Preliminary System Design for an EDA Assistant

Robert St. Amant and Paul R. Cohen
Computer Science Dept., LGRC
University of Massachusetts
Box 34610
Amherst, MA 01003-4610
stamant@cs.umass.edu, cohen@cs.umass.edu

1 Understanding Complex Data

Data analysis plays a central role in our attempts to understand the behavior of complex systems. Derived models or descriptions of complex phenomena are the result of a gradual process of exploring possible interpretations, generating good and bad hypotheses, and gradually refining results. Exploratory studies are the informal prelude to experiments, in which questions and procedures are refined. Exploration is a kind of detective work: before making a formal presentation of a case, the researcher must first ensure that clues are followed to their conclusions and false trails are eliminated. The process can determine the nature of relationships between variables, their functional behavior and interactions. Exploratory results give rise to confirmatory studies in a cycle of successively more refined exploration and confirmation [7, 13].

Exploratory data analysis (EDA) provides a wide range of statistical tools for the early stages of analysis [26]. Simple exploratory results include histograms that describe discrete and continuous variables, schematic plots that give general characterizations of relationships, partitions of relationships that distinguish different modes of behavior, functional simplification of low-dimensionality relationships, and two-way tables such as contingency tables. Combination of such partial descriptions of data allows a more complete picture to emerge.

Human analysts can find it challenging to select and apply exploratory tools effectively. EDA poses a difficult search problem, a problem of *control*. A wide variety of operations apply: arithmetic composition of variables, such as those used in function finding; model-based variable decomposition, as performed by linear regression; partitioning and clustering operations, such as those used in numerical and conceptual clustering systems; feature extraction operations like statistical summaries; transformation operations; and generalization operations. Goal states can only rarely be specified from the start. The process is opportunistic, in the sense that the selection of an appropriate EDA operation can depend on the results of the immediately preceding operation plus the context provided by all previous operations. The search space grows explosively.

While research in statistics and artificial intelligence has addressed issues in the automation of later stages of analysis, such as theory generation, model selection, and experiment design [23], less attention has been given to initial exploration of data. We have developed a novel approach to exploration as search. This paper gives an overview of the design of AIDE, the Assistant for Intelligent Data Exploration, which assists humans in the early stages of data analysis [1].

This research is supported by ARPA/Rome Laboratory under contracts F30602-91-C-0076 and F30602-93-C-0010.

The system adopts a script-based planning approach to automating EDA. Data-directed mechanisms extract simple observations and suggestive indications from the data. Scripted EDA operations are then applied in goal-directed fashion to generate deeper descriptions of the data. Control rules guide the EDA operations, relying on intermediate results for their decisions. The system is mixed-initiative, capable of autonomously pursuing high and low level goals while still allowing the user to guide or override its decisions.

AIDE is currently a prototype under development. We emphasize that the work presented here is incomplete.

2 The Problem

In *Exploratory Data Analysis*, John Tukey describes EDA in this way:

> A basic problem about any body of data is to make it more easily and effectively handleable by minds—our minds, her mind, his mind. To this general end:
>
> - anything that makes a simpler description possible makes the description more easily handleable.
> - anything that looks below the previously described surface makes the description more effective.
>
> So we shall always be glad (a) to simplify description and (b) to describe one layer deeper [26].

Our design revolves around an understanding of the EDA process as the iterative application of four classes of operations: operations for descriptive handling, simplification, extension, and deepening.

