Intelligent Assistance for Computational Scientists: Integrated modelling, experimentation, and analysis

Dawn E. Gregory Paul R. Cohen

Experimental Knowledge Systems Laboratory
Computer Science Department, LGRC
University of Massachusetts
Box 34610, Amherst, MA 01003-4610
{gregory,cohen}@cs.umass.edu

1. Introduction

Computing technology has changed the way scientists work. Among the contributions of this new paradigm are the *computational sciences*, which involve the study of computer simulations rather than physical systems. This transition to a simulated world carries with it an important scientific advantage: the opportunity to run experiments that are expensive, dangerous, or impossible in the real world. Unfortunately, such experiments are often too easy, and the scientist is overwhelmed with empirical data.

The fields of Artificial Intelligence (AI) and Statistics are concerned with modelling and analyzing such large bodies of data. AI employs heuristic reasoning and knowledge to select potential models, and statistical analysis verifies a proposed model. The combination of knowledge-based, heuristic, and statistical techniques is quite successful at modelling experiment data (e.g. [11]).

Our goal is to provide an intelligent, integrated environment for scientific modelling, experimentation, and analysis, called the *Scientist's Empirical Assistant* (SEA). SEA is an assistant to human scientists: it automates model generation and verification, experiment design and data collection, but also relies on a human user for guidance, domain knowledge, and decision-making. SEA designs and runs prospective experiments with a simulator, allowing it to draw stronger conclusions than with post-hoc data analysis alone.

SEA employs a variety of techniques from both AI and Statistics. It uses heuristic- and knowledge-based reasoning to propose models, design experiments, and select analyses. It applies statistical techniques to verify models against experiment data. It develops plans to direct its course of action, and learns which plans are most successful based on past experience. Finally, it models the knowledge of the user to ensure its suggestions and decisions are appropriate.

1.1 A Real-World Example

We begin by describing a real-world study of simulation behavior that illustrates the important features of an automated scientific assistant. In this study, a variety of techniques and strategies are employed in the comparison of two policies for balancing a computational load among the processors in a network. The details of this study are documented elsewhere [7].

Simulating network performance on randomly generated loads allows us to resolve two important questions empirically. First, it is expected that policy 1 does a better job of balancing than does policy 2, resulting in a lower running time. This condition is stated as a *comparison of*

treatments, suggesting the hypothesis that the average running time is significantly different between the policies.

SEA gathers empirical observations of both policies running on simulated networks of several different sizes, and uses a *t-test* to check the hypothesis. Due to large background variance, the t-test shows no significant difference between the policies. However, the net running time of the network can be factored into two parts: the minimum possible running time if the load were perfectly balanced, and the overhead due to insufficient balancing. Reapplying the t-test to the overhead component exposes a significant difference between the two policies, allowing SEA to conclude that policy 1 is indeed a better policy.

The second part of the study examines the conditions under which the network behaves optimally. Optimal performance is achieved when the load is distributed evenly among the processors; the user wants to determine whether this limit is approached *asymptotically* as the load grows. Again, SEA considers the overhead component of the running time, ϵ. If the policies behave asymptotically optimally, ϵ will grow more slowly than the load N. During the analysis, SEA observes that both the mean and variance of ϵ are affected by the size of the network. These results trigger a rule (subsequently confirmed by the user) that asymptotic behavior be examined separately for each size. Further analysis supports this course of action, as SEA discovers that smaller networks appear to approach optimality while larger ones do not.

2. Issues in Automation

In this section we explore some of the difficult issues regarding the design of an automated scientific assistant. In addition to the problems associated with the process of scientific discovery itself, we must also be concerned with questions of human/computer interaction. The questions we need to answer include: Which parts of the process can be automated? What types of knowledge are needed? Where does the knowledge come from? How is the process controlled? What are efficient control strategies? What is the role of the user? Who makes the decisions? Who interprets the results? The answers to these questions specify the important features of SEA.

