CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

IRMA: A SUPERVISORY CONTROL ARCHITECTURE FOR PRODUCTION CONTROL

by
Eric R. Johnson,
Michael R. Fehling,
William B. Poland

TECHNICAL REPORT
Number 89

July, 1993

Stanford University

If you would like to contact the authors please write to:

c/o CIFE, Civil Engineering,
Stanford University,
Terman Engineering Center
Mail Code: 4020
Stanford, CA 94305-4020

CIFE TECHNICAL REPORT #89

Title: IRMA: A Supervisory Control Architecture for Production
Authors: Eric R. Johnson, Michael R. Fehling, William B. Poland
Publication Date: July 1993
Funding Sources: CIFE, "IRMA — Intelligent Real-Time Monitoring and Assessment for Facility Management"
SIMA, "Real-time Model-based Reasoning for Intelligent Process Management.

1. Abstract:

We describe the IRMA framework for supervisory control of complex, distributed production processes and describe an application of this framework to emissions control and troubleshooting for geothermal power plants. We describe Gaussian influence diagrams, the mathematical formalism upon which our control regime is based, and we show how the Schemer multiprocessing architecture supports a demonstration of applicability of our controller and diagnostician to the plant by supporting execution of a simulation model of the plant and integrating them with it.

2. Subject:

This report describes a system that makes it easy to build control systems and troubleshooting programs for production plants. There are three main ideas here: a multiprocessing system that supports process communication can be very helpful in building a control system, building a control system based on a model of the system makes it easier for the system to be built and maintained, and Gaussian influence diagrams is a computation approach that is simultaneously easy to formulate and mathematically rigorous.

3. Objectives/Benefits:

CIFE funded this research because the issues of sensor fusion, reflective control of processing, and complexity that we address with Gaussian influence diagrams and a multiprocessing architecture are equally applicable to facility engineering as to production control. In addition, the paradigms for human-computer interface explored in the project can be useful beyond the domain of facility engineering. Our work aimed at showing that this supervisory control architecture is helpful for building a plant control system.

4. Methodology:

The research was in the form of a proof-of-concept demonstration. After extensive knowledge engineering interviews with PG&E personnel, we constructed a simulation of geothermal plant operations, including crucial features such as: inherent variability of the system and interactions among subsystems, as well as modeling the behavior of thermal and chemical processes in the plant. We then identified which data are available online for a controller such as we are construction, and which controls can be effected online, and created a structure in which only those data and controls were available to the controller and diagnostician. Finally, we built a controller and diagnostician and integrated them with the system, showing that their performance was good.

5. Results:

We found that GID-based control responds appropriately to a wide variety of routine and unusual chemical emissions scenarios, and that in most cases a diagnostic component based on a diagnostic tree can identify sources of thermal cycle inefficiency while calling on the operator to make only those investigations that would be called for by an experienced thermal engineer. Our project created a simulation model of the plant that could be used as a training tool, or for what-if analysis, as well as serving as a testbed for our controller and diagnostician. These components constitute a second major product of the research. A third result of the work is a graphical user interface that shows the progress of the GID solver in an intuitive fashion as it progresses. A fourth product of the research is an extensive HyperCard application documenting many features about the plant, both those simulated, and others as well, in an easily browsable causal network format.

6. Research Status:

The proof of concept is complete, and the system for the Geysers is ready to proceed to the prototype stage. In addition, the IRMA framework and the GID software are usable for a plant control project at other production facilities.

IRMA: A Supervisory Control Architecture for Production Control[1]

Eric R. Johnson[2]
Michael R. Fehling
William B. Poland[3]
email: eric, fehling, and poland @lis.stanford.edu

Laboratory for Intelligent Systems
Room 321 Terman Engineering Building
Stanford University
Stanford CA 94305

Abstract

We present the IRMA (Intelligent Real-time Monitoring Associate) framework for supervisory control of complex, distributed production processes. IRMA integrates decision analytic, optimization, and AI-style tools into a multiprocessing problem-solving environment. We describe the application of IRMA to emissions control and thermal efficiency monitoring at a geothermal power plant. This application uses the Gaussian Influence Diagram (GID) model for control by unrolling a continuous process into a sequence of discrete-time control decisions and optimizing these, taking account of previous observations of the system state. It employs a simple form of Model-Based Reasoning for diagnosis of faults and sources of thermal inefficiency. We give a tutorial introduction to GIDs and describe approaches to model specification and repeated construction of the GIDs used for emissions control. We give examples of the operation of both the controller and the diagnostician, and identify shortcomings of the latter that suggest the use of a Mixture-of-Gaussians ID, which allows discrete variables as well as continuous ones. We also discuss knowledge acquisition and representation issues, and

---

[1]A substantial portion of this document is published as Johnson et al. (1993).

[2]This research was supported by a grant from the Stanford Center for Integrated Facilities Engineering and the Stanford Integrated Manufacturing Association. The grant was in effect from May 1990 to May 1991, but the research continued after that time. Our work was assisted by the cooperation of PG&E personnel, who are not to be blamed for any shortcomings of this document.
[3]Mr. Poland's work on this project was supported by a grant from Intel Corporation.

we describe the Schemer multiprocessing environment upon which IRMA is based and show how it supports our approach to supervisory control. Schemer's flexible support for multiple communicating processes allows a robust and intelligent control system to be created. We conclude with comparisons of IRMA to other decision analytic and process-control applications.

Table of Contents

1. Problem: Control of complex, distributed production processes

This report describes a research project undertaken in 1990 and 1991 in conjunction with the Stanford Center for Integrated Facilities Engineering and Pacific Gas & Elecric. The report has four major sections: introduction to the problem, our general solution approach and research objectives, results of our research, and evaluation of it in light of our research goals. In addition, we give directions for future work, we compare our work to other research, and we give an introduction to Gaussian influence diagrams along with detailed specifications of the controller and results of tests of the controller and diagnostician. We begin by discussing the task domain in general, then our specific focus on geothermal electric plant operations, and we state the research objectives that motivated our choice of focus and our overall approach to the problem.

1.1. General Issue: Supervisory control of production

The IRMA (Intelligent Real-time Monitoring Associate) architecture provides computer-based problem-solving support for supervisory control of complex, distributed production processes. The supervisory control task has a number of attributes that constrain the functionality of a problem-solving architecture, including 1) real-time performance requirements, 2) need for modeling of continuous, physically distributed, and concurrent processes, 3) significant uncertainty about plant state, and 4) diverse task requirements. If a continuous product is being produced, either a continuous representation of it is required, or a discrete approximation of suitably fine granularity must be employed. A physically distributed production process creates concurrent events – events going on at roughly the same time as each other but without any direct causal or informational link. Supervisory controllers of such processes must integrate information from diverse sources, control individual production units, and coordinate their concurrent actions. This often requires balancing apparently incommensurate goals. Uncertainty about system state is a persistent problem, due to imperfect or noisy sensors, limited instrumentation, and the inherent unobservability of critical portions of plant operations. Physically distributed production systems and their control technology compound these problems with data transmission delays, inconsistency of data formats, and unreliability of data transmission facilities. Process management has diverse task requirements, including the following: monitor the process, determine/assess current situation, forecast/model situation evolution, plan/choose control actions, and carry out the appropriate control actions.

Extant research addresses each of preceding issues individually, but an integrated approach to these concerns remains to be established. Conventional production-control systems have addressed the continuous, real-time aspects of the task and, to some extent, have addressed coordination of concurrent distributed processes. Independently, decision analyses have been performed to address uncertainty and evaluate tradeoffs among production objectives (e.g., Chao et al., 1985; Keeney et al., 1986). In addition, rule-based systems have been constructed that apply the methods of knowledge-based problem-solving from the field of artificial intelligence to supervisory control (Shahidehpour and Kraft, 1986). Our research aims to integrate the best features of these approaches.

1.2. Specific Application: Geothermal electric plant control

In pursuit of such an integrated approach, we are developing IRMA, a computer-based system to control geothermal production of electricity at Pacific Gas and Electric's (PG&E's) Geysers facility in northern California. In plants at the Geysers, steam from underground sources contains impurities that damage machinery and produce environmental pollution, which must be abated by costly chemical treatment processes. Supervisors of this process strive to simultaneously maximize production efficiency, minimize emission of pollutants, and limit plant deterioration, by monitoring on-line sensors, investigating and diagnosing problems, and taking remedial action if needed. Critical decisions must be made in real-time to balance these competing concerns.

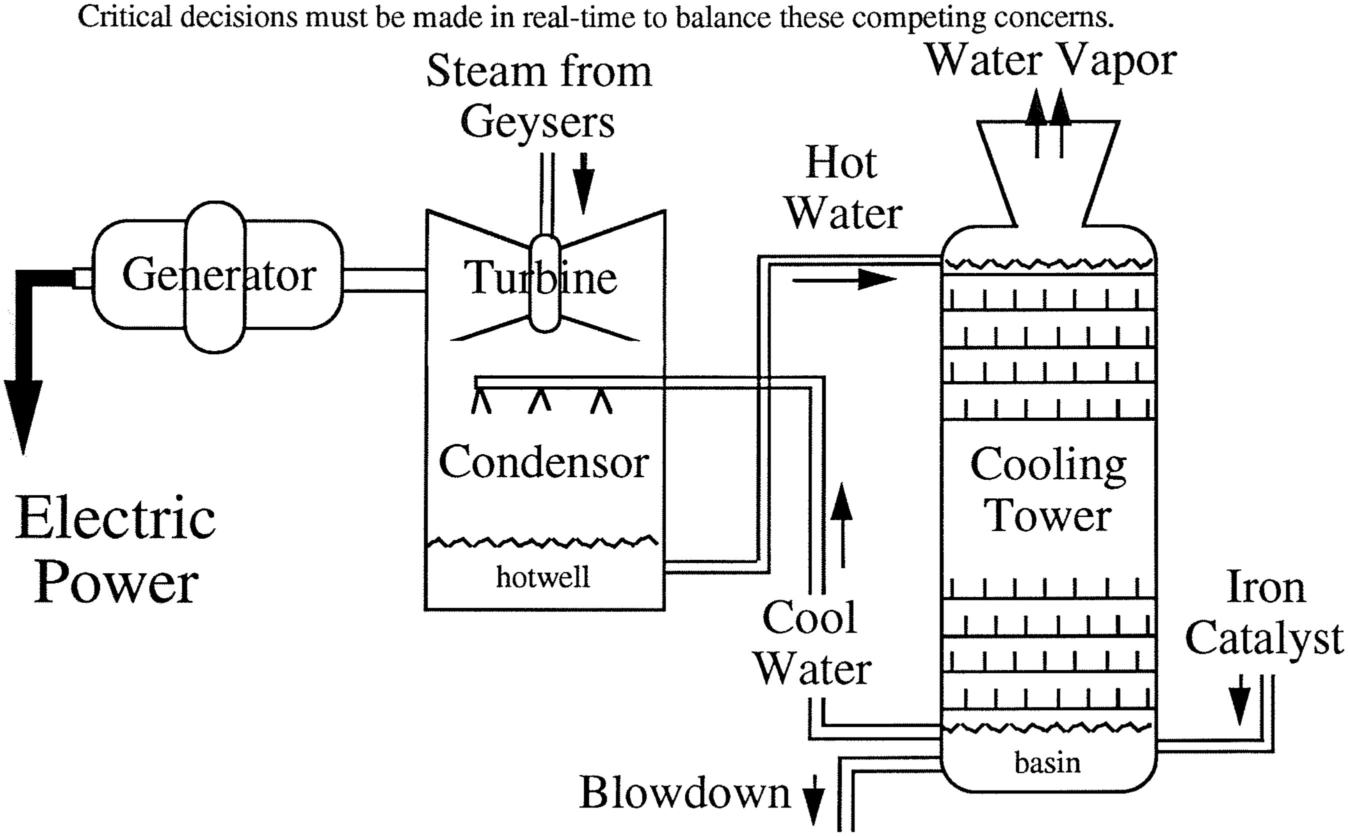


Figure 1. Components of a geothermal generation unit

Our project was undertaken for units 5 & 6 of the PG&E West Geysers field. The following description is true for those units, but other technologies are used for cooling and chemical abatement at units in the East Geysers field. Steam in underground formations is mined and piped to a turbine, whose rotation drives a generator. Each unit generates 55 MWs if it gets adequate steam. West Geysers units have direct-contact condensers, in which the cooling water returning from the cooling tower is sprayed through the steam to cool it. The hot condensed steam is pumped to the cooling tower, where it is cooled by evaporation and convection. Most of the incoming steam evaporates. The rest, called blowdown, overflows the tower basin, and is captured and pumped back into the ground. A cooling tower is made of five cells, all in a row. The condensate is poured onto horizontal plywood headers across the top of the cells. These headers have small holes, which, together with the baffles inside the tower cells, disperse the condensate for maximal cooling.