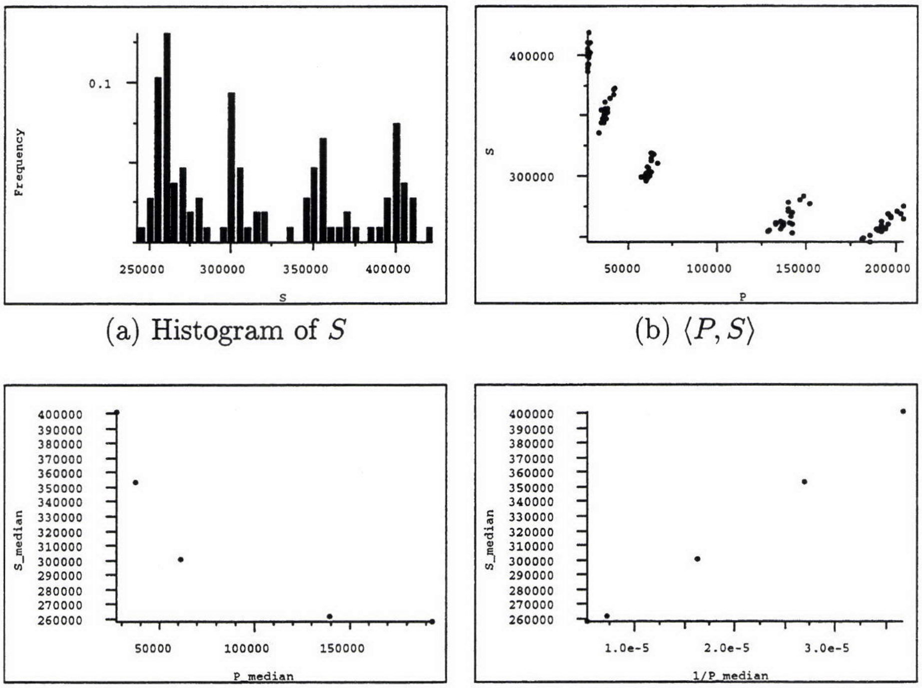
Handling operations produce descriptions of data. These may be single-valued statistics such as means and medians, or structures such as histograms and fitted lines. Handling operations produce results that can be directly used or interpreted by the user.

Simplification operations transform data to facilitate description. A log transform that straightens a skewed relationship is one example. Irregularities such as outliers are more easily detected in linear relationships than in nonlinear ones; a change in density can often enhance patterns in data; many statistical operations, even ones as simple as Pearson's correlation coefficient, rely on linearity. A straightening operation simplifies in that it enhances our observation, manipulation, and evaluation—in a word, our handling—of the data.

Deepening operations increase the detail, accuracy, and precision of descriptions, in the way a microscope enhances observation by trading a global perspective for local detail. An example is the common practice of examining the residuals of a linear fit. The examination itself is carried out by simplification and handling operations. Residual examination can bring to light structure not captured by the line, such as clustering, unequal variance in residuals, or local deviations from linearity.

With descriptive, simplifying, and deepening capabilities, EDA builds descriptions of single variables, bivariate relationships, tables, partitions and clusters. Local descriptions can have non-local, often widespread implications, however; these are examined by *extension* operations. When clusters are observed in the values of a variable x, for example, and similar clusters are observed in y, then an extension operation prompts the examination of the relationship $\langle x, y \rangle$.

EDA can be viewed as controlling the sequential execution of these operations. Handling operations establish descriptions of data wherever they are applicable. Other operators support the



(a) Histogram of S

(b) $\langle P, S \rangle$

(c) $\langle P_{median}, S_{median} \rangle$

(d) $\langle 1/P_{median}, S_{median} \rangle$

Figure 1: Exploring ship and port costs

descriptive process: simplification operations facilitate descriptions by transforming data into more appropriate forms; deepening operations refine descriptions to increasing levels of detail; extension operations incrementally widen the scope of descriptions.

3 A Brief Example

Much of our research deals with the behavior of AI planners in demanding simulation environments. One such system is TransSim, a transportation planner/simulator [20]. In TransSim, ships travel between ports carrying cargo along assigned routes. Bottlenecks and other occurrences change the environment in unexpected ways. The planner must react dynamically to these changes by rescheduling dock and ship assignments and rerouting cargo through different intermediate ports.

An early experiment examined relationships between resource costs. We collected measurements of cumulative port cost (P), ship cost (S), and dock cost (D), the number of ships (N), and trial duration (TD). We were particularly interested in the relationship between the resource costs P and S over the duration of a trial. While we set N at different values, we fixed per-day port, ship, and dock costs.