2.1 Integrating Multiple Perspectives

Let us begin by considering that SEA and a human scientist will have different perspectives on an empirical study. The user's view is knowledge-rich and domain-centered, based on one or more *research questions* that have arisen from a scientific agenda. Research questions are those framed in the jargon of a scientific domain, such as: "Does policy 1 achieve optimal performance?" Generally, such questions cannot be answered directly — they must first be translated, formalized, and quantified.

SEA's view of science is more operational and process-centered, concentrating on its knowledge of scientific procedures and empirical tactics. These procedures focus on *empirical questions*, the questions that can be answered by running experiments and examining the resulting data. SEA's approach to science is to generate a variety of empirical questions and select the most promising one, design and run an experiment that gathers appropriate data, and then derive an answer to the question through analysis of the data. Given a well-formed empirical question, each of these operations can be achieved with simple procedures.

The difference in perspectives between SEA and its user implies a critical research problem: how can the skills of each be coordinated to successfully achieve scientific goals? This issue falls into the realm of *mixed-initiative* systems. According to Allen [1], the important issue in such a system is to identify the conditions under which control of initiative should switch from one agent to another. In our design, SEA maintains control most of the time, but relinquishes authority to the user under two conditions. First, SEA knows nothing about the domain beyond what is captured in the simulation program, so it relies entirely on the user for high-level domain knowledge. Second, SEA often generates several potential courses of action, and the user must decide which alternative is best. Thus, the user is treated as both a domain and strategic expert, while SEA plays the roles of bookkeeper, experiment-runner, data analyst, and number-cruncher.

2.2 Modelling and Data Analysis

We now consider SEA's perspective of the scientific method in more detail. SEA's task is to construct a model that predicts the behavior of some target system, as represented by a domain simulator. A model is a collection of *statements* about a set of *experiment variables*. Variables are measurements of interesting features of the domain, as specified by the user. Statements about the variables may describe functional equivalence, asymptotic limits, causality and temporal dependency, effects of a treatment variable, and so on. Each statement must be verified both statistically and logically before it is added to the model.

The analysis techniques that verify a statement depend on its type. Many of the current approaches to scientific modelling support only one type of statement and/or analysis, so the association between statement and analysis is implicit in the design. Even in systems that integrate several statement types (e.g. [9]), the choice of analysis is hard-wired into the control mechanism.

SEA accommodates a variety of statement types through explicit knowledge of the mapping from conjectured statement to analysis technique. The transformation involves an intermediate representation, the *hypothesis*, to bridge the conceptual gap between them. At first it seems odd to say that a conjecture isn't the same thing as a hypothesis, so let's look at an example. Suppose SEA needs to test a statement of functional equivalence, such as $Y = 3X + 5$. Standard practice is to analyze the *residuals* of this statement, $\epsilon = Y - (3X + 5)$;[1] if the statement is correct, the residuals have a mean of 0 ($\bar{\epsilon} = 0$), low variance ($\sigma_\epsilon < T_\sigma$), and are not related to X ($\epsilon \perp X$).[2] Thus, the single functional equivalence statement infers three distinct hypotheses. These in turn suggest statistical tests; here we might employ a t-test to decide if the mean is 0, re-sampling to obtain a confidence interval on the variance, and correlation to determine if ϵ is related to X.

The explicit mapping from statements to hypotheses and from hypotheses to analyses works in the opposite direction as well. Suppose the analysis shows that the residuals have mean 0 and low variance, but are also positively correlated to X. This suggests that the original statement needs to be modified, increasing either the coefficient or the exponent of X.[3] Thus, explicit knowledge also supports the formulation of new statements.

---

[1] The transformation from statement to hypothesis often involves the creation of new variables.

[2] In some cases, it is not necessary to verify all three conditions. For example, if the statement $Y = 3X + 5$ were derived from the method of least squares, the residuals ϵ are independent of X by definition.

[3] Increasing the exponent under these conditions is essentially the heuristic introduced by BACON.3 [8].