The incoming steam contains H_2S, which is noxious if released to the atmosphere, and which turns to sulfuric or sulfurous acid in water. First and second stage jets blow noncondensible gases, including H_2S, from the condenser through the inter- and after-condensers (collectively, the auxiliary condensers) to the incinerator, where the H_2S is burned, turning it into sulfite. The remainder from this process is returned to the condenser. This reduces emissions and keeps backpressure down. H_2S in the condensate must be treated by adding iron chelate to the tower basin. This frees the hydrogen, leaving only sulfur, which combines with the sulfite to make water-soluble thiosulfate, which may be released to the environment.

$$H_2SO_3^- + S^0 \rightarrow H_2 + S_2O_3^=$$

Since the iron chelate is constantly lost with the blowdown, incineration is the cheaper form of H_2S abatement. Thus it is desirable to maximize partitioning, the portion of the H_2S that is in the gases. High pH in the basin makes the abatement in liquid phase more efficient, but it reduces the partitioning. Acid or caustic soda can be added to keep pH in its optimal range, which is between 6.5 and 7. While this is only mildly acidic, other parts of the system are more so, and the physical plant is continually worn down by this.

Roughly 200 sensors at units 5 & 6 can be read from the plant operator's console. The automated plant monitoring system records and time-stamps all data. In addition, periodic inspection and chemical analysis of the condensate and effluent gases are performed to get additional information about the state of the system. Some control activities can be initiated from the console, while others require physical manipulation of the component in question. The manager of this process strives to maximize production of electricity, minimize emission of pollutants, and prevent damage to the plant, by monitoring on-line sensors, investigating and diagnosing problems, and taking remedial action if needed. We aim to help her/him in this task with the IRMA supervisory control system.

1.3. Research objectives

Our research objectives relate to our approach to supervisory control as a sequence of episodes in which control problems are solved rationally. The goals motivating our development of IRMA are to demonstrate the usefulness of a supervisory control environment with multiple problem-solving components to the PG&E Geysers geothermal plants, to find a systematic approach to choosing an appropriate solver from among these tools and creating problem-solving models, to identify a set of tools that constitutes an appropriate environment for process management, and to find a knowledge representation that is congenial to the experts' way of thinking, and that supports this form of problem-solving. Another of our long-term research objectives is to determine how a system like IRMA can best function on the spectrum from full automation (Tse and Fehling, 1989) to providing advice and guidance to human process managers (Rutledge et al., 1989; Paterson and Fehling, 1992). The goals motivating our development of Schemer are to develop an architecture that supports rational use of process-management resources (CPU, RAM, data, solvers), and to articulate and realize a model of rational community problem solving. Our particular research goal for this project with PG&E is to identify the benefits and drawbacks of GIDs as one problem-solving approach in the repertoire of this problem-solving approach.

2. The IRMA approach: DA and Supervisory Control

To respond to the diverse challenges of geothermal process management, our research emphasizes coordinated and integrated use of multiple, concurrent problem-solving (PS) methods. At a basic level, our desire to create a supervisory control system that behaves rationally leads us to focus on decision analytic approaches to control, insofar as decision analysis has an explicit philosophical orientation toward rationality. However, recently there have been suggestions that the rationality of DA is flawed if it does not balance the costs of identifying the most rational action against the benefits of this action. Because process managers must seek to maximize performance under real-time constraints and uncertainty, we focus on methods that embody what I. J. Good (1963) has called "type II rationality," balancing timeliness of task completion against performance optimality. We are investigating a range of PS methods that realize different tradeoffs of optimality against performance efficiency, rather than relying solely on optimizing procedures that may squander precious time.

This paper focuses on three key aspects of IRMA: continuous monitoring and choice of control decision methodology, use of decision analysis as one such methodology, and the Schemer architecture to support integration of various solution processes . Other IRMA features already in place or contemplated include a decision tree analyzer, dynamic model construction (DMC), planning and heuristic-search methods, truth maintenance, additional optimization routines (e.g., linear or non-linear programming), support for temporal projection, and architectural support for prototyping, such as checkpointing, stepping, trace, and scenario scripting.

2.1. Model-based Supervisory Control

We base IRMA's approach to process management on the following conception of the PS steps entailed by process monitoring and control: 1) detect and diagnose potential problems by assessing performance data in terms of a high-level diagnostic model of the system, 2) choose a solution methodology in light of problem solving resources available and the apparent nature of the problem, 3) formulate and solve the problem, 4) implement the solution, and 5) revise the diagnostic model to reflect problem occurrence and the response undertaken.

Step 2 illustrates our commitment to using multiple, alternative PS methods when performing complex tasks under realistic conditions. Note that use of a specific solution method requires that problem representation be appropriately framed.[4] We refer to a problem, as framed for a chosen PS method as a *solution basis*, and to the process of framing as *dynamic model construction* (DMC). We are investigating automated DMC for systems like IRMA that employ multiple PS methods each requiring construction of solution bases from a common knowledge-representation substrate. We are addressing knowledge representation requirements of a system employing a heterogeneous set of PS methods. Subsequent publications will describe our work on DMC.

2.2. Decision Analysis

Optimization of a model of the system is important for control applications, because it offers a basis for formal analysis of the goodness of the control. By reasoning about the model, we can endeavor to characterize the impact of approximations in the model on the control policy and thereby show how close the resulting control of the system is to the best

[4]Use of a heuristic search methodology would require formulation of the problem in terms of state, operators, stipulation of search heuristics, specification of goal states, etc. Alternatively, use of decision modeling methods such as those we discuss in the sequel require problem formulation by identifying the range of decision (action) options, the collection of outcome states that could be reached by those actions, a set of added decision variables that contingently affect the relationship between actions and outcomes, a joint probability distribution over these factors encoding their probabilistic interactions, and a value or utility measure comparing the relative desirability of the alternative outcomes.

that could be achieved with a given understanding of the system. Decision analysis (Howard, 1966) is an optimization approach that explicitly models both preferences and uncertainty. We feel that the ability to balance dissimilar goals (i.e. to support tradeoffs among preferences), and to represent uncertainty probabilistically motivate a central role of decision analysis in control of systems like the Geysers plants, where goals are heterogeneous and information is imperfect.

Our general approach to control is motivated by our focus on decision analysis. Closed loop control makes continuous use of process feedback. Open loop control allows a control policy to be in force for an increment of time without reference to the system state within that time period. We employ iterative open-loop control with a suitably fine time-granularity because it achieves most of the benefits of closed loop's coupling to the process while allowing each decision point, in principle, to be a decision modeling step, incorporating value analysis, assessment, problem-structuring, and sensitivity analysis as needed.

2.3. Schemer runtime environment

In view of the heterogeneous array, and possibly simultaneous occurrence, of problems faced during process management, our research emphasizes ways to coordinate and integrate multiple, concurrent problem-solving (PS) tasks. And because process managers must seek to maximize performance under real-time constraints and uncertainty, our work addresses on the concerns of what I.J. Good (1963) calls "type-II rationality" balancing timeliness of solving a problem or completing a task against solution optimality. This motivates our use of a range of PS methods, embodying various tradeoffs of optimality against performance efficiency rather than relying solely on formally optimal but computationally costly PS procedures.

Schemer is, among other things, a runtime environment which supports preemptive prioritized scheduling, communication (both synchronous and asynchronous), and coordination of multiple concurrent problem-solving computer processes (Fehling et al., 1989). We employed the Schemer architecture to support and integrate the problem-solving tools described here. Schemer manages attention and supports the "choice of solution" approach, allowing either quick or careful focused response as appropriate. Schemer's communication management infrastructure can help integrate sensors & human observations. It support for synchronous communication supports quiescent handlers which are triggered by pertinent information. These points will be fleshed out in the next section of this report.

A community may manifest resource-bounded rational agency if it is composed of multiple agents, each with its own objectives, and the agents whose objectives contribute to the community's objectives are given the resources they need by the community. This conception motivates the Schemer runtime environment. Schemer supports a community of computational "handlers", subroutines with specific purposes which, taken all together, will accomplish the user's goals, if coordinated properly. Schemer supports a flexible range of communication approaches to allow these routines to coordinate their activities. A high-level routine with control of important computational resources (notably CPU time, but also access to the keyboard and the monitor, and eventually also RAM) implements the "community's" resource allocation policy in the form of CPU scheduling directives and the like. Each agent may pursue its objectives by means of an internal rational community of sub-agents with all the structure of the community in which it is embedded; or it may do so in an unanalyzable way.

3. Research results

We evaluate our experience with knowledge engineering, describe the Geysers application of IRMA, then discuss the two main components of the Geysers application, the iron feed rate controller and the thermal efficiency diagnostician, and finally assess the role of Schemer in IRMA.

3.1. Knowledge acquisition and representation

We address knowledge acquisition, knowledge representation, and finally a tool that we created to assist both processes in our research.

3.1.1. Knowledge acquisition

We worked with Kim Stucki, Brian Benn, and Shaun Brady of the PG&E Geysers facility. Messrs Stucki and Brady are thermal engineers, and Mr. Benn is a chemical engineer. We visited the Geysers site a half dozen times to interview them and others, taking careful notes at interviews, and gathering source documents. In addition, we interviewed PG&E personnel on the telephone and sent them periodic written summaries of our understanding of the situation as it developed. We attempt to evaluate the knowledge engineering approaches we used according to how effective they were at capturing expertise, and how useful the expertise was for problems-solving. This section compiles our subjective impressions of the efficiency and comfort level of our knowledge engineering interaction with the experts using various approaches to knowledge engineering; no effectiveness data was collected.

Knowledge engineering was complicated by the fact that some of the issues we asked about were apparently poorly understood, and the answers changed significantly during the course of the investigation. In some cases, this reflected a changing state of knowledge of the experts[5], and in others it may be that we simply misunderstood the first time we were told[6]. We conclude that one should cross-check fundamental facts about the application carefully before using them to structure the analytic approach in a fundamental way.

Sending periodic written summaries for the experts' review did not turn out to be a good approach to knowledge engineering (although it may be advisable for other reasons): the experts did not feel that they could offer any comments or corrections without doing a thorough rewrite of the documents we sent.

We produced a prototype model of crucial parameters of the Geysers system, arranged for a continuous bar-graph readout of any selected subset of variables during a simulation, and showed it to the experts for their feedback. We believe that this methodology can help reduce the information overload associated with a written report, allowing the expert to focus on overall qualitative and quantitative response of components of our model in "real time". But this did not elicit much feedback from the experts. We conclude that either we had gotten it all essentially right, but we cannot rule out the possibility that some flaw in our user interface prevented the domain experts from comparing the dynamics of our simulated plant to those of their actual one.

Use of a "causal" network[7] during interviews helped communicate to the experts the range of phenomena upon which we were focusing our work, and one of the experts was quite comfortable with a diagnostic tree, but none of them seemed comfortable following causal pointers in a diagram of twenty nodes. Perhaps a more experienced decision analyst could use this tool better, and perhaps it is more useful for specific decisions than for a general situation in which the variables were not clearly defined. The experts were comfortable giving setpoints, measures of variability, and measures of sensitivity of one variable to its causal predecessors (if the variables in question were numeric), once these were established.

Use of source documents was helpful, but not without costs. This approach trades knowledge engineer's time for domain experts' time. This can be helpful, but it is important for the knowledge engineer to vet all conclusions s/he draws with the domain expert, to avoid misinterpretation or misapplication of data.

---

[5]Initially it was said that low pH helps H_2S abatement, but subsequent instrumentation made it possible to test this, and the evidence turned out contrary to this in an important class of cases.

[6]We understood initial descriptions of water chemistry in the tower basin to suggest that concentrations equilibrate slowly (over the course of six hours), whereas later we were told that this takes place within an hour.

[7]We ultimately interpreted the arcs in the network probabilistically and elicted them as such, but we found that notions of causality were more comfortable for the experts. In addition, Shachter and Heckerman (1987) implies that an ID should be elicited and structured as if it were causal.

The experts seemed comfortable filling in frames corresponding to the various knowledge representations we investigated.

3.1.2. Knowledge representation issues/approaches

We investigated four general knowledge representation (KR) schemas: information and control, causal, taxonomic, and scenario-based. In addition, we feel it is important to explore KRs with temporal attributes; in our case we did not do this because all the mechanisms we studied are realized over the end of one time period.

An "information and control" schema comprising component, sensors, monitoring activities, and controlling activities proved easy for the experts to assess, and gave a helpful overview of the parameters of interest in the system, as well as giving specific information for structuring influence diagrams.

As noted above, the experts were not entirely comfortable with a causal network, but they could specify for any given system attribute what its normal value is, how much intrinsic variability it has, what its causal predecessors are, and how it responds to their variations. This data can prove helpful in constructing both influence diagrams and diagnostic trees.