We begin with summary statistics for S, cumulative ship cost: the median is about 310K, the interquartile range 95K, and there is a slight skew toward lower values. More significantly, when we examine a histogram of S (Figure 1a), there are four distinct clusters in the data. Our preliminary partial description of S comprises the statistics and our observations about the clustering.

When we turn to the relationship $\langle P, S \rangle$ (Figure 1b), we see a different pattern: the values fall into *five* clusters. The distinct separation in $\langle P, S \rangle$ values, as well as the observation that one of the modes in the histogram of S (Figure 1a) corresponds to a cluster twice as large as the others, leads us to return to our description of S. We establish an alternative description of S as containing five clusters, consistent with the clusters in $\langle P, S\rangle$.

Continuing with our analysis of $\langle P, S\rangle$, we see that values in the leftmost cluster (which we denote ps_a) can be fit by a straight line. In fact, this is true for all five clusters, though it is a different line in each case. Once we settle on the description of each cluster as a line, we can add the observation that the slopes of the lines decrease as the clusters move toward the right.

If we plot the central locations of the clusters ps_a through ps_e (i.e., the median coordinates P_{med} and S_{med} for each cluster), as shown in Figure 1c, we see that these five summary points can be fit by a smooth curve. Further exploration shows that the curve is of the form $S_{med} = c/P_{med}$. When we perform this transformation (Figure 1d) it straightens the curve, leaving no clear pattern in the residuals.

Extending the analysis, we find that the five discrete values of the variable N, the number of ships, correspond exactly to the clusters found in the relationship $\langle P, S\rangle$ and the variable P alone. If we treat membership in a specific cluster as a new categorical variable, C_{mem}, the two-way table of frequencies, below, shows the relationship. Because we have experimental control over the number of ships but not the cumulative costs, we count an observation about the former measurements as an explanation of the latter. In this case, N explains the clustering of resource costs.

	N = 15	N = 20	N = 40	N = 60	N = 80
C_{mem} = a	22	0	0	0	0
C_{mem} = b	0	22	0	0	0
C_{mem} = c	0	0	21	0	0
C_{mem} = d	0	0	1	20	0
C_{mem} = e	0	0	0	0	21

We can continue by considering other variables. It turns out, for example, that a simple functional relationship with trial duration accounts for the difference in slopes between the clusters. A more complete analysis is given in a report under preparation. This description, though brief, should give the flavor of the analysis.

To summarize, we begin with *initial descriptions* of the data, such as the observation of gaps between adjacent values. From these we generate *indications* [19], or suggestive characteristics: the data fall into clusters. Based on these indications we apply specific EDA *procedures*: we break the data down and analyze the clusters individually. These procedures may involve *iterative refinement*, as with the alternative descriptions of clusters in S. When possible, we *simplify*; instead of dealing with all data points, we work with a reduced dataset of just the median points of the clusters. We then follow two separate courses: we *extend* current results, here by considering other variables, and we *deepen* results, in this case by turning from a surface description of the clusters to an examination of the behavior of the data within each cluster. As we proceed we watch for potential *generalizations*, such as the set of related straight lines that fit the clusters. The result is a coherent, structured description of the data.

4 Basic Operations

Datasets are the basic level of data representation. A dataset is an extension of the familiar relational table of attributes and values. Extensions capture implicit and explicit knowledge we have about the data. Datasets and attributes can be specialized as distinct types, the most common cases being relationships (or multivariate subsets of datasets) and variables (or univariate relationships).

While there exist operations applicable only to variables, or to relationships, there is a large set of shared operations that make it desirable for all data to be handled in tabular form.

A dataset may contain sets of other, as values of an attribute. Datasets and attributes are annotated with information about their characteristics as well. A variable may be annotated with the information that it is a dependent variable in an experiment; a relationship $\langle x, y, z\rangle$ may contain the annotation that $z = x + y$. Dataset attributes are furthermore conceptually multidimensional, for reasons that will become clear below. The annotations and associations between related structures let us build a complex network of results during analysis.

