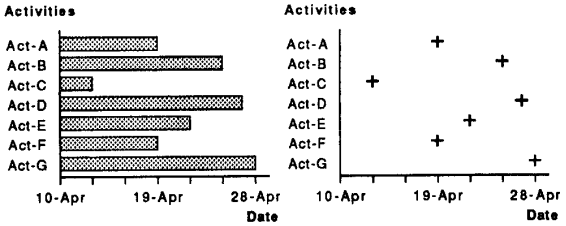

Figure 3-1: Dates are more appropriately shown with plot-chart than bar-chart.

Properties of Relational-structure describe the way relations map from elements of one set to another and include:

- coverage - whether every element of a set can be mapped by a relation to at least one element of another (e.g. are missing values possible?),
- cardinality - the number of elements to which a relation can map from an element of one set to those of another, and whether this number is fixed or variable, and
- uniqueness - whether a relation maps from an element of one set to a unique element(s) of another set (i.e. whether a relation uniquely partitions a set).

Relational characteristics are used to distinguish graphical technique expressiveness in the same way as object characteristics. For example, plot-charts are usually robust with respect to coverage, uniqueness and cardinality. In contrast, bar charts are not expressive of relations which do not guarantee every object has a value because they confuse missing bars with zero values.

Relationships Among Relations occur frequently in databases and are often viewed as n-ary relations. For SAGE to make use of conventional styles, like interval charts, it is necessary to represent these relationships as complex data types. SAGE has a set of complex data-types which provide a language for users to characterize their relations. For example, the start-date, end-date and duration of an activity must be characterized as the beginning, end and size of a time interval in order for SAGE to select the interval chart illustrated in Figure 6-3. Previous systems [3, 6] were unable to identify interval relationships and would generate two separate displays (one for the dates and another for duration). SAGE's characterization includes other complex data-types including *statistical abstractions (Mean, Standard Deviation)*, and *2-D coordinate-locations*.

4. Representing Characteristics of Goals

Thus far, we have discussed static database characteristics. A presentation system must also consider the role of an application's or user's goals in viewing data. Goals must often be communicated and changed by a user when presentations are requested and can greatly alter the appropriateness of graphical techniques and the ways they are combined. Several information-seeking goals became apparent in our observations of graphical display variations and analytic tasks across several domains. Presentations should vary depending on whether goals emphasize:

- accurate value lookup (which favors selection of text tables and makes stricter judgements about the acceptability of other techniques),
- comparison of values *within*, but not *between* different relations, which results in evaluation criteria favoring separate pictures rather than a single *composite* for several relations (e.g. presenting separate charts for the *expected-cost* and *actual-cost* relations, rather than merging them into a single picture),
- pairwise or n-wise comparison of relations for the same data set (which favors using the same technique to encode *expected-cost* and *actual-cost* and to position objects to be compared adjacently and in the same picture),
- distributions of values for a relation, (e.g. where concern for the frequency of occurrence of values and not the relation between values requires distributional displays),
- correlations among attributes (e.g. as shown by scatter plot or other composite techniques for which two or more properties of a single graphical object convey different relations), and
- data-locating needs for the objects referred to by a relation (e.g. different locating or indexing goals will dictate how the activities in Figures 6-2 through 6-4 are sorted or organized).

Casner [7] has also recently enumerated a set of primitive operators which can be used to construct detailed program-like sequences representing users' computational tasks. These operators guide the selection and integration of graphical techniques which allow perceptual operators to replace logical ones. In contrast, our approach has been to provide users with a concise language for communicating their goals to a presentation system. Nonetheless, the two approaches have led to overlapping and complementary characterizations of user goals.

In summary, we have developed methods by which application programs or end users can characterize their data and informational goals when communicating with a graphics system. These influence judgements of the expressiveness and usefulness of techniques, the ways they can be combined, and the distribution of information across pictures.

5. Representing Graphical Techniques & Design Methods

The design of displays is a three-stage process in which graphical techniques are selected, refined and integrated subject to the constraints of data and goal characteristics and the structural and functional properties of the techniques. The selection stage generates a set of candidate techniques (e.g. text-table, indentation, plot-chart, bar-chart, size, color, shape) which satisfy expressiveness criteria for the data to be presented. The candidates are ordered by the effectiveness with which they satisfy each information-seeking goal. The notion of an effectiveness ordering originated with [3] and was based on the ease with which graphical properties are perceived.