We thought carefully about taxonomies or hierarchies, but found them to be of only limited value. Taxonomies have to do with nesting and instantiation of classes/aggregates. We had two generating units (numbers 5 and 6), but their interaction was weak enough that we treated them as independent. This was the only obvious class of repeated objects in our domain. We looked for other taxonomies and considered two closely, one by aggregate component (cooling tower, condenser, incinerator, turbine, steam pipes, weather), and another by underlying substance (H_2O, sulfur, hydrogen/pH, iron, corrosive chemicals, energy, time, gases). The taxonomy by component was helpful for knowledge acquisition, to make sure frame instances at all components were identified and specified. But it did not partition the variables in a modular way – there were strong interactions between variables pertinent to different components. In addition, subcomponents or attributes of components did not appear to share any useful properties – we could set up the taxonomy, but there would be nothing to inherit. The taxonomy by substance succeeded in decomposing our 80-attribute system into a set of attribute classes within which interactions are strong and between which interactions are weak. The "interactions" of variables normally corresponded to change in form of a bit of substance, e.g. main-steam-pressure is related to water-in-basin, which is related to water-evaporated. But again we found few useful properties shared by instances of these classes.

The taxonomy by substance did suggest formulation and use of a "conservation law": a substance is neither created nor destroyed within the system. We feel this may have been a productive approach, but our experts were loath to undertake this investigation. They had apparently considered it before, insofar as they had a names for two important prongs of this investigation: sulfur balance, and heat balance. Their reluctance may stem from the substantial deviations from the underlying conservation laws due to poorly understood interactions of the system with its environment (e.g. substantial loss of water volume due to unknown leaks).

We identified a number of problems/scenarios/syndromes, but these never proved useful in problem-solving; we always reasoned at a lower eventwise level without any particular problems. For the record, the syndromes are as follows: low iron concentration, sulfur solids fouling, turbine blade plating, bad partitioning, low pH/poor liquid abatement, air leak, incinerator trips off, and turbine loses blade.

3.1.3. Knowledge management tool

In response to the success of a close-in view of a causal network, the fact that representations of causality mesh nicely with the information-and-control knowledge schema, and the utility of this information for problem-solving, we built a knowledge base structured as a causal network. Each node corresponded to a feature/variable of the system, and each had arcs from its causal predecessors. At each node there was an informal frame of data: a verbal description that addressed the following topics for that feature:

- definition & measurement,
- base value (setpoint),
- variability (spread),
- sensitivity to predecessors, and
- numerical formulation of the above information (if any).

We found it helpful to cite the source of each fact entered in this database to facilitate resolution of apparent conflicts. We implemented the network in HyperCard, with one card for each feature/variable, and with lists of causal predecessors and successors that could be used for traversing the network. In addition, we maintained a picture of the network whose nodes were mouse-sensitive, allowing direct access to any variable.

These textual summaries of a feature's role in the causal network were helpful in structuring the body of knowledge for our understanding, they provided a scratch pad for derivation of formal relationships, and they served to document the models we subsequently created. Ultimately we intend to tie this database to automated model construction by imitating the way we used it in constructing solution models for the system manually.

3.2. Overview of our application

IRMA demonstrates the applicability to PG&E Geysers geothermal plants of a supervisory controller employing a GID-based iron-feed-rate controller for the emissions control system and a diagnostic tree processor to identify sources of thermal inefficiency. The systems showed how to integrate disparate sensor data regarding the following system attributes: weather, emissions, steam flow, chemical makeup of the circulating water, and thermal properties of the steam; and it filtered out measurement and process noise using the GID approach. The controller balanced H_2S emissions against financial cost when choosing an optimal chemical abatement policy. The controller used only data available online, and the diagnostician was generally able to diagnose plant mishaps with only limited human assistance. In this application of IRMA, the supervisory monitoring process is to be performed partially by the computer and partially by the human operator: the emissions control decision is made automatically every hour, and the operator is to invoke the thermal diagnostician when s/he feels it is needed.

For the Geysers demonstration project we wanted to prove that our system could be integrated with the data sources online at the Geysers, but we did not want to link the project under development to the actual plant. Instead, we constructed a simulation of the plant, and attempted to control this simulation. The model simulated the behavior of the plant under both normal and abnormal conditions (mishaps). We made sure to include a significant amount of noise in the behavior of the simulation, to ensure that the problem we solved was not oversimplified. The iron feed rate controller was only given data from the simulation that is available online; the diagnostician explicitly noted all requests for human investigation which its inferences required, and it was designed to minimize these. In addition to serving as a testbed for the controller and diagnostician, the model supported 'what-if' analysis wherein the analyst changes one or more aspects of system state and observes the response of the system.

3.3. GID-based iron feed rate control model

Control of iron feed rate is central to the emissions control process at the Geysers. We describe the structure of our control model, specification of its parameters, how it is used, and the results when it was used to control our simulation of the plant.

3.3.1. Why to use Gaussian influence diagrams

Gaussian influence diagrams (GIDs) are a particularly simple solution technique to formulate, because, unlike their more common discrete-valued counterparts, these influence diagrams of continuous-valued variables are completely specified by precisely the data we have. If we accept the GID's limitation that only a multivariate Gaussian distribution of state variables may be represented, we may formulate a well-stated decision problem by merely translating each of the elementary data given by the experts (setpoints, spread, and sensitivities) into single numeric values. We feel that this direct correspondence to the way our experts think about the problem is an important argument in favor of using the GID as a solution technique. Stated differently, use of GID obviates "discretization" of a continuous distribution and explicit assessment of probabilities.

An introduction to GIDs is given in an appendix. Shachter and Kenley (1989) define GIDs, gives a solution algorithm, and notes that the GID is able to implement the linear-quadratic-Gaussian approach to control. Its probabilistic basis allows GID to integrate disparate sensors & human observations robustly and defensibly. The explicit value function inherent in any ID allows one to specify tradeoffs among conflicting or apparently incommensurate goals. Furthermore, the use of this formalism provides assurance that the system can provide an appropriate response to all system states, in distinction to some rule-based approaches where some situations fall through the "cracks" of the database and engender no response at all.

3.3.2. Structure

With the guidance of PG&E process engineers, we formulated the emissions control problem at the Geysers as a sequence of hourly control decisions taking account of information from previous hours and the effect of current actions on future hours. A set of variables was identified whose joint distribution captures the entire impact of previous history on the current situation. These variables, labeled "-1" in figure 2, constitute such a "Markov state." The reduced complexity of the future hour (labeled "+1"), and the existence of only one such hour in our formulation, are based on our judgment that these captured the effects of the current decision on future circumstances adequately, the most important such effect being that most of the iron catalyst fed in now will still be dissolved in the condensate in the future. Figure 2 shows an influence diagram modeling crucial aspects of the iron feed rate problem as we understand it.[8]

This influence diagram represents the following physical characteristics of the system. The iron is fed into a basin of condensed steam (water), and it causes chemical reactions that reduce the emission of H_2S into the atmosphere. The basin loses some of its volume to blowdown every hour; hence the expensive iron catalyst must be fed into the system constantly to maintain appropriate emissions levels. The rate of blowdown is faster when there is more water in the system, which can occur either because there has been more incoming steam flow, or because a low ambient temperature (wet bulb) reduces evaporation of the condensate. The value function balances the cost of the iron against the amount of

[8]"No-forgetting" arcs (Howard and Matheson, 1981) into the second decision are omitted to reduce diagram clutter.

H_2S emissions. Other factors enter into this system, but their impact is less important. None of the variables except the value function is considered to be completely determined by its conditioning variables (i.e. none has zero conditional variance).

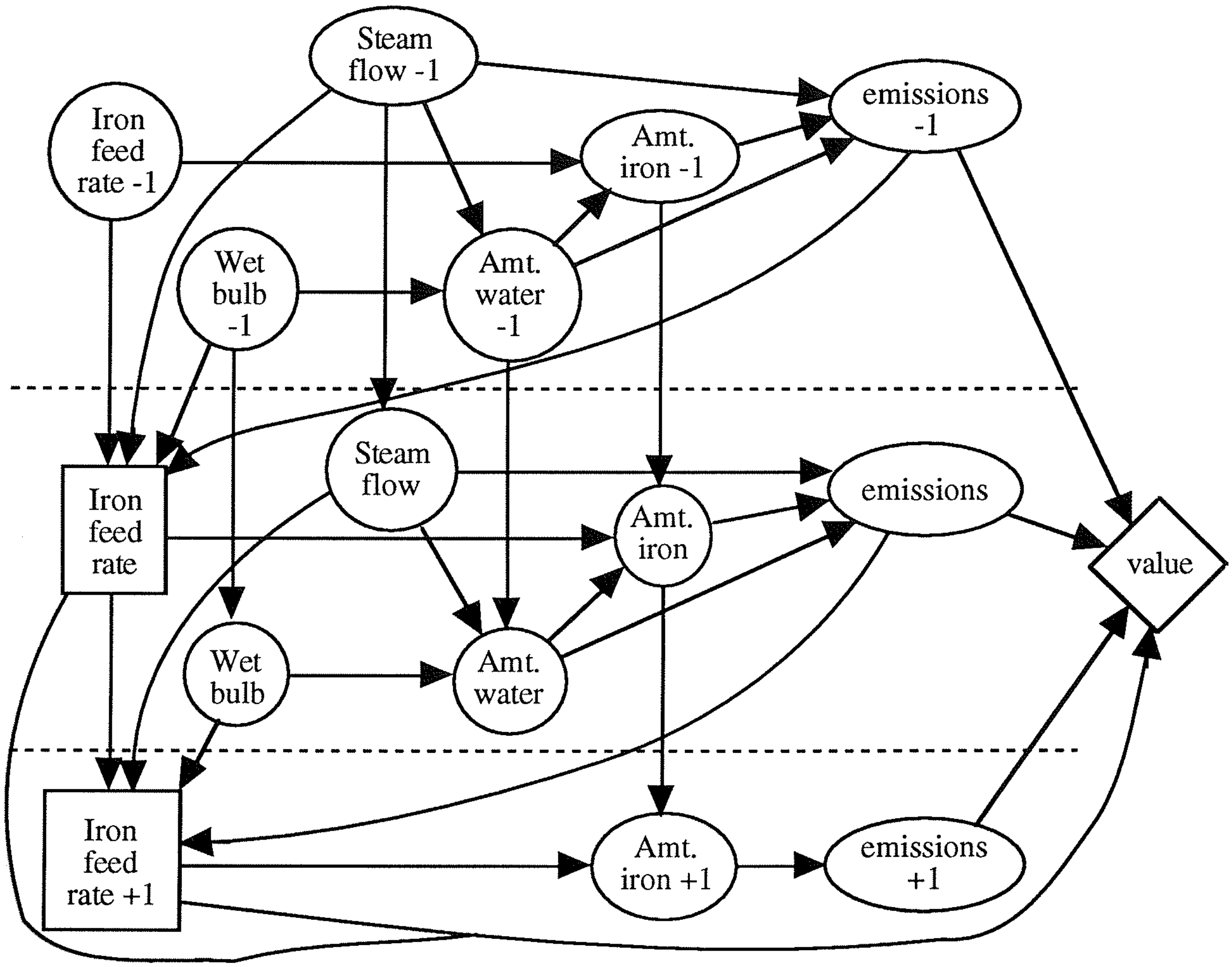


Figure 2. Geysers emissions control influence diagram

3.3.3. Parameters

To assess the parameters of the model, we interviewed the domain experts, studied system operations data, modeled the system's dynamics, and then created the closest approximation to this model directly expressible in a (linear) GID.[9] We found that an adequate controller can be created in this way. But linearizing the model conflicted with the notion that the chemical reactions characterizing the system are governed by ratios, not differences. In particular, the Geysers chemical engineer told us that the concentration of iron (the *ratio* of the amount of iron to the amount of water) was the crucial factor affecting H_2S emissions, and that its impact on emissions was quadratic, not linear. We found that

[9]A GID is necessarily a linear model because in a multivariate Gaussian distribution, the conditional mean of any variable given any others is a linear function of the given variables.

the inferences could be improved and the control of the system could be made more stable by applying the GID model to the logarithms of the original variables, allowing all the arc strengths into the emissions node to be interpreted as exponents of the conditioning variables, not coefficients of them. Under this interpretation, the strengths of the arcs from log steam flow, log amt.iron, and log amt.water into log emissions are 1, -2, and 2 respectively, implying linear, inverse quadratic, and quadratic influence of the original variables on emissions. This reformulation required that the distribution of the state variables be specified as multivariate lognormal, which had the helpful side effect of ensuring that all variables are always positive.