We manipulate a dataset in three distinct ways. The first kind of manipulation involves subdividing or combining elements of datasets. In order to construct fitted lines to each cluster in the TransSim data, we partitioned the original data into five smaller datasets. We then examined each cluster independently, at a finer level of detail. We call an operation that breaks data into smaller but similar parts a *data decomposition.* A data decomposition generates a mapping from the elements of a dataset to membership in a set of new, derived datasets. Operations that bin data, exhaustively partition data, or nonexhaustively cluster data all decompose data in this way. The inverse operation, *data composition*, combines separate datasets. Data decompositions and compositions let us capture independent detail in subsets of data and recombine results.

The second kind of manipulation involves deriving new attributes for elements of the dataset. During the Transsim analysis we performed a transformation of cluster medians to remove curvature. Operationally the procedure transformed a single attribute, P_{med}, into a new attribute, P', where $P' = 1/P$. We call such an operation an *attribute transformation.* A special type of transformation is attribute composition: consider the transformation from attributes port cost, ship cost, and dock cost to $S + P + D$, or total cost. Another type of transformation is attribute decomposition: to conclude that a given line is a good fit to a relationship, we need to generate the residuals by subtracting $\hat{P}$ from P—by decomposing the attribute into structure and residual.

Because dataset attributes may contain datasets as well, another useful transformation is the mapping of a statistic over the elements of an attribute. Our data decomposition of the dataset T generates five new datasets. More precisely, the result is a new dataset, T_c, which contains a single attribute "Clusters", whose value is these new datasets. We can now apply an attribute transformation to the "Clusters" attribute, to produce new attributes such as "x-median" and "y-median". We continue by fitting a line, a further attribute transformation. Functions for attribute transformation include arithmetic operations, exponentiation, and higher level dataset operations such as these.

The third kind of manipulation is *reduction*, which is the type of function performed by summary statistics such as means and medians. Correlations and partial correlations are similarly reductions of bivariate and multivariate relationships.

Designing a system around such general structures and operations is not simply an attempt at conciseness. Rather, the design is geared specifically toward the kinds of operations appropriate for EDA. Consider the example of generating a histogram for a discrete variable N in a dataset T. To build a histogram we divide a variable into bins and count the number of observations that fall into each bin. Using the operations describe above, we can implement a histogramming procedure as follows. First we apply a data decomposition to T. The result is a set of new datasets that form disjoint subsets of T, stored as an attribute "Data" in a new dataset, T_h. Our operations now work on T_h. An attribute transformation of "Data", based on the count statistic, gives the size of the bins. Another transformation using the mode statistic gives the value of N associated with each bin. These distinct values and their counts can then be displayed directly in histogram form.

This may seem an inordinate effort to produce such a basic structure, but consider a simple extension: the contingency table. To build a contingency table between N and C_{mem}, an attribute that records the membership of each data point in a specific cluster, we follow essentially the same procedure. This time the decomposition simultaneously bins N and C_{mem}. Decomposition operations are not limited to single variables; they can just as easily decompose two variables or an entire dataset. The result of the operation is a dataset, T_c, with a two-dimensional attribute containing the partitions, one for each unique value of N and C_{mem}. Again we transform by the count statistic. Calculating the N, C_{mem} values for each bin is equally straightforward. The desired contingency table data is the result. A simple variation of this procedure produces a box plot, by computing letter values rather than counts. More complex two-way tables can also be generated in a similar way, by calculating statistics for a third variable in each of the partitions.

Producing structures in this way has two abstract benefits: complex procedures can often be seen as natural extensions of existing procedures, and natural connections between conceptually similar structures become clear. While it is trivial to observe that a contingency table is a two-dimensional histogram, it is worth stressing that the two structures can be produced by very similar procedures, and that both procedures are different instances of a single canonical, divide-and-conquer EDA procedure. A data decomposition breaks a dataset into smaller parts; dataset-level attribute transformations compute a set of features of the reduced data; these features are explored both individually and in aggregate. The procedure is a simple, powerful example of generalization, as described in the conceptual clustering and constructive induction literature [18, 10].