Our approach differs from [3] in that orderings must vary with different goals. Goals involving data comparison cause graphical techniques to be considered before textual techniques (with positional techniques considered first). The correlation goal results in a similar ordering, but with scatter plots and related correlational techniques considered first and textual techniques rejected. The distribution goal similarly favors goal-specific techniques, but then uses the comparison ordering when the former cannot be used. The accurate-lookup goal requires textual techniques to be considered first. Locate goals are used in combination with other goals and serve as an additional filter on techniques considered during the next stage.

Technique refinement, the next stage of design, supports goals by adding additional constraints on the layout and ordering of elements of displays. These constraints guide subsequent stages of synthesis and rendering. Sorting decisions are commonly employed by human designers, have powerful effects on the effectiveness of a display, but have not been addressed in previous systems. Sorts or groupings of axes, nodes, and otherwise unconstrained objects (e.g. color-coded circles) support goals emphasizing locating or searching for values. For

example, an axis encoding a nominal set can be sorted alphabetically or by the corresponding value in the other axis. The sort decision can be based on a locate goal (as in Figures 6-3 and 6-4.

Notice that sorting the vertical axis by date (in Figure 6-4) also supports viewing the *distribution* of the start-date values and serves as an alternative to distribution-specific techniques (when the latter cannot be used because of other competing goals). Relatedly, the use of scatter plots are rejected for correlations when competing goals occur to locate the set elements not represented (i.e. the labels of marks).

Therefore, the effect of goals at the refinement stage is to filter techniques considered previously and to constrain their instantiation. These decisions are made by a general module which considers the underlying structural properties of each technique and whether these support different goals. As a result, new techniques can be added without writing additional refinement programs. Next we consider the construction of displays for multiple relations.

Whenever a user requests that several relations be presented, SAGE attempts to integrate them in a single display. For example, a manager interested in allocating resources in a factory might request a single display which shows a set of operations, the resources they need, their scheduled starts and ends, and possibly the factory order on which they work, its importance and the customer for whom the order is destined. A system which creates six isolated displays for these six attributes would not be very effective.

SAGE integrates this information in a single display by combining candidate techniques using a set of synthesis or rules (an extension of the composition rules used by [3]). These rules specify the types of graphical objects within each technique that can be merged to form more complex pictures. Examples of synthesis rules include methods for:

- merging one or both axes of 2-axis charts (Figure 6-2),
- merging node labels in networks with labels of *retinal* techniques (e.g. color, shape, size) or gauges,
- merging two or more trees or hierarchical textual techniques to create multi-level networks or text hierarchies (as in Figure 6-1),
- combining hierarchical techniques with axes to create structured charts (as in Figures 6-1,
- merging text columns to form tables of multiple relations, and
- merging graphical objects which encode information using different compatible properties (e.g. to create size- and color-coded points along an axis as in Figure 7-1).

Before merging techniques, SAGE tests to determine whether doing so would violate any graphical constraints that are already imposed within the component pictures. For example, the color-coded interval bars of Figure 6-3 were created by merging two pictures: one representing time-intervals with bars and the other encoding resources with color. They could be merged because the color of the bars was unconstrained, while shape, size, and position are not constrained when one uses color to represent resources.

To support processes of technique selection, refinement, and synthesis, SAGE required a representation of the structural (syntactic) properties of graphical styles. This level is an extension of earlier work [3], and includes a definition of the objects that make up pictures, their graphical properties, and the spatial relations among them. For example, a bar-chart representation contains vertical and horizontal axes, a set of labels on each, and a set of rectangular bars or lines. Constraints include the fact that the axes are orthogonal, the order of elements along a nominal axis is free to vary but defaults to lexicographic, the bars are aligned relative to one axis and extend parallel to the other, the color of bars is free to vary, but their shape must be constant and their size is constrained by the need to refer to the axis.

In addition to this primarily syntactic analysis, it is also useful to consider the *pragmatic* or *functional* properties of graphical displays: (1) common communicative functions served across all picture styles by different graphical primitives and (2) higher functions served by different combinations of techniques merged by synthesis processes.

At the most basic level, every display contains objects which serve functions of either encoding sets or of expressing the correspondence among elements of two or more sets. The bars of charts express correspondence between elements of two axes. Similarly, the links of networks express correspondences among set elements encoded as nodes. Subtle spatial offsets encode correspondence among hierarchically arranged text elements (as in Figures 6-1 and 6-3).