2.3 Automated Experimentation

One of SEA's important features is its ability to design and run experiments with a *simulator*. For our purposes, the simulator can be any LISP-based computer program. SEA generates an experiment plan for use by the instrumentation package CLIP [2], which in turn gathers experiment data. Because the apparatus for the experiment resides alongside SEA in the computer, there is no immediate need to interface with the real world. However, the experiment planning capabilities based on CLIP should extend easily to a real-world interface through commercially available devices (e.g. [5]); unfortunately, there is no current standard for managing such devices.

In restricting our experiment-design capabilities to the manipulation of LISP-based simulators, it may be possible to address one of the most difficult questions in scientific discovery: how are *new* experiment variables created? Our working hypothesis is that new variables are derived from knowledge about the structure of the world; for example, in the world of LISP, any *global variable* is potentially an interesting experiment variable. Thus, knowledge about LISP helps to resolve a critical issue in experiment design. We hope to identify a variety of variable creation techniques for simulators with known structure, such as those written for the simulation substrate MESS [3].

3. System Design

Figure 1 provides an overview of SEA's operation. The shaded area contains the various types of data managed by the system, while the outer region shows the *goals* associated with each type of data. We also see the roles of other agents, including the user, the simulator, and the instrumentation package CLIP, as providers of information.

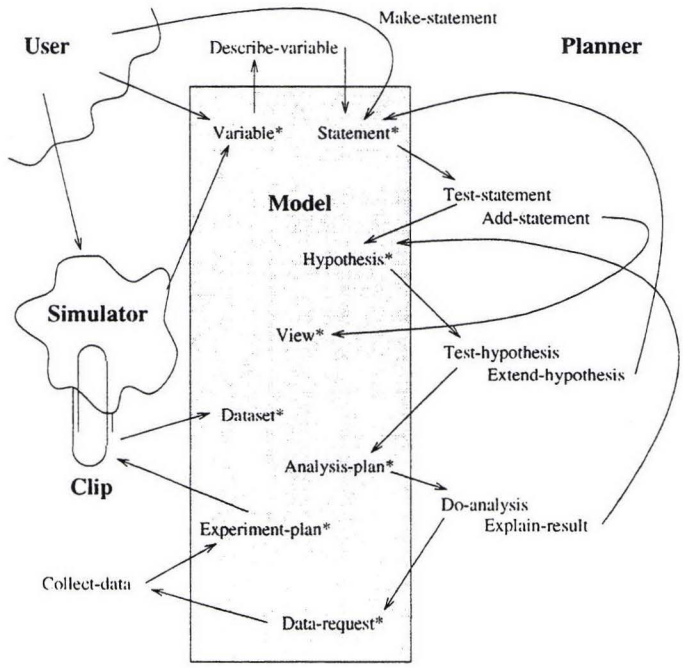

Figure 1: Overview of sea's model-building strategy.

Starting at the user region, SEA's implementation of the scientific method follows the arrows down the right side of the diagram and up the left side. First, the user provides a set of experiment variables and one or more statements about the expected behavior of the variables. Then, SEA turns the statements into hypotheses, and the hypotheses into analysis plans. Each analysis plan generates a request for experiment data, which is eventually gathered by CLIP. Once the data is collected, new knowledge is created moving up the right side of the diagram — new hypotheses are generated by explaining the results of analysis, and new statements are generated by combining verified hypotheses.

3.1 Intelligent, User-Oriented Control

Each of the operations employed in constructing the model is stored as a plan fragment for a *partial-hierarchical planner* (PHP) [6]. PHPs construct a plan incrementally from plan fragments, interleaving planning steps with execution of the plan. While this policy can result in a course of action that is less than optimal, it is necessary in dynamic domains such as model-building, because each action may have a significant impact on subsequent planning decisions.

Each plan fragment describes an action that satisfies a specific goal given a set of preconditions. The functional part of the plan is composed of elementary programming constructs, such as conditionals, loops, and variable bindings. In addition, it may post new goals, imposing a hierarchical structure on the evolving plan. Thus, while classical planning is essentially bottom-up, composing elementary actions into a high-level plan, PHP is top-down, gradually instantiating an abstract plan until an executable action is encountered.