We have developed a scripting language to implement these procedures. Scripting has an advantage over earlier rule-based approaches [11, 24] in that well-understood procedures and decisions can be explicitly coded. The procedure for histogramming is implemented by the sequence script below (edited slightly for length).

```
(define-sequence-script variable-histogram-script (the-variable)
  :doc "First generate a dataset of partitions, then calculate the
       size of each partition."
  :output-type histogram
  :bindings ((partition-dataset (partitions count min max)))
  :script (:sequence
          (script-partition the-variable
                            #'(lambda (x) x)
                            :output partition-dataset)
          (script-transform partition-dataset
                            #'(lambda (partition)
                                (script-reduce partition 'count))
                            :name count
                            :key (attributes partitions))
          . . .))
```

We have already mentioned variants of the histogramming procedure, which are implemented by similar scripts. Other scripts compute initial values for parameters of resistant lines, transformations of skewed relationships, and so forth. Each script produces a set of new or transformed structures, appropriately annotated and related to existing structures. Scripts may call one another for intermediate results. As scripts execute, a growing semantic net of datasets, relationships, and their attributes and annotations is generated.

5 Control

Scripts provide an alternative to search with sequential control of operations. Nevertheless, they are only part of a solution; we still face the problem of deciding which scripts to apply. Guidance is provided by *control scripts* which allow combinations of scripts beyond sequencing, and by two complementary mechanisms: *indications* and *intentions.*

The extended scripting language is based on work in knowledge-based signal processing [5]. A control script extends sequential scripts in several ways. Control scripts contain constructs that allow sequencing, iteration, mapping, and conditionalizing of both actions and subgoals. A script tests its input variables with a set of constraints, using a simple extension of pattern matching. A script to explore a bivariate relationship is shown below. The script becomes applicable when a higher level script establishes an "initiate exploration" goal for a bivariate relationship. In the case of relationships, it is useful to explore the variables it contains before exploring the relationship itself. We cannot determine, for example, that a contingency table is a reasonable description of a relationship before we check whether its variables are discrete. This script first generates features for the relationship, then explores the variables of the substructure, then generates indications for the relationship, and finally explores the relationship. We have developed much more complex scripts which, for example, implement iterative improvement algorithms for resistant lines and heuristic straightening procedures for nonlinear bivariate relationships.

```
(define-script initiate-relationship-exploration-plan (structure)
   :satisfies (initiate-exploration (:typep bivariate-relationship))
   :script
   (:sequence
      (generate-features structure)
      (:map (substructure (internals structure))
         (subgoal explore-sg1 (generic-explore substructure)))
      (subgoal indications (generate-indications structure))
      (subgoal explore-sg2 (explore structure))))
```

Script execution is driven by indications and intentions. Indications are suggestive characteristics of the data, most often involving evaluation of a statistic or descriptive structure. Indications establish goals for exploration. For example, evidence of clustering is an indication, as is the presence of outliers, and curvature in the residuals of a linear fit. In each of these cases the indication leads to goals which, if satisfied, explain its presence. Indications are generated based on features of structures, and may traverse the relationships between structures for their calculations. The indication below checks whether a relationship contains two discrete variables.

```
(define-indication discrete-valued-relationship-p
    ""
   :class bivariate-relationship
   :features (identity)
   :form #'(lambda (relationship)
             (structure-match '(:indirect contents
                                 ((:has-indication discrete-valued-p)
                                  (:has-indication discrete-valued-p)))
                               relationship)))
```

Indications provide guidance based on *internal* characteristics of data. Intentions, in contrast, impose *external* considerations on exploration; for example, "Determine whether there is a difference in behavior of the system for cases in which $R < 1$ and cases in which $R \geq 1$," or "See if F is a plausible direct cause of S." Intentions may be provided explicitly as user directives, or implicitly, as assertions made by the user to provide the system with more information. An initial causal model, generated by the user, is an example of a set of implicit intentions to test the plausibility of specific causal relationships. Indications and intentions provide data-driven and goal-driven guidance to exploration. In a nutshell, indications help generate descriptions suggested by the data, while intentions help generate descriptions appropriate for the goals of the analysis.