The importance of these distinctions is that they guide synthesis operations: two objects can be merged if they perform the same function with respect to at least one common set of data. Therefore, it is possible to merge hierarchically arranged text with an axis, or the labels of gauges with nodes because these function as encodings of data sets. Similarly, color-coded points can be merged with bars because they both function as encodings of correspondence between sets.

This analysis also allows SAGE to consider the relation between synthesis operations and goals. Each goal defines a set of general pragmatic conditions which synthesized pictures must serve. For example, the correlation goal requires that a single object encode the correspondences for all relations (producing multidimensional encodings). The pairwise comparison goal (e.g. for comparing labor-cost & materials-cost) requires two groups of

correspondence objects using the same technique and with a single merged set of objects representing their set elements (the latter producing alignment). The most important point is that these decisions are based on the pragmatic analysis of **design representations** and not on the arbitrary selection of particular synthesis rules. Therefore, new rules can be added, and their results automatically considered by SAGE from the perspective of different goals.

In summary, we described an approach to automating the design and presentation of graphical displays of information. This has required representing the structure and function of components of graphical styles as well as conventions for merging them to achieve complex communicative goals. Creating a general application-independent system has also required developing a data characterization language for describing users' data. We turn now to discussion of our experiences using SAGE to support other systems.

6. Applications Using SAGE

SAGE is a research prototype implemented using KnowledgeCraft and Common Lisp. It has been integrated with several research systems to provide information presentation capabilities. Our experiences provided opportunities to test and extend the generality and completeness of our data characterizations and library of graphical techniques. The following cases demonstrate some of the displays SAGE designs and illustrates the range of ways it can be used: under direct control of end users or application programs and for supporting debugging and visualization of program behavior. Use of SAGE in conjunction with a text generating explanation system is presented elsewhere [4].

6.1. Iterative Refinement of Presentations

The first case involves using SAGE to provide presentations of information requested by users of a project management database query system. The example illustrates the iterative process by which managers refine both the information to be presented and their goals for viewing it across four successive displays. At each step, a manager queries the database and the system automatically passes the results to SAGE. The only direct user-SAGE interaction is for specifying different presentation goals for the data. Because the details of the query interface are independent of SAGE's operation, we list only the objects, relations and corresponding user goals that are SAGE's inputs, and omit the details of the query interaction.

In this case, a manager is reviewing the needs of groups in his company for computer resources, based on their scheduled activities in a new project. The purpose of the first request is to examine the departments, their divisions and activities, and the resources required by the activities. The manager requests a display where 1) departments and resources can be located easily and 2) relations involving resources and departments accurately represented. Figure 6-1 contains SAGE's inputs and the resulting presentation.

Show Relations: **For Objects:**
Has-part — Departments
Responsible-for — Departments, Activities
Requires-resource — Activities, Resources

Presentation Goals:
Lookup-accurately - Has-part, Responsible-for, Requires-resource
Locate-easily - Resources, Departments

SAGE's Response:

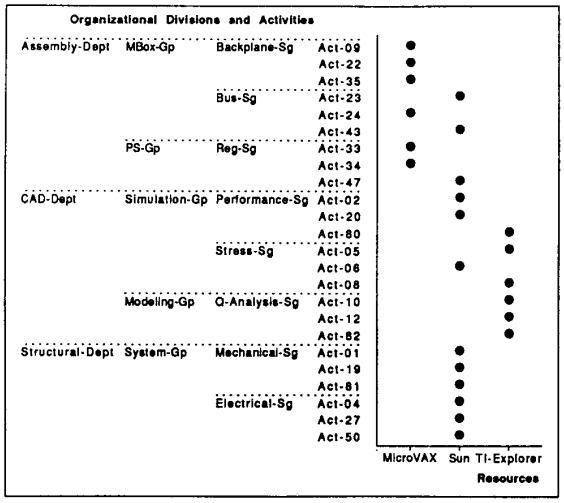

Figure 6-1: SAGE's Response to Initial Request

Generating this display required retrieving three ordered lists of candidate techniques that could express the three relations and satisfy the goals. Then, a compatible subset of these was selected that could be integrated using synthesis rules. First, SAGE selected three independent text tables because the *lookup-accurately* goals rate text as the most preferred technique. However, the *locate-easily* goals caused two pictures to be sorted incompatibly: one by resource, the other by department, thereby preventing them from being integrated.

An integrated picture was created by (1) encoding the Has-part and Responsible-for relations using two hierarchical-column techniques and merging them in a single hierarchy and (2) encoding the Requires-resource relation using a plot-chart technique and merging its vertical axis with the leaves of the hierarchical-column (merging activities). Notice that the plot-chart allows resources to be located easily along the horizontal axis and departments are located easily as sorted roots of the

hierarchical picture.