If a model has been specified using linear parameters, a logarithmic formulation whose marginal response around setpoints is locally identical may be derived from parameters of the linear formulation as follows:

To get:	Calculate:
mean of logs	log of mean
standard deviation of logs	standard deviation / mean
arc strength of logs	arc strength * mean of parent / mean of child

While the lognormal distribution specified in this way matches the local behavior of the corresponding Gaussian, it understates the mean and variance of the distribution somewhat. We initially formulated linear parameters for our model and then transformed them to logs to preserve local responses, using the formulas given here. The parameters of both the linear and logarithmic formulations are given below. A line is drawn dividing the arc-strength table into arcs within a time period and arcs between time periods.

	LINEAR PARAMETERS			LOG PARAMETERS		
NODE	MEAN @5	MEAN @6	ST.DEV.	MEAN @5	MEAN @6	ST.DEV.
Iron feed rate	4.4	4	-	1.5	1.4	-
Wet bulb	47	47	5	3.85	3.85	0.11
Steam flow	750	750	50	6.62	6.62	0.07
Amt. water	1875	1875	40	7.54	7.54	0.02
Amt. iron	21	20	2	3.04	3.00	0.10
Emissions	8	5.5	3	2.1	1.7	0.4

ARC FROM	ARC TO	LINEAR STRENGTH		LOG STRENGTH	
		@ 5	@ 6	@ 5	@ 6
Wet bulb	Amt. water	-6.2	-6.2	-0.15	-0.15
Steam flow	Amt. water	0.2	0.2	0.08	0.08
Iron feed rate	Amt. iron	0.4	0.4	0.08	0.08
Amt. water	Amt. iron	-0.0019	-0.0018	-0.17	-0.17
Steam flow	Emissions	0.01	0.01	1	1
Amt. iron	Emissions	-0.67	-1	-2	-2
Amt. water	Emissions	0.009	0.012	2	2
Amt. iron	Amt. iron	1	1	1	1
Amt. water	Amt. water	0.92	0.92	0.92	0.92
Steam flow	Steam flow	0.80	0.80	0.80	0.80
Wet bulb	Wet bulb	1	1	1	1

Figure 3 shows parameters of the logarithmic formulation of our problem for unit 5 in GID notation.

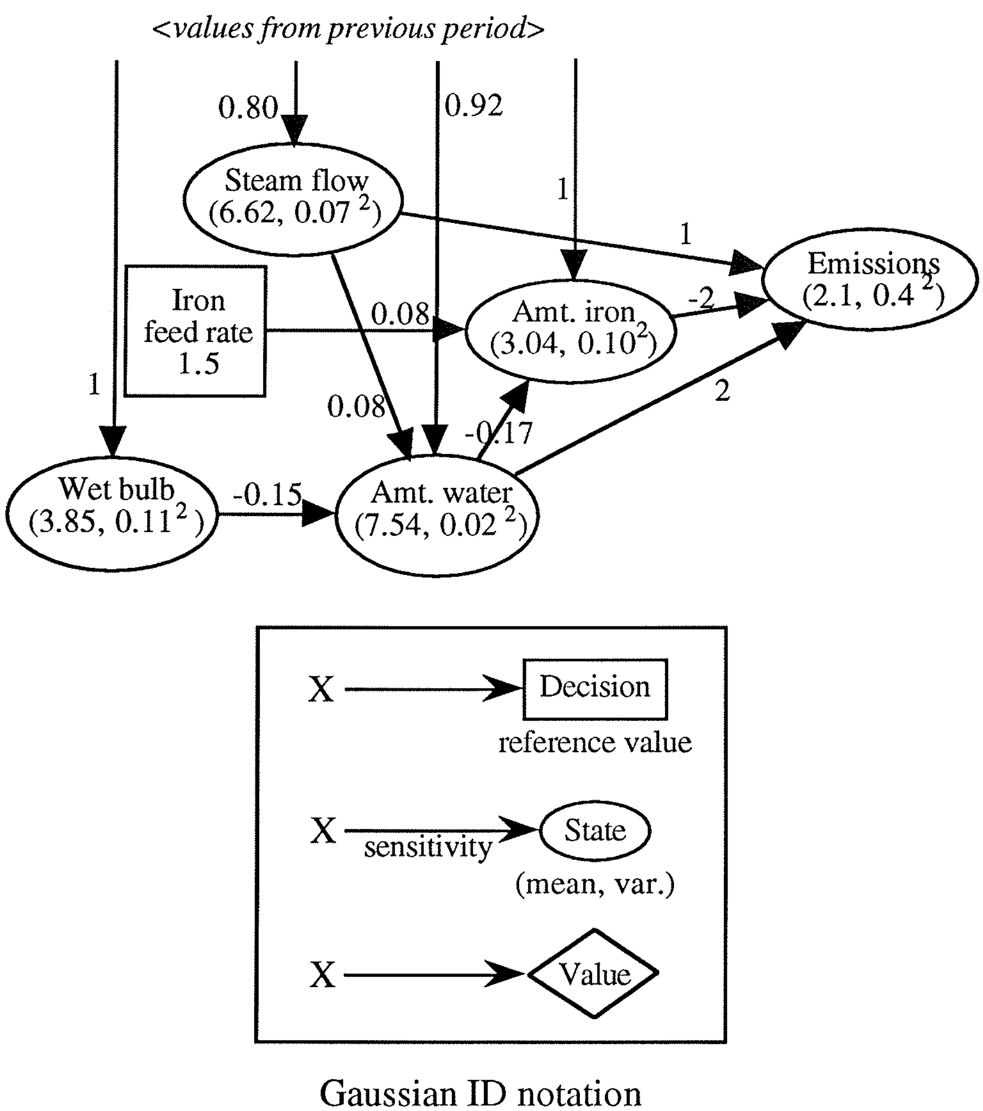


Figure 3. GID fragment, with parameters specified (logarithmic formulation)

The value node balances the cost of the iron catalyst against the costs of H_2S emissions. PG&E faces a regulatory ceiling on emissions of H_2S: if its plant is found to be in violation, it pays a large fine; otherwise it pays nothing. Therefore value is a step function of H_2S emissions, with the step at the regulatory ceiling. Value is also a linear function of iron feed rate (reflecting the cost of the iron catalyst). Note that two problems are evident here: matching a step function with a quadratic function, and matching the sum of a linear and a step function of the underlying variables with a quadratic function of those variables' logs. We solved both problems by creating pseudo-data of the costs in question for a set of typical circumstances and fitting a curve of the required form to those points. We then increased the H_2S cost parameters to represent the fact that the PG&E decision-makers place additional value on low H_2S emissions beyond merely avoiding a fine – PG&E is a "green" company. The resulting value functions for the linear and logarithmic formulations were:

$-0.25 \text{ emissions}^2 + 2 \text{ emissions} - 6.3 \text{ iron.feed}$ and

$-60 (\ln \text{emissions})^2 + 120 (\ln \text{emissions}) - 8 (\ln \text{iron feed rate})^2 - 14 (\ln \text{iron feed rate})$

3.3.4. Use

For each hour of this repeated decision-making process, we identified the following stages of formulation, inference and control: 1) create the generic ID, 2) update time-dependent variables, 3) propagate the previous hour's inferences, 4) update for current evidence, and 5) optimize. The first three steps constitute the problem-formulation phase of a decision-making process; steps 4 and 5 constitute solution of the decision problem. We begin with the ID shown in figure 3. Our database for the plant specifies mean and conditional variance for each chance node, as well as the strength of each conditioning arc. These specifications are generally independent of the time of day, with one important exception: the ambient wet bulb temperature varies systematically through the day. To address this, step 2 adjusts the means for the wet bulb nodes and propagates the impacts of the adjustments to nodes downstream from them, most importantly to the amount of water in the system. Step 3 accounts for the inferences drawn from the previous hour, by putting the previous hour's posterior probability distribution for its "current" hour into the "-1" hour variables and propagating the effect of those changes through the rest of the diagram. Note that there still is variance in the state nodes after these procedures. The fourth step is to absorb the evidence from the four nodes whose value is available online at the plant: iron feed rate, steam flow, wet bulb, and emissions, and to propagate this information through the rest of the diagram. After observation, the variance of these variables can be set to zero. Finally the GID solution algorithm is applied.

3.3.5. Performance

Figure 4 shows the behavior of the linear and logarithmic emissions controllers under an exogenous shock to the system, reduction of the simulated system's iron concentration to 3 ppm. This constitutes a more extreme shock than is likely to occur in practice, and it shows the responses of the system and the controllers clearly. In both cases the system had been running stably with normal iron concentration (14 ppm) prior to the shock, and the controllers had a relatively good inference of the iron concentration at that point. The inferred concentration shown here is the quotient of the inferred amt.iron and amt.water after absorbing other evidence from the period in question.

Since temperature, iron feed rate, steam flow and emissions are measured by online sensors at the Geysers, we made these values from the simulation available to the controller, and we show them here. Both the actual (simulated) and inferred values for the other two crucial parameters, amt.water and iron.conc. (=amt.iron/amt.water), are shown. Numerical data for this "3 ppm test" are given in an appendix.

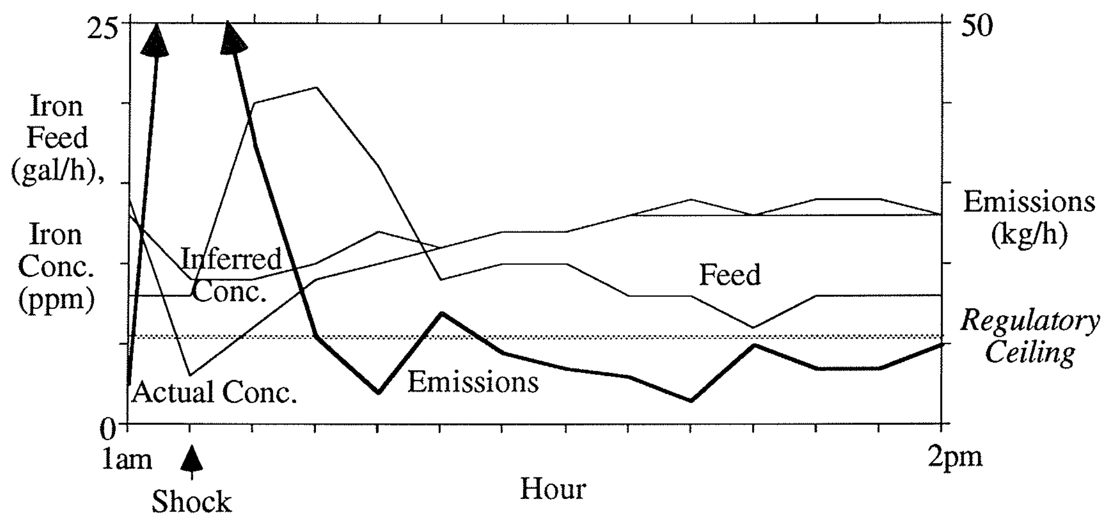


Figure 4(a) Logarithmic Controller

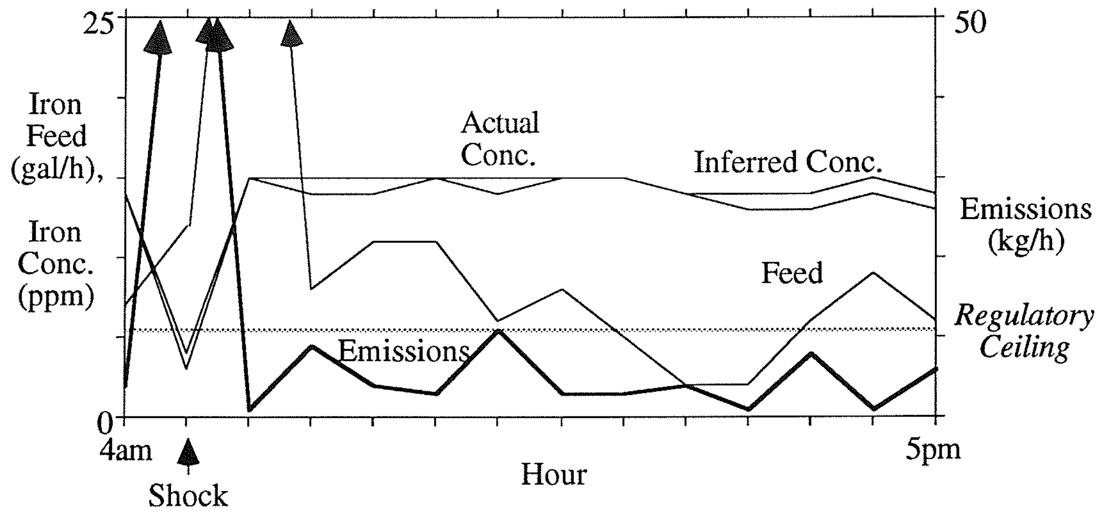


Figure4 (b) Linear Controller

Figure 4. Geysers model response to exogenous shock

Note that even when the iron concentration is held relatively constant, there is a great deal of noise in the underlying simulation of H_2S emissions relative to the regulatory ceiling of 11 kg/hr, so determining an appropriate target concentration is relatively hard. Both controllers return the system to an appropriate concentration level relatively quickly. The linear controller returns the system to its equilibrium more quickly, but it exhibits cyclical behavior once the system has returned to equilibrium because it over-responds to the noise in the system. The inferences made by both systems were good. The ability of the logarithmic GID to control our strongly nonlinear simulation suggests that transformations of variables may be a good general approach to make its restrictive assumptions reasonable. Its initial under-response to the shock is partially explained by our use of lognormal parameters which understate the mean and variance of the distribution of H_2S emissions in order to achieve a better local fit to our understanding of the data.