Indications and intentions determine whether a specific script is relevant to meeting a specific goal. Complex interactions can take place between indications or intentions. For example, AIDE currently contains two simple indications of univariate clusters. Clustering is indicated (1) if there is a significant gap between adjacent values, as determined by a nearest neighbor clustering algorithm and a heuristic threshold, or (2) if there are "regions of constancy" within the variable, or discrete values common to several observations. The first indication tends to find large clusters, the second small ones. Scripts activated by these indications will decompose the variable according to the two different criteria produced. Sometimes, however, there is overlap between the criteria (e.g., the results of timed trials in which successes are clustered around low values, but with a significant number of failures at a high cut-off threshold.) In these cases the clustering criteria produced by the indications, which are used as input in the scripts, can be examined and perhaps modified to account for the interaction.

Focusing heuristics act in this role [5]. Focusing heuristics take advantage of the sequence of operations leading up to a specific decision (the plan context) and the features, indications, and intentions associated with the data under consideration (the data context) to decide which of the applicable plans and structures should be pursued, which delayed, and which abandoned. Focusing heuristics implement local, context-dependent control. The heuristics have the purpose of

- selecting appropriate plans, and suppressing inappropriate plans, for exploration of a structure;
- capitalizing on similarities between structures so that earlier results may be reused (e.g., successive line fits to clusters);
- reconsidering earlier decisions based on new information (e.g., reparameterizing a clustering script to increase consistency);
- implementing user directives to override default heuristics.

Focusing is associated with plan selection for goal satisfaction and with selecting parameters for selected plans. Focusing heuristics are activated the presence of specific indications or intentions, while actions are implemented as Lisp code. Focusing heuristics thus take the form of rules. In the univariate clustering example above, involving low, clustered success scores and high, constant failure scores, a "handle" goal is established during exploration of the variable. Two distinct instances of the script "isolate-clusters" are activated, corresponding to the different indications. Each indication supplies a partitioning criterion for further exploration. A focusing heuristic is associated with the "handle" goal. The heuristic may suppress either script or change the clustering criterion to a different value. In this case the heuristic would first pursue the partitioning between successes and failures, preferring a small number of large clusters to a larger number of small clusters. On completion of this search path (i.e., when the "handle" goal becomes satisfied) the focusing heuristic is reactivated to decide whether the other partitioning should be considered as well.

EDA processing shows strong similarities to the kind of processing common in blackboard-based interpretation systems that perform signal interpretation [15]. Such systems merge diverse sources of information by means of evidence aggregation and differential diagnosis. They can furthermore often take advantage of hierarchical decompositions of domain concepts. A speech understanding system such as Hearsay, for example, combines elements at the word level to produce structures at the phrase level. Hearsay relies on a natural mapping of words to one level of abstraction, phrases to a higher level. During data analysis, for example as we process a set of clusters, the attributes we generate from the clusters allow us to construct higher level generalizations, such as similar linear fits. Whenever we deal with reductions or transformations of data we can speak of possible generalizations as well as deeper levels of detail. A blackboard design captures these concepts naturally.

6 Related Work

This work draws on a number of different sources. The clearest relationship is to early work in developing concepts of statistical strategy, or the formal descriptions of actions and decisions involved in applying statistical tools to a problem [13]. Gale and Pregibon's REX system, for example, implemented a strategy for linear regression [11]. Oldford and Peters implemented a complex strategy for collinearity analysis [21]. The goals of AIDE bear a resemblance to those of Lubinsky and Pregibon's TESS [16], which supports analysis by accommodating user knowledge of context in a search good descriptions of data. AIDE extends this work by supplying an appropriate taxonomy for EDA operations and using this taxonomy to guide search at different levels of abstraction.