Next, the manager realizes the need to consider activity dates and durations, but not the parts of departments. In addition, because it is not easy to locate a particular activity in the previous display, the manager decides that the new display should strongly support this goal for all object types.

Show Relations:	**For Objects:**
Start-date,End-date	Activities, Dates
Duration	Activities, Number-of-days
Requires-resource	Activities, Resources
Responsible-for	Departments, Activities

Presentation Goals:
Lookup-accurately - *<All Relations>*
Locate-easily - *<All Objects>*

SAGE's Response:

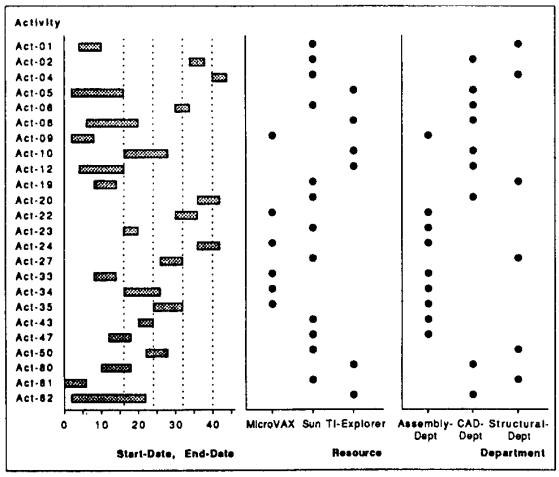

Figure 6-2: Satisfying Locate and Accuracy Goals

Figure 6-2 demonstrates: the encoding of date and duration relations with a single bar, the use of charts so that all objects can be located easily, and the integration of the three charts using the common vertical axis. This display is sufficient for the goal of finding many facts for one object at a time (e.g. the department, resource and start-date of Act-10). However, it is not helpful for exploring the relationships between when an activity occurs, which resource it uses and which department is responsible for it. Figure 6-3 shows a different display for the same information, but with the new goal of *viewing-correlations* among all relations.

SAGE realizes this goal by searching for an integrated display to represent the relations with a single set of graphical objects, each dimension of which encodes a different relation. Figure 6-2 is rejected because it violates this constraint (using three sets of graphical objects). SAGE attempts alternative techniques for each relation until a set is found which can be integrated in this way. Notice that the vertical position of the bars actually indexes two data sets: activities and departments. SAGE's representation allows this inference from the fact that a bar's vertical position indexes an activity and each activity's vertical position encodes its relation to a department.

Show Relations:	**For Objects:**
Start-date,End-date	Activities, Dates
Duration	Activities, Number-of-days
Requires-resource	Activities, Resources
Responsible-for	Departments, Activities

Presentation Goals:
Show-correlation - *<All Relations>*

SAGE's Response:

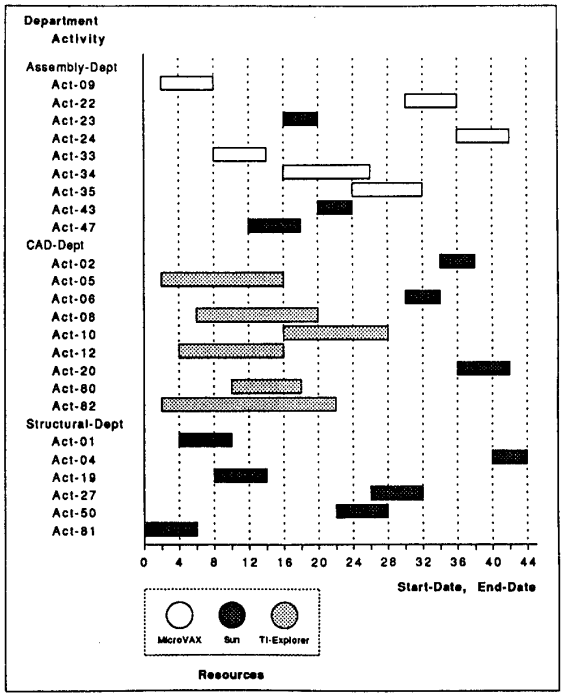

Figure 6-3: Satisfying Correlation Goals

Figure 6-3 is effective for viewing correlations or performing multivariate lookups (e.g. find the Suns used early by the Structural Department), it can be improved further for specific tasks. Specifically, the new task is determining the need for each resource on each date, and only secondarily, the related activities and departments. Figure 6-4 shows the results of this goal change. SAGE gave preference to the Requires-resource relation when selecting techniques so resources could be located easily. The indentation technique, which orders resources vertically, was used for this relation instead of color which

supports search less effectively than position. The Locate-easily goal for Dates of the Start-date relation leads to sorting of activities by Start-date.