Figure 5 shows a profile of the overall costs of the linear and logarithmic control approaches. The figure of merit being graphed here is the sum of the cost of the iron chelate and an assessment of $70 for each hour where the simulated emissions were above the regulatory ceiling. The numbers in parentheses show the overall costs of the two approaches. In this example, the linear controller's costs were slightly lower, despite its high costs in the hour after the shock.

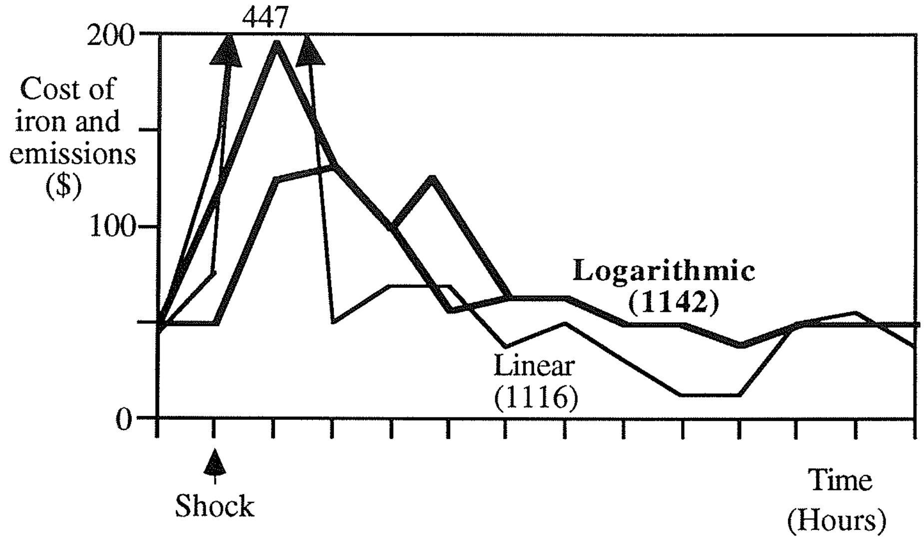


Figure 5. Overall cost profile of the two controllers

While our adjustment of the value function to reflect "green" sentiments was informal, it could be made more formal if it made a significant difference, and sensitivity analysis can establish the impact of this representation of preferences on the system and bound its effect. It would be impossible even to address value tradeoff questions in traditional PID process controllers, which make no explicit reference to an objective function. However, we were not entirely satisfied with the ability of GID's quadratic value function to capture the apparent discontinuity in our decision-maker's value tradeoff preferences at the regulatory ceiling. It may be that extensions to GIDs which allow the value function to be the exponential of a quadratic (described in Shachter and Kenley 1989, but not yet implemented in IRMA) will give enough flexibility to specify value tradeoffs robustly. We believe that both controllers would do a good job, but online tests are required before this can be stated with confidence.

3.4. Thermal inefficiency diagnostic tree

This subsection describes the component of our system that identifies the causes of thermal inefficiency at the Geysers plant. The domain expert in this area (Mr. Shaun Brady) described his problem solving thought processes in detail, and many aspects of that thought can be captured very faithfully in the logic of a diagnostic tree, so we employed that formalism for thermal cycle trouble-shooting. We describe the structure and parameters of this problem-solving tool, we illustrate its use, and we describe its effectiveness diagnosing sources of inefficiency in our plant simulation.

3.4.1. Structure and Parameters

Mr. Brady told us that his troubleshooting logic follows the diagnostic tree shown in figure 6, where incoming arrows indicate factors which could cause the parameter in question to take an inappropriate value. He diagnoses inefficiency beginning at the root and stepping through the tree investigating possible sources of problems at each juncture. For example, suppose that efficiency is impaired due to a malfunction of the motive steam system. He begins at the root by investigating whether sufficient power is generated. If not, he investigates main steam flow, backpressure, and turbine effectiveness. Presumably backpressure would be too high, so he then investigates TTD (a measure of condenser effectiveness) and cooling water temperature. Finding TTD also to be too high, he is led to investigate, among other things, the motive steam system.

We built a diagnostic component for our system that mimics this logic, using specific rejection values to determine the appropriateness of the following parameters: backpressure, TTD, vent gas flow, condenser air flow (= air leak), cooling water temperature, tower effectiveness, steam flow, and steam rate. The following problems were modeled as Booleans: dirty condenser, insufficient motive steam, auxiliary condenser problem, recirculation of steam through the tower, bad distribution of water to cooling tower cells, tower fill is clogged, and steam valves closed. The following parameters are indicted if other explanations of inadequacies failed, insofar as they are uncontrollable or not easily measured: excess noncondensibles (for condenser loading), unknown condenser problem (for TTD), wet bulb temperature too high (for CWT), and low incoming steam flow; in this way, any situation which "falls through the cracks" of the diagnostic tree's expertise is explicitly flagged, and as specific a description of the situation as possible is given. Backpressure and Power generated are logically determined by their antecedents.

The following rejection points were used to determine the appropriateness of each parameter: backpressure > 4.75, TTD > 18.5, vent gas flow > 7, condenser air flow (= air leak) > 3, cooling water temperature > 81°, tower effectiveness < 0.85, steam flow < 650, and steam rate > 20.

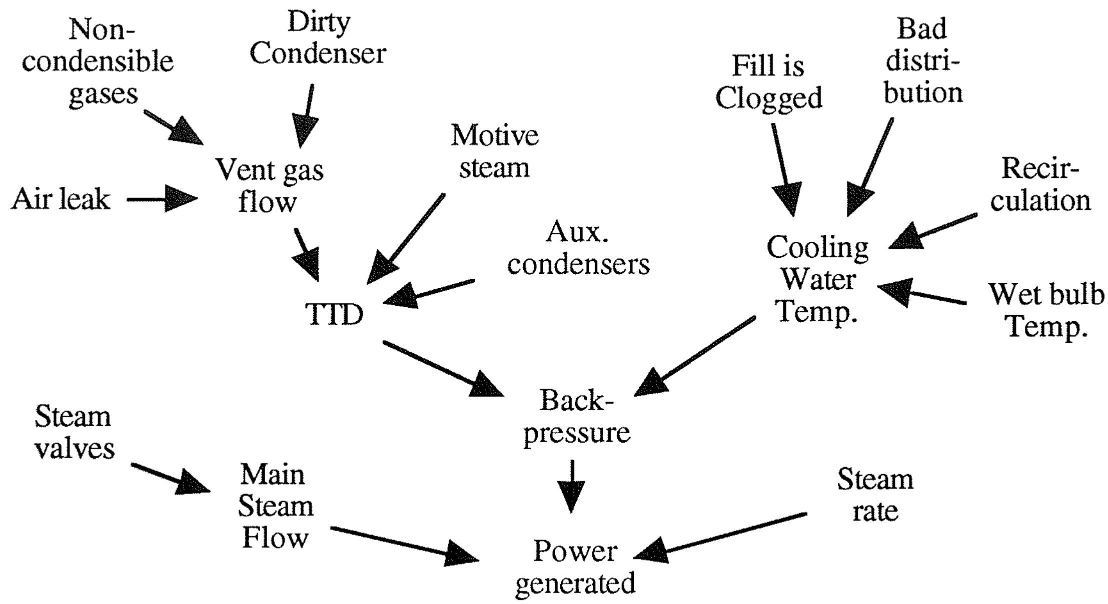


Figure 6. A tree for diagnosing causes of thermal inefficiency

3.4.2. Use

There are two issues of concern in the use of a diagnostic tree: how to find out the value of the parameter in question, and how to judge whether it is acceptable. For the Geysers system, most of the investigations of parameters called for here would have to be carried out by system operators or engineers and input into the system manually, as these quantities are not available online. Having made this commitment to human involvement, we then made all the simulation's parameters available to the diagnostic system, tracked the amount of human investigation that was called for by the system, and tried to design the diagnostic tree solver to minimize inconvenience to personnel. To track the human investigation required, and to assure ourselves that the diagnostician was working properly, we gave it the ability to give a verbal trace of the conclusions it reached and logic and crucial data it considered in the process. An example is given in an appendix (section 9).

3.4.3. Performance

Two types of judgment errors can be made at any juncture: deciding that a parameter is unacceptable when nothing is wrong (a "false positive") and deciding that it is acceptable when it is really giving evidence of an underlying problem (a "false negative"). A false positive is viewed as more costly, because it leads to time-consuming unnecessary remedial action, whereas a false negative may well be rendered moot by the problem becoming evident in the near future.

The approach to judging whether a parameter is acceptable, as described to us by Mr. Brady, uses a fixed rejection point to determine whether each parameter is at an appropriate level, but it is clear to us that he makes more refined judgments than this in practice, because these thresholds are routinely violated without causing him alarm. The thresholds are defined for an optimal system, and he makes allowances for environmental conditions, noise in the system, and the effect of known faults in the system. In order to reduce false positives, we made the thresholds more lenient. This required us to allow the operator to specify to the diagnostic system that a variable has an unacceptable value, even if it was within its rejection point due to favorable conditions and a lenient criterion. These judgments normally concern the acceptability of parameters near the root of the tree such as backpressure, whose appropriate values the operators know well; the system's diagnostics become more powerful and require less operator assistance as more problem-specific parameters are checked.

In our test suite of 21 exogenous shocks to the underlying simulation, the thermal diagnostician typically diagnoses the problem unaided a few times, it diagnoses the problem with the operator specifying that backpressure is too high in all but a couple cases, and it is able to identify the problem with one more "assist", regarding either TTD or cooling water temperature as appropriate, in all the remaining cases. An example of this test suite is given in an appendix.

We believe our system tracks the qualitative logic of the Geysers thermal expert's reasoning reasonably well. A trace of the logic printed by the thermal diagnostic system when diagnosing a dirty condenser is given in an appendix. But while this system works adequately, we would like to do better. We would like for its judgments of acceptability of a parameter to be made taking account of environmental conditions (e.g. wet bulb) and known faults (e.g. minor air leaks that are judged not to be worth looking for and remedying). In addition, we would like for the reasoning to take account of noise in the system by retracting a judgment which was "very close" when subsequent contrary evidence arises. And finally, we would like to be able to explicitly model the possibility that a parameter reading is entirely inappropriate due to a broken sensor.

3.5. Supervisory control and Schemer

We describe our implementation of Schemer, the support it gives for problem solving, and the specific IRMA framework for supervisory control built on top of Schemer.

3.5.1. Our implementation of Schemer

3.5.1.1. LIS Schemer overview

This description of Schemer given in section 2 may seem a bit abstract. Our current implementation of Schemer is written in Lucid Common Lisp with Common Lisp Object

System on a Dec workstation. For concreteness we include here the instructions to coders from the documentation of LIS schemer.

> Here is a simplified picture of schemer: there is one schemer object; it has multiple handlers; each handler has multiple ports; handlers may connect each of their ports to one or more ports of other handlers and communicate with other handlers through the connected ports. In this implementation, there is a CLOS object for each schemer, handler, and port. Each schemer and handler has a process, which is initially dormant, and a function, which is what it does once it is activated. You breathe life into the schemer (you give it the entire CPU), and it then divides up its time among its handlers. You can specify the size and frequency of the time slice that each handler will be given to exercise explicit control of this important problem-solving resource. If a handler initiates synchronous communication, its intended communication is noted in the appropriate port data structure, and it hangs, consuming no CPU resource, until the communication is completed (i.e., the schemer skips its time slice in its rotation). I have written a function for the schemer, which you may use if you wish, or you may use your own. Mine is called `'time-slice`. Normally the function a handler executes is an infinite loop waiting for synchronous inputs and responding to them. You write your own handler functions. To make a handler communicate through a port, you put a call to one of the port "methods" (a method is a function that takes objects as arguments) in the handler function.