In machine learning there have been several broad approaches to data analysis that bear some similarity to the approach described here, especially at the level of data manipulation. Systems such as BACON [14], Fahrenheit [28], and E* [22] use attribute transformation to discover functional relationships between variables. In the area of conceptual clustering, systems such as COBWEB [10], Autoclass [6], and ITERATE [4, 3] use data decomposition and composition operations to group observations appropriately. The significance of this work, of course, is not at the level of data manipulation, but rather in the heuristics or biases they apply to the search for patterns. Some of these biases have been in corporated into AIDE indications and scripts.

Work in automated scientific discovery, taken beyond function finding, is also relevant. The PENCHANT system [27], applied to the domain of developmental biology, discovers patterned behavior in data. Terms used to describe the system are similar to the concepts we discuss, but at a different granularity (e.g., indications and features/attributes are wrapped up in the single concept of "quantities".) Building on the 49er system, Zytkow and others [25, 29] have developed a taxonomy of "regularities", which include equations and contingency tables. These regularities are captured in a general form in the representation presented here. Other issues in the related area of knowledge discovery in databases [17] are relevant as well.

Causal modeling systems, especially systems such as TETRAD [12, 9], offer yet another perspective on data exploration. Much of the external knowledge brought to bear in exploring experimental data is in the form of causal relationships: y is a dependent variable, and cannot cause variation in x; the correlation between z and w is due to sampling bias. When a system relies on statistical tests for its reasoning, exploration can comes into play to ensure that assumptions of the tests are not violated (e.g., linearity for partial correlations [2].)

Central to AIDE is the opportunistic, incremental approach to discovery described by Tukey, Mosteller, and other advocates of EDA. There are some obvious difficulties with the approach: maintaining consistency in an incrementally growing set of descriptions can be difficult; in many cases local techniques can miss simple global patterns [8]; local techniques can lead to a plethora of spurious results [22]; the problem of scaling in cluster analysis is notoriously difficult. Nevertheless many such problems can be alleviated in a system does not always act autonomously, focuses on both data and on goals of the analysis, and pursues exploration paths with the aid of external knowledge and the context supplied by its own actions. We have designed AIDE to address these concerns.

Acknowledgments

This research is supported by ARPA/Rome Laboratory under contracts F30602-91-C-0076 and F30602-93-C-0010. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes not withstanding any copright notation hereon.

References

[1] Robert St. Amant and Paul R. Cohen. A planning representation for automated exploratory data analysis. In D. H. Fisher and Wray Buntine, editors, *Knowledge-Based Artificial Intelligence Systems in Aerospace and Industry Proc. SPIE 2244*, 1994.

[2] Robert St. Amant and Paul R. Cohen. Toward the integration of exploration and modeling in a planning framework. In *Proceedings of the AAAI-94 Workshop in Knowledge Discovery in Databases*, 1994.

[3] Gautam Biswas, Jerry Weinberg, and Cen Li. Iterate: A conceptual clustering method for knowledge discovery in databases. In B. Braunschweig and R. Day, editors, *Innovative Applications of Artificial Intelligence in the Oil and Gas Industry.* Editions Technip, 1994. to appear.

[4] Gautam Biswas, Jerry Weinberg, Quin Yang, and Glen R. Koller. Conceptual clustering and exploratory data analysis. In *Proceedings of the Eighth International Conference on Machine Learning*, 1991.

[5] Norman Carver and Victor Lesser. A planner for the control of problem solving systems. *IEEE Transactions on Systems, Man, and Cybernetics, special issue on Planning, Scheduling, and Control*, 23(6), November 1993.

[6] P. Cheeseman, D. Freeman, J. Kelly, M. Self, J. Stutz, and W. Taylor. Autoclass: a bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning.* Morgan Kaufmann, 1988.

[7] Paul R. Cohen. *Empirical Methods in Artificial Intelligence.* MIT Press, 1994. In press.