Show Relations:	**For Objects:**
Start-date,End-date	Activities, Dates
Duration	Activities, Number-of-days
Requires-resource	Activities, Resources
Responsible-for	Departments, Activities

Presentation Goals:

Show-Correlation - *<All Relations>*

Locate-Easily - Resources, Start-date

SAGE's Response:

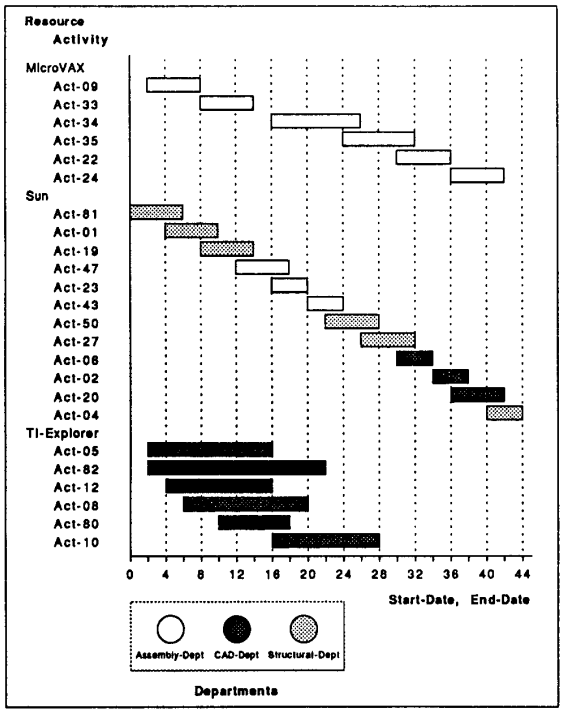

Figure 6-4: Satisfying Correlation & Locate Goals

In summary, this was an example of SAGE's support of a database query interface. It demonstrates the importance of user's goals and the iterative nature of goal and information specification in an exploratory environment. The combination of SAGE and a query interface has been ported to other systems and is an effective way to peruse quantitative databases.

6.2. Supporting Developers of Software Systems

SAGE has been integrated with the CORTES system [8], which schedules activities and resources in a factory. CORTES schedules opportunistically by iteratively: (1) determining the most critical resource and the most critical activity that requires it, (2) choosing a reservation date for that resource and activity, and (3) testing that the new reservation does not cause constraint violations elsewhere in the system. If this test fails, CORTES will select an alternative date and test again. When no satisfactory date can be found, CORTES will backtrack by undoing a previous reservation and rescheduling it. Thus, each reservation is a single state in a search-space, which is explored using a form of chronological backtracking.

Developers had difficulty analyzing CORTES behavior because each step in the search would change many types of information affecting the system's decisions. Our first attempt at supporting CORTES developers involved identifying their analysis needs, determining what collection of information would support each need, and organizing those needs into a menu of commands for gathering this information and sending it to SAGE. One need was the ability to peruse *correlations among certain attributes* of the search states. Specifically, the CORTES developers requested displays showing the order in which search states were explored, which states led to successful reservations, and information about each state, including the date of the new reservation, the reserved resource, and that resource's criticality. The resulting display is shown in Figure 7-1.

Again SAGE uses a single object set (circles) to express correlations among the different attributes: vertical position shows search state, horizontal position shows reservation date, color shows the resource reserved at that state, and size shows the resource's criticality. the vertical positions of the circles capture this. The hierarchical exploration of search states is expressed via the indentations among the axis labels, with the successful states highlighted.

This show many relationships underlying the behavior of the scheduler. First, considerable backtracking occurred early, but did not recur after an acceptable reservation was found for the highly critical Resource2. Second, clusters of reservation dates were tried prior to backtracking, and one can see the order in which CORTES considered alternative dates when attempting to resolve a constraint violation. Third, the criticality of the resources generally decreased as the schedule neared completion. Fourth, the search opportunistically switched among resources in making reservations as the search progressed. Our experiences with CORTES demonstrated SAGE's utility for debugging and monitoring the performance of a system during its development.