3.5.1.2. Interprocess communication

Our interprocess communication infrastructure is based on that set out in (Reid 1980). Reid intends to prove a result about the duality of communication and control, so her concern is to guaranteed the widest expressiveness of her representation of communication, with particular emphasis on dynamic aspects of communication. Accordingly, she allows dynamic creation and destruction of components (which we call handlers in Schemer), communication ports, and connections among ports. Information may be communicated from one component to another by way of connected ports. She requires the direction (inbound, or outbound) and synchronicity (synchronous or asynchronous) of ports to be specified when they are created, and requires compatibility along these dimensions for a pair of ports to be connected. A component may connect any pair of ports whose identities it knows. In synchronous communication, the read and the write complete together, with one or the other waiting, if necessary. Asynchronous communication may fail, but it does not suspend the processing of either component. Reid allows a port to be connected to multiple ports for communication.

Reid identifies a set of design-time commitments and shows that any choice among these commitments gives equivalent expressiveness to any other. For synchronous communication, the two commitments are to what we call multiplicity and degree of synchrony. The multiplicity of synchronous communication is relevant only when a port is multiply-connected. It may either be disjunctive, in which case exactly one of the connected ports takes part in the communication, or conjunctive, in which all connected ports take part. Reid discusses tightly and loosely coupled synchronous communication, which we will characterize later as a discussion of the degree of synchrony. The difference is that in tightly coupled synchronous communication the send and receive always terminate simultaneously, whereas with loosely coupled synchronous communication, if the send is initiated first, the message is queued and the sending component is free to proceed. Reid also identifies two design dimensions for asynchronous communication: what we call locality, and one dimension of destructiveness of communication. For asynchronous communication, she distinguishes transmitted communication, which is lost if the intended recipient component, port or link is created after the message is sent, and retained communication, which is lost if the sending component, port or link is destroyed before the message is received. Her second distinction for asynchronous communication is whether reading a message destroys it. She assumes that writing a message does not destroy any other message.

LIS Schemer implements Reid's semantics, with the following modifications. We implemented retained writes conjunctively only; and reads of both localities disjunctively only. We view these restrictions as being motivated by the semantics of the communication, and we have never had a use for the sorts of communication excluded. We support mixed multiplicity of transmitted communication, so that a message from one multiply-connected port to another may be sent to all recipients without obliging the recipient to receive from all senders. We require that all ports be typed by locality (transmitted or retained). This typing determines the destructiveness of subsequent communication acts. All transmitted communication is read destructively but not overwritten; all retained communication is read nondestructively but is overwritten by subsequent messages. As before, we feel these restrictions are motivated by the semantics of the communication types, and we have had no need for the types excluded. This typing by locality supports mixed synchrony. To understand mixed synchrony, note that to synchronize communication, the read must wait until a message is available, and the write must wait until the message is read. We characterize each half of this communication as having level one synchrony, and each half of strictly asynchronous communication as having level zero synchrony. Loosely coupled synchronous communication is one form of mixed synchrony, where the read has level one synchrony (it will not complete until a message is received), but the write has level zero synchrony (it will complete and proceed regardless). Mixed synchrony supports strict synchronous, strict asynchronous, loosely coupled synchronous communication, and also the converse case where the write is at level one (it will not complete until someone has read the message), but the read is at level zero. Higher levels of synchrony are required to synchronize conjunctive communication of multiple components. At present LIS Schemer supports only levels zero and one synchrony; higher levels of synchrony can be emulated using multiple communications at level one synchrony. LIS Schemer provides a naming service for handlers to support appropriate connections of ports.

3.5.1.3. Multiprocessing

LIS Schemer is implemented on a single-processor DEC workstation. Accordingly its multiprocessing is accomplished by time-slicing among the handlers. The general architecture supports integration of handlers running on multiple processors, but our implementation does not yet do so. As implied above, LIS Schemer supports dynamic creation and destruction of handlers. It also allows the length or frequency of their time slices to be dynamically revised in response to changing environmental requirements. Our implementation supports embedded Schemers – a handler may either be a primitive unit (a lisp function), or it may be a Schemer itself, time-slicing among component handlers and generating behavior as an aggregate of its components' behaviors. LIS Schemer supports indefinite deferral of a handler. It supports interruptibility within the granularity of our implementation (a few seconds). We are investigating ways to achieve a finer time granularity, to allow more speedy interruption when required. And finally, LIS Schemer supports a non-busy wait when a component is hung for i/o from a port. This feature, together with the discipline of writing handler functions in an infinite loop headed by a synchronous read, allows efficient and responsive use of the processor.

3.5.1.4. Triggering

Synchronous receipt of informational commands and requests by Schemer handlers may be likened to triggering of a knowledge source in a blackboard system (Hayes-Roth, 1985; Nii, 1986). Informational commands either request a report of the handler's data state or cause a change in it, while requests normally cause a handler to perform some action. The quiescent state of the handler awaiting synchronous communication corresponds to the state of a blackboard knowledge source which has not yet been triggered; informational commands that change a handler's data state are like "partial matches" that change the likelihood of subsequent triggering or the nature of the subsequent response; and requests that draw a response correspond to actual triggering.

3.5.2. Schemer problem-solving support

Schemer is a development environment and a substantive problem solving approach as well as a formal architecture. Schemer resource management runtime architecture encompasses all PS approaches, providing infrastructure needed by all, and suggesting or favoring none. LIS Schemer's "utility handlers" and other infrastructure support system development and give substantive guidance toward specific PS approaches.

The support for development provided by LIS Schemer includes the following: routines that simplify the maintenance of ports and links through the entire life cycle of a port (create and register port; ask the schemer to identify other ports; connect it to them; utilize, unregister, and destroy it); a dialog box for loading source code files, which, together with the port-linking discipline supported by these routines allows the developer to swap out a running handler, read in a new definition, and swap in an improved instance of the handler which picks up where it left off; a handler template that supports development of new handlers; a handler that supports query and manipulation of ports throughout the system; a function that returns a pointer to a given handler for manipulation of its priority or other resources; a handler that periodically reports the status of each handler, as reported in their "status ports"; a handler that purges crashed handlers and cleans up associated data structures; and an execution history and handler-level runtime profiler. We expect to be able to develop handler-level step and trace functions as an additional development aid based on the checkpoints identified for the execution history.

Utility handlers that can support a variety of applications include: a handler which will send a "wake-up communication" to another handler at a pre-specified time, to support periodic execution of a function without an intervening busy-wait; a "blackboard" handler which supports a publicly-readable and -writeable data structure for other handlers; a handler that supports asynchronous processing of keyboard and mouse events for X Window dialog boxes; a handler that re-generates other handlers as needed; and a handler that inputs user commands and dispatches them to the appropriate destination.

3.5.3. IRMA

In order to illustrate the role of ports, we give some typical uses of different types of ports. We use the word synchronous here to mean having synchrony of level one. Unless explicitly noted to the contrary, communication discussed in this paragraph is transmitted and disjunctive. Conjunctive synchronous writes initiate both instances of the simulation and make sure that the failure of this to occur is made evident by the writing party being hung up. Synchronous reads are heavily used by service handlers to be always available when needed, and yet not to use any CPU resources. They are also used to receive permission to use a non-shareable resource, suspending handler operations until the resource is available; disjunctive writes of either level of synchrony may be employed to give such permission. These writes, and instructions for activities whose results are not required immediately are normally performed asynchronously so that if the service is not performed, the requester is able to undertake remedial activity. Asynchronous reads are used in unusual cases where a handler's primary mission is not to respond to requests/commands, but when it nonetheless should be able to respond to them. Examples include the wake-up handler that never sleeps (= hangs for input) in order to allow its clients to sleep until a specified time, and the master-control handler, one of whose modes is to continually re-initiate simulation, diagnosis, and control to demonstrate the components of the system in action. Retained ports are useful when there are aspects of a handler's state that it wishes to make available to others without concerning itself about when it will be accessed. The most common example is for handlers to post their "status" in a retained port for the status-report handler to read and report on. Retained writes are also helpful for promulgating access to a shareable resource (e.g. a pointer to a shareable X Windows display). In addition, the blackboard handler receives new information synchronously and displays it on a retained port, so that the blackboard is always readable by other handlers.

Here is a list of the handlers specific to the Geysers implementation of IRMA, and their functions.

- The dispatcher reads user input from a dialog box and dispatches it to the appropriate recipient.
- The handler-creator creates a new instance of any handler on command.
- The scripter reads a script of commands from a disk file that specifies exogenous conditions to simulate, instructions to control simulation, or instructions to modify iron control directives or aid diagnoses. This supports a test suite of exogenous shocks to the system that would take hours of attention if controlled manually.
- The master control handler implements technical changes that optimize the system for consideration of thermal or chemical components or for "production" operation of all components optimally. It supports simulation of a single time period, continuous simulation, or strictly synchronous responses to a test suite of input conditions and commands.
- The two instances of the simulation handler (for units 5 and 6 respectively) will simulate an hour on command, will employ either simple or GID control for iron

feed rate on command, and will allow query or manipulation of any state variable in the simulation during simulation or stasis.

- Two instances of the GID chemical controller recommend an iron feed rate based on GID analysis and send graphics manipulation commands to the graphics manager to document their progress, if requested. They will toggle between linear or logarithmic formulations on command.
- Two instances of the simple chemical controller recommend an iron feed rate quickly but nonoptimally, for use when focusing on the thermal diagnostician.
- The graphics manager displays an influence diagram or diagnostic tree, accepts commands for manipulation of components of these images, and suppresses or allows graphical output on command.
- There are two text-window managers that manage sections of an X Windows display, reporting system status of the simulation and controller respectively.
- There are two report writers that write a report of the state of the simulation to disk, and which also annotate this report with user's comments on command.

4. Evaluation of the research

We treat two major topics in this evaluation of our research: the role of Schemer as a basis for the IRMA prototype, and the success of IRMA's supervisory control paradigm in this application. We also evaluate our work in light of other research goals that motivated it.

4.1. Schemer

We found the Schemer multiprocessing architecture to be an important part of the problem-solving (PS) system because 1) it allows the PS tools described above (both symbolic and numeric) to be run and coordinated without constraining the timing of one control process to serve the other, 2) it allows for background processes that maintain system integrity and provide robustness in the face of exceptional situations, 3) it supports interruptibility, for cases where an important problem must be dealt with quickly and single-mindedly, 4) its support for inter-process communication allows PS routines to poll for their data, even when a real-time data capture system is designed for interrupt-oriented communication, 5) it enables background PS processes, such as re-estimation of model parameters, and 6) it allows human operators to initiate, redirect, or advise ongoing PS processes, e.g. to override control recommendations. We found Schemer's flexible support of control structures particularly helpful. We have been able to run the entire Geysers application in "field" mode, with all components running asynchronously, in "demo" mode, where all activities are sequenced for clear presentation in a demonstration run for PG&E personnel, and in "verification" mode, where a test suite of activities and natural conditions is fed in from an input file.

Schemer's control of CPU and communication connectivity give the control application a way to explicitly manage these computational resources, which has added clarity to the problem-solving process. It would be beneficial if Schemer could also manage RAM, another important computing resource, and if the relationship of its resource management paradigm to a rational or economic framework were more evident. Notwithstanding this failing, Schemer can clearly be seen to support the vision of rational community problem solving set out at the end of section 2.3.

4.2. Supervisory control

The concept of supervisory control proved quite appropriate for the task. We were able to run the plant simulation, the two major control processes, and various support processes asynchronously (as they would be in an online situation), and we found that we achieved the anticipated benefits. The operator or analyst could interact with or advise any individual component without interfering with the others. Although we did not describe it here, we constructed another iron feed rate controller that operated very quickly but crudely, and we were able to swap it in and out as circumstances dictated; this ability can be as valuable at the plant as it was for our demonstration project. The diagnostic component was designed according to a supervisory model – that there is an ongoing monitoring process (in the operator's head) that invokes it as needed. We have not been able to investigate automation of this process in our research so far, but the architecture of the system supports it cleanly, and this is an area of long term interest to us. The chemical controller was able to use simple GID solution components to solve a dynamic problem with dynamic parameters by means of a relatively simple addition of a module to handle time-dependent priors. The benefits from this decoupled and reflective approach to control were evident even in this environment where the overall structure of the problem is not changing much and where time requirements are relatively forgiving; the supervisory control approach seems even more attractive for domains such as modern assembly lines which must be flexible regarding both quantity of output and type of product. IRMA's approach seems equally suited to support other aspects of process management such as fault detection, failure analysis, and fault correction, and scheduling and rescheduling of regular maintenance in a production facility.