SEA uses a planner originally developed for a similar system, the *Assistant for Intelligent Data Exploration* (AIDE) [10]. This planner employs a mechanism called *focusing* to manage decision points in the evolving plan. Whenever a new goal is posted, the planner selects a set of plan fragments that unify with the goal. If the set is empty, the goal cannot be satisfied. If there is only one plan in the set, it is executed incrementally. If many plans are available, the planner creates a *focus point* for the associated goal. Before execution can proceed, one of the plans in the focus point must be selected. If the plan fails or makes poor progress, the planner can be "refocused" to another plan in the focus point.

In our initial version of SEA, the selection of a plan from a focus point is entirely the user's responsibility, as are decisions to refocus the planner. However, as the plan library grows with new statements, hypotheses, and analysis techniques, these decisions will become more frequent and may overburden the user. Thus, we are interested in developing policies for automatic focusing and refocusing. Many such policies can be hand-coded from successful scientific strategies, such as the "scientific discovery" heuristics of programs like Bacon [8] and IDS [9].

3.2 Structured Domain Knowledge

Knowledge about the scientific domain in question is acquired from the user on an as-needed basis. This knowledge is stored as collections of statements called views. A view is a set of statements with a consistent interpretation of simulation behavior. When a new statement is added, it is incorporated into all views that it agrees with; if the statement would introduce an inconsistency, a new view is created that contains a consistent subset of the original statements. For example, the statement that optimal performance depends on the size of the network is inconsistent

with the statement that a policy is optimal; thus both statements should not reside in the same view. In this way, SEA maintains multiple perspectives on behavior at varying levels of detail.

SEA's connection to the "real world" revolves around its experiment variables. These variables are hooks into the simulation itself, and as such represent either parameters of the simulation that are specified in an experiment protocol, or measurements of simulation behavior taken during or at the end of an experiment trial.

Each statement describes some expected relationship among the experiment variables, and thus induces structure on SEA's knowlege of the domain. Some statements may be provided as "givens" by the user, such as the fact that optimal running time for a network is the total number of unit-sized tasks divided by the size of the network. Other statements represent empirical questions, and these are verified against experiment data. When SEA reaches an impasse in its scientific process, it may request that the user identify new experiment variables or supply additional statements about the variables.

4. Conclusions, Status, and Future Work

SEA is an assistant for human scientists who study the behavior of computer simulations. Although this is only a subset of all possible scientific domains, it is an important subset, because any real-world system can theoretically be simulated in a computer. This restriction to studies of simulation behavior allows SEA to quickly come up to speed at the task of experiment design.

SEA has a flexible design that is not based on any specific analysis technique or type of hypothesis; rather, it focuses on the tasks involved in any type of analysis, such as formulating a hypothesis, gathering appropriate data, and explaining the results. In this way, SEA creates a language for defining analysis techniques, grounded in the notion of a *statement type*. Users should be able to easily incorporate new analysis techniques into SEA's plan library by encoding a set of plans appropriate to the technique.

The SEA architecture provides a framework for studying a variety of issues in automated scientific discovery. These include techniques for designing experiments and creating new experiment variables, the creation of new, computer-intensive analysis techniques, and the formalization of effective scientific strategy. It is our goal to provide a system that is flexible and efficient, in order that a variety of research in the area of scientific discovery can be integrated on a single platform.

4.1 Status

The current version of SEA implements many techniques employed in the aforementioned study of load-balancing policies, as well as a variety of other strategies based on statistical analysis. These techniques focus on two types of statements: functional equivalence and the effects of a treatment variable. They consist of five different plans for generating hypotheses from these statements, and statistical hypothesis tests including the t-test, analysis of variance, linear regression, and resampled confidence intervals. Each test generates a request for data, which is fulfilled by a simple experiment design in which each independent parameter is varied over at most three values. Once analysis is complete, SEA can generate new hypotheses to explain the results; for example, it suggests that overhead may differ between the two policies even though their net running time is not significantly different.