7. Concluding Remarks

This paper presents a system called SAGE, which automatically designs presentations of information. SAGE research areas include developing a complete taxonomy of data and goal characteristics, general representations of

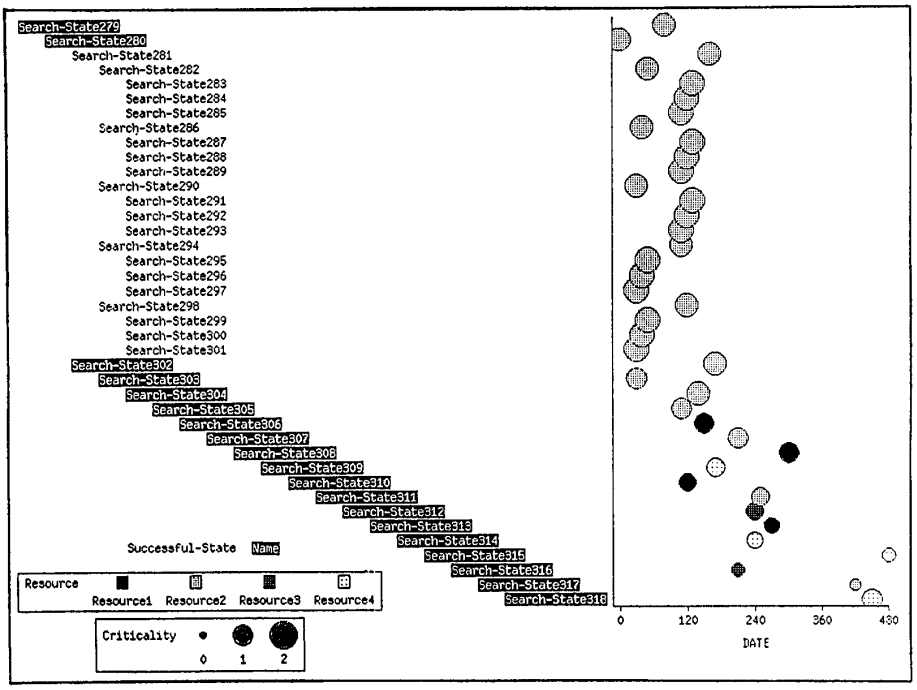

Figure 7-1: Visualizing Search Behavior of Constraint Satisfaction Systems

graphical techniques, and new mechanisms for integrating and refining techniques. SAGE's value as an application is its ability to:

- design creative displays when complex combinations of data are needed,
- allow end users to iteratively change their presentation goals,
- allow system designers to experiment with combinations of information and goals to construct a display which best meets the needs of their application,
- allow system developers to rapidly design effective displays that visualize the intermediate results of their systems, in order to debug and evaluate system behavior.

Our current work involves integrating SAGE with an automatic programming system. The goal is for SAGE to provide presentation support for all application programs generated by the system. An important ingredient of this work is a methodology for modeling domain-specific analysis tasks in a manner which enables them to be mapped automatically onto SAGE's current set of information-seeking goals.

References

[1] E.R. Tufte, *The Visual Display of Quantitative Information*, Graphics Press, 1983.

[2] McDermott, J, "Developing Software is like Talking to Eskimos about Snow", *Proceedings of AAAI-90*, Boston, Mass, July 1990.

[3] Mackinlay, J.D., "Automating the Design of Graphical Presentations of Relational Information", *ACM Transactions on Graphics*, Vol. 5, No. 2, April 1986, pp. 110-141.

[4] Roth, S.F., Mattis, J.A., and Mesnard, X.A., "Graphics and Natural Language as Components of Automatic Explanation", in *Architectures for Intelligent Interfaces: Elements and Prototypes*, Sullivan, J. and Tyler, S., ed., Addison-Wesley, Reading, Mass, 1991.

[5] Roth, S.F. and Mattis, J, "Data Characterization for Intelligent Graphics Presentation", *Proceedings of the CHI '90 Conference*, ACM, Seattle, Washington, April 1990.

[6] Gargan, R.A., Sullivan, J.W., and Tyler, S.W., "Multimodal Response Planning: An Adaptive Rule Based Approach", *CHI '88 - Human Factors in Computing Systems*, ACM/SIGCHI, Washington, D.C., May 1988, pp. 229-234.

[7] Casner, S.M., *A Task-Analytic Approach to the Automated Design of Information Graphics*, PhD dissertation, University of Pittsburgh, 1990.

[8] Mark S. Fox, Norman Sadeh, and Can Baykan, "Constrained Heuristic Search", *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989, pp. 309-315.