4.3. Other aspects

We feel that the diagnostic tree processor that could be extracted from the thermal diagnostician and the GID solver that forms the basis of the iron feed rate controller constitute an important basis for a toolbox for a supervisory controller. Other tools that could well prove useful include an LP solver, a database manager, a truth-maintenance system, and a MoGID solver (as discussed in section 5). We found GIDs to be very useful in forming inferences about the underlying state of a very noisy (simulated) system, and we found that they could do so robustly even if data were missing at a decision point. We were less satisfied with the ability to express value tradeoffs in the implementation of GID we were using. Only the tradeoffs implicit in a quadratic objective function could be captured. The diagnostic tree was found to support a clearly auditable analysis of the source of thermal efficiencies, and it showed a basic competence, but the use of fixed rejection values that are insensitive to contextual features was found to be unduly restrictive. Our setpoint-spread-sensitivity knowledge representation, while not yet fully formalized and automated, constitutes the basis of an important bridge between the way experts think about the domain and the kinds of information needed by solvers. We also note that a tremendous amount of effort went into constructing the GID for control, but that, in retrospect, much of the effort could be formalized into a system that dynamically constructs such models. In addition, we identified only a few problem-specific heuristics for choosing the appropriate solver for a given problem.

5. Future research

Here we describe some future research that is motivated by these reflections.

5.1. Mixture-of-Gaussian IDs

We are investigating a generalization of GIDs that allows both continuous and discrete nodes. The conditional distributions in these nodes are mixtures (weighted sums) of Gaussian components. In these mixture-of-Gaussians IDs (MoGIDs), we allow continuous-valued nodes to be conditioned by discrete nodes: for each discrete condition of the parent, the child may have a different conditional Gaussian distribution. Generalization of the GID solution algorithm for MoGIDs with discrete decision and chance variables appears feasible as long as continuous variables are prevented from conditioning discrete ones. MoGIDs can express anything that can be expressed in a discrete ID (by setting all component Gaussians' variances to zero) or a GID (by giving every node only one Gaussian component). In return for their increased complexity, MoGIDs give us the ability to reason about discrete conditioning events, such as component malfunction.

This may provide a more powerful, integrated approach to some of IRMA's tasks such as thermal control than is afforded by the fault-tree methods currently used. In the diagnostic tree described in section 3.3, the underlying premise is that each node represents a physical subsystem whose operation is in one of two states, fixed or broken. The use of fixed setpoints and the inability to revisit conclusions drawn earlier in a traversal of the tree reflects the assumption that the system is roughly deterministic. We have found this assumption to be problematic. We feel that the thermal system could be modeled better by a MoGID with discrete variables representing whether the physical devices or sensors in question are fixed or broken with different Gaussian distributions for the output of each device in each of these cases. For instance, a working sensor will reflect the true value of the quantity it measures with some variance, whereas it will read "zero" with no variance if it is broken; for modeling cooling water temperature, there could be one Gaussian distribution for it under normal conditions, and another less favorable one if tower steam is recirculating (see figure 7 for a generic ID fragment).

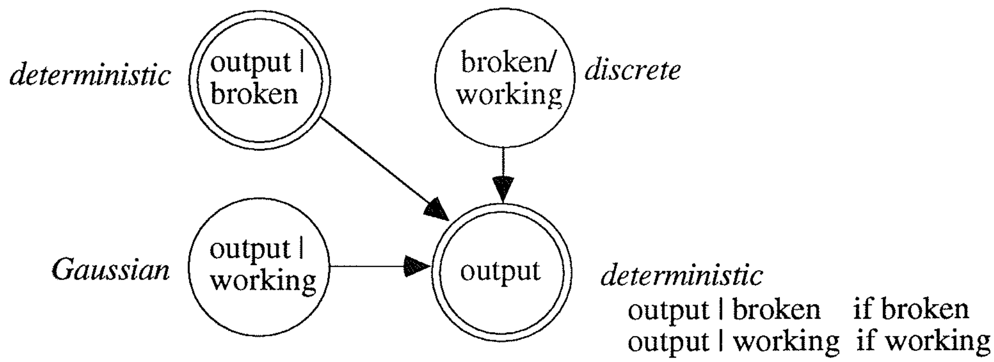


Figure 7. A mixture of Gaussians model of device output

If we take the graphical structure of the diagnostic tree and make each of its junctures a deterministic node with only two states (broken or fixed) in a discrete ID, we can capture the essential logic of the tree. If we then introduce noise into our output model in one or both of these states and reflect quantitative relationships among nodes' responses, we have a MoGID model of the system that can absorb evidence of environmental conditions before reaching inferences about the state of a component. For example, it would be straightforward to build such a model that "knows" that backpressure of 4", while perfectly acceptable when ambient temperature is 60°, is excessive when ambient temperature is 35°. We believe this approach can cut down "false negative" diagnoses and handle multiple mishaps.

5.2. Dynamic Model Construction

While our approach to control is oriented toward the use of multiple diverse problem solving components, and while we started our research with a number of solvers in mind, we find ourselves pressed toward using roughly the same technique (IDs with continuous variables) for both problems we studied carefully. This may be explained by considering three important aspects of problem difficulty: complexity, uncertainty, and time pressure.[10] In both issues we addressed in the Geysers application there is some need to handle complexity, uncertainty is important, and time pressure is weak, so we have been led to move from quick deterministic analytic techniques such as a diagnostic tree to more expressive solution techniques such as MoGID. If time pressures were greater and the system more nearly deterministic, the diagnostic tree would be more appropriate, or if the system were more complex but less uncertain, a system of equations solver might be more appropriate. Figure 8 shows a taxonomy of solution techniques, where increased expressiveness in either dimension generally requires increased runtime.

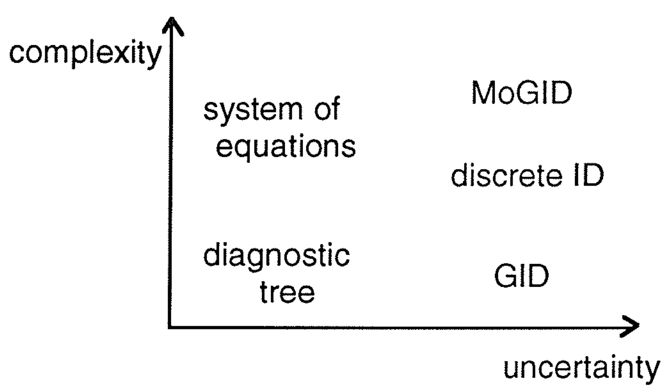


Figure 8. Taxonomy of solvers

We believe that automatic model construction offers the possibility of control systems based on models being created, maintained, and upgraded by domain experts without experience in modeling. By suitable extensions to the hypertext knowledge-engineering tool described in section 3.1, we hope to research creation of a graphical interface for system description that can be used by end users, and that can support automated construction of different sorts of models for different needs, e.g., end users could add or modify branches in the diagnostic tree as conditions change. This will require further research into knowledge representation, choice of solver, and construction of a solution

[10]These aspects of problem difficulty are motivated by the problem dimensions given in (Howard, 1968): complexity, uncertainty, and time dependence; and the dimensions of computer program costs in (Schoppers and Linden, 1990): response time, processing power required, program space, and inattentiveness to problem detail.

basis from a general-purpose database. An important next step on this research path will be to generalize the model construction capabilities in Geysers IRMA as follows: 1) make both the thermal diagnostician and chemical controllers completely data-driven by identifying and isolating data about system features being addressed, 2) merge the two resulting databases into one by identifying or creating a formalism that supports them both, 3) fill in the data frames in this database consistently, 4) make an ad hoc model constructor that construct appropriate solvers (e.g. the ones we constructed by hand) in routine situations, and 5) generalize the logic of this constructor with regard to choice of solver, granularity of model, and discretization and unrolling of time.

5.3. Other possible extensions

Other possible extensions to the research include symbolically compiling GID inference to avoid repeated arc reversals, employing Kjaerulff's triangulation procedure (Kjaerulff 1992) to speed ID inference, developing faster, GID-specific solution algorithms, constructing and performing a full-fledged decision analysis for recommendations of expensive remedies to thermal inefficiencies, and automatic re-estimation of chemical control system parameters from online data.

While Schemer's control of CPU and communication connectivity is helpful, it would be beneficial for Schemer to control RAM as well. In addition, it would be helpful for the control of these computational resources to be cast in explicitly economic terms (giving computational entities funny money and making them buy the resources they need). This approach was set out in (Miller and Drexler, 1988) and (Drexler and Miller, 1988) and a preliminary implementation was given in (Waldspurger, et al., 1992). We feel that this is an important research direction for Schemer research.

6. Relationship to other work

Tatman and Shachter (1990) describe an efficient computational procedure for solving dynamic programming problems (problems with repeated decisions) under uncertainty in influence diagrams. This work uses of the notion of a set of nodes summarizing history (a Markov state). It introduces a "sub-value node" that isolates each period's contribution to value, allowing the subproblem for each time period to be solved independently of other periods. Although our chemical control problem has a repeated decision structure, we felt the effect of future decisions in our problem could be captured in a simpler way than this.

Dean et al. (1990), Kirman et al. (1991), and Paterson and Fehling (1992) employ a similar approach to ours in their work on Bayesian control systems for robotic activity and information gathering. The latter, whose domain is the activity of an unmanned Mars rover, describes the use of GIDs for operational decisions with particular reference to specification of a quadratic value function. This article explicitly addresses the decision whether the rover should act or bring humans at mission control "into the loop". A similar concept to this "mixed initiative" behavior of robots is the notion of an "active" Decision Support System, where the DSS performs analyses of its own and reports these to the analyst as well as serving as a workbench for her/him. This notion was presaged by the empirical investigations of Shoval (1986). Implementations are described in various papers (e.g., McCoy and Boys, 1987; Mili, 1989; Manheim et al., 1991; Raghavan, 1991; and others).

D'Ambrosio and Fehling (D'Ambrosio et al., 1987) describe the MCM controller for continuous production of phosphorus that performs inference in a qualitative system model whose variables were in three broad classes: observations, states, and high level syndromes, following the CASNET model (Weiss et al., 1978). The modeling in IRMA supports, but does not require, such use of multiple classificatory layers. However, qualitative modeling is currently not supported. Both MCM and IRMA employ the Schemer real-time, multitasking architecture to manage supervisory control activities.

Ramamurthi and Agogino also use influence diagrams for process control (Ramamurthi and Agogino, 1988; Ramamurthi et al., 1990). There are three main differences between their work and ours. First, influence diagrams in their system are discrete, not continuous. Second, their use of arc reversal for probabilistic inference is off-line, which decreases runtime but does so at the cost of limiting competence to the cases for which a particular arrangement of arcs is appropriate. And finally, their only control method is the influence diagram, whereas IRMA uses the GID as one tool in a toolbox.

7. Summary

IRMA presents a promising approach to supervisory process control. Its employs decision analytic tools that are oriented toward rational control decisions, and other solution methodologies that offer different thoroughness/runtime tradeoffs, to allow a supervisory function to make rational choices among them. In this implementation, the supervisory function is left to the human operator, but the IRMA architecture allows it to be automated, and the Schemer environment on which it is built encourages this sort of reflective meta-control. The IRMA architecture is based on the Schemer multiprocessing architecture for distributed, real-time problem-solving applications. Our development of Schemer for this project has focused on flexibility in the interprocess communication infrastructure, which has supported a variety of efficient and clear ways to coordinate heterogeneous problem solving technologies. In particular, we find that Schemer architecture gives significant support to the supervisory control approach. IRMA takes an episodic problem solving approach to supervisory control: discretizing the time dimension of a continuous process allows us to formulate a control problem at a geothermal plant as a repeated decision problem. IRMA integrates diverse tools into a system that can use different problem-solving approaches, such as Gaussian influence diagrams (GIDs) and diagnostic trees, as dictated by the nature of the problem and the problem solving resources (time, data, solvers) available. IRMA can accept advice or redirection from the operator while processing. Acquisition of a body of knowledge about plant operations, and synthesis of this knowledge into a creditable computer model of two PG&E Geysers geothermal plants allowed us to demonstrate the value of the IRMA supervisory control approach for this domain. The GID-based controller in our prototype gives normatively defensible emissions control directives using limited information. Using GIDs, we can formulate the control problem in terms accessible to domain experts and solve it in a normatively defensible way. The GID approach integrates weather, emissions, and steam flow data, and filters out measurement and process noise; and it balances H_2S emissions against financial cost. We found that use of variables transformed to log space is helpful in modeling chemical reactions in a GID. The thermal diagnostician in our prototype was able to find most sources of thermal inefficiency in our simulation using a diagnostic tree and information that is available online in the Geysers generating units. This includes data regarding weather, chemical makeup of the circulating water, and thermal properties of the steam. In our work, we implemented and employed causal network for knowledge representation. The setpoint / spread / sensitivity knowledge framework upon which it is based appears to be both accessible to domain experts and useful for diverse solvers and automatic modeling.

Our ongoing research addresses research problems that we were able to state as a result of this IRMA prototype for PG&E Geysers geothermal plants. Thermal diagnostics may be better done with an influence diagram employing mixtures of Gaussians. We are building a MoGID solver. We are investigating ways to automatically construct different kinds of problem solving systems from a single knowledge base. We are pursuing a more explicit use of economic concepts for computational resource allocation. And finally, another of our long-term research objectives is to determine how a system like IRMA can best function on the spectrum from full automation to providing advice and guidance to human process managers.