[8] Cuthbert Daniel and Fred S. Wood. *Fitting Equations to Data.* Wiley, 1980.

[9] Marek J. Druzdzel and Clark Glymour. Application of the TETRAD II program to the study of student retention in u.s. colleges. In *Proceedings of the AAAI-94 Workshop in Knowledge Discovery in Databases*, 1994.

[10] Douglas Fisher and Pat Langley. Conceptual clustering and its relation to numerical taxonomy. In William Gale, editor, *Artificial Intelligence and Statistics I.* Addison-Wesley, 1986.

[11] W. A. Gale. Rex review. In W. A. Gale, editor, *Artificial Intelligence and Statistics I.* Addison-Wesley, 1986.

[12] Clark Glymour, Richard Scheines, Peter Spirtes, and Kevin Kelly. *Discovering Causal Structure: Artificial Intelligence, Philosophy of Science, and Statistical Modeling.* Academic Press, 1987.

[13] D.J. Hand. Patterns in statistical strategy. In W.A. Gale, editor, *Artificial Intelligence and Statistics I*, pages 355–387. Addison-Wesley, 1986.

[14] Pat Langley, Herbert A. Simon, Gary L. Bradshaw, and Jan M. Zytkow. *Scientific Discovery: Computational Explorations of the Creative Processes.* MIT Press, 1987.

[15] Victor R. Lesser, S. Hamid Nawab, and Frank I. Klassner. IPUS: An architecture for the integrated processing and understanding of signals. *Artificial Intelligence*, 1994. To appear.

[16] David Lubinsky and Daryl Pregibon. Data analysis as search. *Journal of Econometrics*, 38:247–268, 1988.

[17] Christopher J. Matheus, Philip K. Chan, and Gregory Piatetsky-Shapiro. Systems for knowledge discovery in databases. *IEEE TKDE special issue on Learning and Discovery in Knowledge-Based Databases*, 1993.

[18] R. S. Michalski and R. E. Stepp. Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An artificial intelligence approach.* Morgan Kaufmann, 1983.

[19] Frederick Mosteller and John W. Tukey. *Data Analysis and Regression.* Addison-Wesley, 1977.

[20] Tim Oates and Paul R. Cohen. Toward a plan steering agent: Experiments with schedule maintenance. In Kristian Hammond, editor, *Second International Conference on Artificial Intelligence Planning Systems*, pages 134–139. AAAI Press, 1994.

[21] R. Wayne Oldford and Stephen C. Peters. Implementation and study of statistical strategy. In W.A. Gale, editor, *Artificial Intelligence and Statistics I*, pages 335–349. Addison-Wesley, 1986.

[22] Cullen Schaffer. Domain-independent scientific function finding. Department of Computer Science Technical Report LCSR-TR-149, Rutgers University, New Brunswick, NJ, 1990. PhD Thesis.

[23] Jeff Shrager and Pat Langley. *Computational Models of Scientific Discovery and Theory Formation.* Morgan Kaufman, 1990.

[24] Ronald A. Thisted. Representing statistical knowledge. In W.A. Gale, editor, *Artificial Intelligence and Statistics I*, pages 389–399. Addison-Wesley, 1986.

[25] Molly Troxel, Kim Swarm, Robert Zembowicz, and Jan Zytkow. From law-like knowledge to concept hierarchies in data. In *Proceedings of the AAAI-94 Workshop in Knowledge Discovery in Databases*, 1994.

[26] John W. Tukey. *Exploratory Data Analysis.* Addison-Wesley, 1977.

[27] Raul E. Valdes-Perez and Aurora Perez. A powerful heuristic for the discovery of complex patterned behavior. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 326–334, 1994.

[28] Jan M. Zytkow. Combining many searches in the fahrenheit discovery system. In *Proceedings of the Fourth International Conference on Machine Learning*, pages 281–287, 1987.

[29] Jan M. Zytkow and Robert Zembowicz. Database exploration in search of regularities. *Journal of Intelligent Information Systems*, pages 39–81, 1993.