In addition to the plans associated with hypothesis testing and experiment design, SEA also has facilities for acquiring an initial domain model based on simulation variables. It also maintains a chronological record of all actions taken and the results derived, in the form of a scientific notebook. We are currently in the process of implementing a graphical user interface on the Macintosh^{TM}.

A formal evaluation of SEA is planned for the near future. The evaluation addresses several key issues, such as the applicability and efficiency of the system, its flexibility in supporting new modelling techniques, its ability to scale efficiently and coherently, and its suitability as an assistant to human scientists. The evaluation focuses on SEA's performance in studying several disparate domains, including the load-balancing problem, a comparison of sorting algorithms, the discovery of predictive rules in a relational database, and a simulation of a real-world planning problem, such as transportation planning or air-campaign planning.

4.2 Opportunities for Learning

One interesting question we hope to explore regards the automation of procedures for selecting an appropriate action at a focus point. A possible approach applies reinforcement learning [4] to acquire a policy that selects the plan most likely to succeed at achieving its goal, given the current context. For example, the user may prefer certain types of analysis, and different users may have different criteria for deciding whether an analysis was successful. Other context variables may include the domain being studied, v the plan that posted the current goal, and the goal that triggered that plan. In addition, success may depend on the bindings of plan variables, such as the hypothesis being tested, the significance of a test, summary statistics, and so on. Although policies learned in this way are guaranteed to improve system performance in the limit, it is at issue whether SEA will be able to gain enough experience with different users, domains, and analysis techniques to acquire a truly effective policy.

Acknowledgments

This work is supported by DARPA/Rome Laboratory under contract number F30602-93-C-0076, and by a National Science Foundation Graduate Research Fellowship. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes not withstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency, Rome Laboratory, the National Science Foundation, or the U.S. Government.

References

[1] James F. Allen. Mixed initiative planning: Position paper. Presented at the ARPA/Rome Labs Planning Initiative Workshop. See URL http://www.cs.rochester.edu/research/trains/mip/, 1994.

[2] Scott D. Anderson, Adam Carlson, David L. Westbrook, David M. Hart, and Paul R. Cohen. CLASP/CLIP: Common Lisp Analytical Statistics Package/Common Lisp Instrumentation

Package. Technical Report 93-55, University of Massachusetts at Amherst, Computer Science Department, University of Massachusetts, Amherst, MA, 1993. This document is available under http://eksl-www.cs.umass.edu/publications.html.

[3] Scott D. Anderson and Paul R. Cohen. On-line planning simulation. In *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems*, pages 3–10, 1996.

[4] A. Barto, S. Bradtke, and S.T. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.

[5] National Instruments Corporation. Labview is a visual programming environment for control of data acquisition products. See http://www.natinst.com/labview for more information.

[6] Michael P. Georgeff and Amy L. Lansky. Procedural knowledge. *Proceedings of the IEEE Special Issue on Knowledge Representation*, 74(10):1383–1398, 1986.

[7] Dawn E. Gregory, Li-Xin Gao, Arnold L. Rosenberg, and Paul R. Cohen. An empirical study of dynamic scheduling on rings of processors. In *Proceedings of the 8th IEEE Symposium on Parallel and Distributed Processing*, 1996.

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

[9] Bernd Nordhausen and Pat Langley. An integrated approach to empirical discovery. In Jeff Shrager and Pat Langley, editors, *Computational Models of Scientific Discovery and Theory Formation*. Morgan Kaufmann, 1990.

[10] Robert St. Amant and Paul R. Cohen. A planner for exploratory data analysis. In *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems*, pages 205–212. AAAI Press, 1996.

[11] Jan M. Zytkow, Jieming Zhu, and Abul Hussam. Automated discovery in a chemistry laboratory. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI-90)*, pages 889–894, 1990.