8. References

Chao, H.P., P.A.Morris, B.R.Judd, and S.C.Peck, 1985, “Analysing complex decisions for electric utilities”, *Long Range Planning* v 18 n 2 Apr 1985 p 46-55.

D'Ambrosio, Bruce, Michael Fehling, Stephanie Forrest, Peter Raulefs, and B.Michael Wilber, 1987, "Real-time process management for materials composition in chemical manufacturing", *IEEE Expert*, summer 1987, p.80-93.

Dean, T., K.Basye, R.Chekaluk, S.Hyun, M.Lejter, M.Randazza, 1990, "Coping with uncertainty in a control system for navigation and exploration", in AAAI-90 Proceedings. Eighth National Conference on Artificial Intelligence, Cambridge, MA, USA: MIT Press. p. 1010-15.

Drexler, E., M.Miller, 1988, "Incentive engineering for computational resource management", in The Ecology of Computation, ed. B.A.Huberman, North-Holland, Amsterdam.

Fehling, Michael R., Art M.Altman, and B.Michael Wilber, 1989, "The Heuristic Control Virtual Machine: an implementation of the schemer computational model of reflective, real-time problem-solving", chapter 9 of Blackboard Architectures and Applications, V.Jagannathan, R.Dodhiawala, and L.Baum, (eds.) Academic Press, Inc.

Good, I.J., 1983, Good Thinking, University of Minnesota Press, Minneapolis.

Hayes-Roth, B., 1985, "A Blackboard architecture for control", *AI Journal*, 26(3):251-321.

Howard, R.A., 1966, "Decision Analysis: applied decision theory", Proceedings of the Fourth International Conference on Operations Research, John Wiley and Sons, New York, p.55-71. Reprinted in The Principles and Applications of Decision Analysis, 1983, R.A.Howard and J.E.Matheson, p.95-113.

________, 1968, "The Foundations of decision analysis", IEEE Trans. Sys. Sci. Cybern. SSC-4(3):211-219. Reprinted in The Principles and Applications of Decision Analysis, eds. R.A.Howard and J.E.Matheson, p.579-589.

Howard, R.A., and J.E.Matheson, 1981, "Influence diagrams", in The Principles and Applications of Decision Analysis, eds. R.A.Howard and J.E.Matheson, p.719-762.

________, 1983-4, The Principles and Applications of Decision Analysis, eds. R.A.Howard and J.E.Matheson, Strategic Decisions Group, Menlo Park CA.

Johnson, E.R., M.R.Fehling, W.B.Poland, 1992, "Continuous Decision Models for Process Control", HICSS-26(3):333-343

Keeney, Ralph L., John F.Lathrop, Alan Sicherman, 1986, "Analysis of Baltimore Gas and Electric Company's technology choice", *Operations Research* v 34 n 1 Jan-Feb 1986 p 18-39.

Kirman, J., K.Basye, T.Dean, 1991, "Sensor abstractions for control of navigation", in Proceedings, 1991 IEEE International Conference on Robotics and Automation (Cat. No.91CH2969-4). Los Alamitos, CA, USA: IEEE Comput. Soc. Press, 1991. p.2812-17.

Kjaerulff, Uffe, 1992, "A computational scheme for reasoning in dynamic probabilistic networks", *Uncertainty in Artificial Intelligence, proceedings of the eighth conference*, Morgan Kaufmann, p.121

Manheim, M.L 1988 "An architecture for Active DSS", HICSS-21(3), p.356-365.

Manheim, M.L., S.Srivastava, N.Vlahos, C-P.Tseng, 1991, "Working with an intelligent assistant: Experiments with a Symbiotic DSS for production planning and scheduling", HICSS-24(3):386.

McCoy, Michael S., Randy M.Boys, 1987, "Human performance models applied to intelligent decision support systems", IEEE Proceedings of the National Aerospace and Electronics Conference 1987. Publ by IEEE, New York, NY, USA. Available from IEEE Service Cent (Cat n 87CH2450-5), Piscataway, NJ, USA p 1352-1359.

Mili, F., 1989, "Dynamic view of decision domains for the design of active DSS", HICSS-22(3), p.24-32.

Miller, M., E.Drexler, 1988, "Markets and computation: agoric open systems", in The Ecology of Computation, ed. B.A.Huberman, North-Holland, Amsterdam.

Nii, H.Penny, 1986, "Blackboard systems", *The AI Magazine*, 7(2):38-53 and 7(3):82-106.
Paterson, Thomas S., and Michael R.Fehling, 1992, "Decision methods for adaptive task-sharing in associate systems", submitted to the 1992 National Conference on Artificial Intelligence, AAAI-92, San Jose CA, July 1992.
Raghavan, Sridhar A., 1991, "KNOB: A Mutation-based approach for generating stereotype behaviors in active decision support systems", HICSS-24(3):417.
Ramamurthi, K., and A.M.Agogino, 1988, "Real time expert system for fault tolerant supervisory control", *Computers in Engineering*, Proceedings of the 1988 ASME International Computers in Engineering Conference and Exhibition, p.333-339.
Ramamurthi, K., D.P.Shaver, and A.M.Agogino, 1990, "Real time expert system for predictive diagnostics and control of drilling operation", The Sixth Conference on AI Applications, IEEE.
Reid, Loretta, 1980, Control and Communication in Programs, UMI out-of-print books on demand, 300 N. Zeeb Rd., Ann Arbor MI 48106.
Rutledge, G., G.Thomsen, I.Beinlich, B.Farr, L.Sheiner and L.Fagan, 1989, "Combining qualitative and quantitative computation in a ventilator therapy planner", in Proceedings: The Thirteenth Annual Symposium on Computer Applications in Medical Care, p.315-319.
Schoppers, Marcel and Ted Linden, 1990, "The dimensions of knowledge based control systems and the significance of metalevels", presented at Stanford Spring Symposium, 1990.
Shachter, R.D., 1986, "Evaluating Influence Diagrams", *Operations Research*, Vol. 34, No. 6, pp. 871-882.
Shachter, R.D., D.E.Heckerman, 1987, "Thinking backward for knowledge acquisition", *AI Magazine*, 8(3):55-61.
Shachter, R.D. and C.R.Kenley, 1989, "Gaussian influence diagrams", *Management Science*, v 35 n 5, p. 527-550.
Shahidehpour, S.M. and George D.Kraft, 1986, "Applications of artificial intelligence (AI) to distributed processing in a power system environment", Proceedings of the Power Plant Dynamics, Control and Testing Symposium 6th v 1. Publ by Univ of Tennessee, Knoxville, TN, USA p 24.
Shoval, Peretz, 1986, "Comparison of decision support strategies in expert consultation systems", *International Journal of Man-Machine Studies* v 24 n 2 Feb 1986 p 125-139.
Tatman, Joseph A. and Ross D.Shachter, 1990, "Dynamic Programming and Influence Diagrams", *IEEE Trans. Syst. Man Cybern.*, vol. SMC-20, pp. 365-379, March/April 1990.
Tse, E. and M.R.Fehling, 1989, "An Intelligent Control System", in A.C.Sanderson, A.A.Desrochers, and K.Valvanis (eds.) Proceedings of the IEEE Symposium on Intelligent Control. Albany, NY, p.194-200.
Waldspurger, C.A., T.Hogg, B.A.Huberman, J.O.Kephart, W.S.Stornetta, 1992, "Spawn: a distributed computational economy", *IEEE Transactions on Software Engineering*, 18(2):103-117.
Weiss, S.M., C.A.Kulikowski, S.Amarel, A.Safir, 1978, "A model-based method for computer-aided medical decision making", *Artificial Intelligence* 11:145-172.

9. Appendices

This section contains an introduction to Gaussian Influence diagrams, the worksheet used to identify parameters of the GID-based controller, and results of various tests of the system.

9.1. Introduction to Gaussian influence diagrams

In the following subsections, we give an introduction to influence diagrams, and to a form of influence diagrams called Gaussian influence diagrams that allows specification of continuous-valued variables.

9.1.1. Influence diagrams in general

Influence diagrams (IDs) provide a general, flexible way to model decisions (Howard and Matheson, 1981). IDs explicitly represent probabilistic relevance, availability of information at decision points, and preference tradeoffs. An ID consists of a set of nodes, representing the variables, and arcs (arrows), representing probabilistic relevance or information, connecting some of the nodes. Round or oval nodes represent chance (random) variables, rectangular nodes represent decision (control) variables, and a diamond node represents the value or utility function expressing the decision maker's preferences.

A node's parents are the nodes with outgoing arcs to that node. There are two types of arcs. A conditioning arc, into a chance node, indicates that the probability distribution for the chance variable is conditioned on the values of the node's parents. An informational arc, into a decision node, indicates that the outcomes of the parents are known at the time the decision is made. Each chance node has an associated probability distribution conditioned on each possible combination of outcomes of the parent variables. If there are no parents, then the distribution is unconditional or "marginal." A deterministic node, indicated by a double oval, represents a state variable for which the *conditional* distributions are all single values occurring with certainty (the marginal distribution may show uncertainty still).

Each decision node embodies a set of alternatives. A decision policy specifies one of these alternatives for each set of outcomes of the parent variables, or a single alternative if the decision node has no parents. The criterion used to select optimal decision policies is to maximize expected utility, using the objective function represented by the value node. The maximum-expected-utility criterion for decision making is derived from axioms of utility theory as shown by von Neumann and Morgenstern, Savage, and others. A fully specified ID may be solved exactly (without numerical integration) for the optimal decision policies, if all variables have discrete values (Shachter, 1986) and in some cases with continuous variables, as illustrated below. The solution procedure applies a sequence of ID transformations that leave the joint probability distribution of the variables of interest unchanged

9.1.2. Gaussian influence diagrams

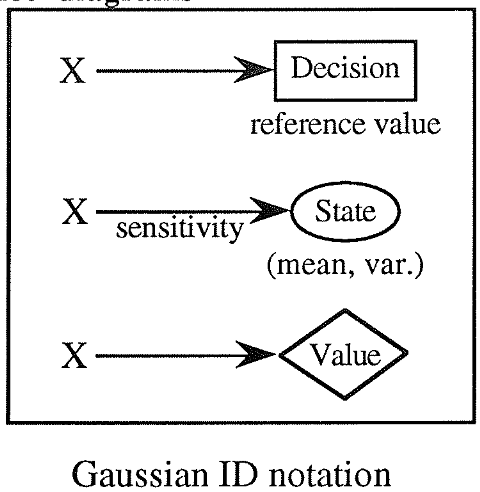


Figure 9. Gaussian ID notation

A Gaussian Influence Diagram (GID) is an ID which asserts the probability distribution of the chance variables to be multivariate Gaussian (normal), the decision domains to be continuous and unbounded, and the value function to be a quadratic function of the state variables (Shachter and Kenley, 1989). The GID can be used to represent and solve efficiently the linear-quadratic-Gaussian control model (see Kenley, 1986 for efficiency comparisons for Kalman filtering), but it is more general.

The GID may be specified in a simple, elegant way (see figure 1) as a consequence of the following properties of the multivariate Gaussian distribution: (1) the conditional distribution of any (scalar) variable given any others is Gaussian, (2) the conditional mean is a linear function of the conditioning variables, and (3) the conditional variance is constant (independent of the values of the conditioning variables). Each chance node in the GID is specified by an unconditional mean and a conditional variance. Each conditioning arc is specified by a single coefficient, giving the linear sensitivity of the mean of a node to the value of the parent indicated by the arc. This coefficient is called the strength or sensitivity of the arc. Decision nodes and the informational arcs into them do not need to be specified numerically, except for an arbitrary, initial reference value for each decision node, to allow specifying downstream nodes. The conditional distribution of a chance variable in terms of these parameters is derived below.

Suppose a chance node Y has parents $X_1, \ldots, X_n$, i.e. chance variable Y is to be conditioned on variables $X_1, \ldots, X_n$. Let μ be Y's unconditional mean. Let $b_1, \ldots, b_n$ be the coefficients of the linear conditional mean: $b_i = \partial Y(X_1, \ldots, X_n)/\partial X_i$ for $i=1, \ldots, n$. Then Y's conditional mean is:

$$E[Y \mid X_1, ..., X_n] = \mu + \sum_{i=1}^{n} b_i(X_i - EX_i),$$

so that the expectation of this conditional mean over $X_1, ..., X_n$ is the unconditional mean μ. μ is also the conditional mean gien that all of Y's parents are at their means.

Let v be the conditional variance of Y: $Var[Y|X_1, ..., X_n]$. The GID specifies node Y by μ and v and specifies the arcs to Y by $b_1, ..., b_n$. Since $Y \mid X_1, ..., X_n$ is Gaussian, it can be expressed explicitly in terms of these parameters and an additional Gaussian variable:

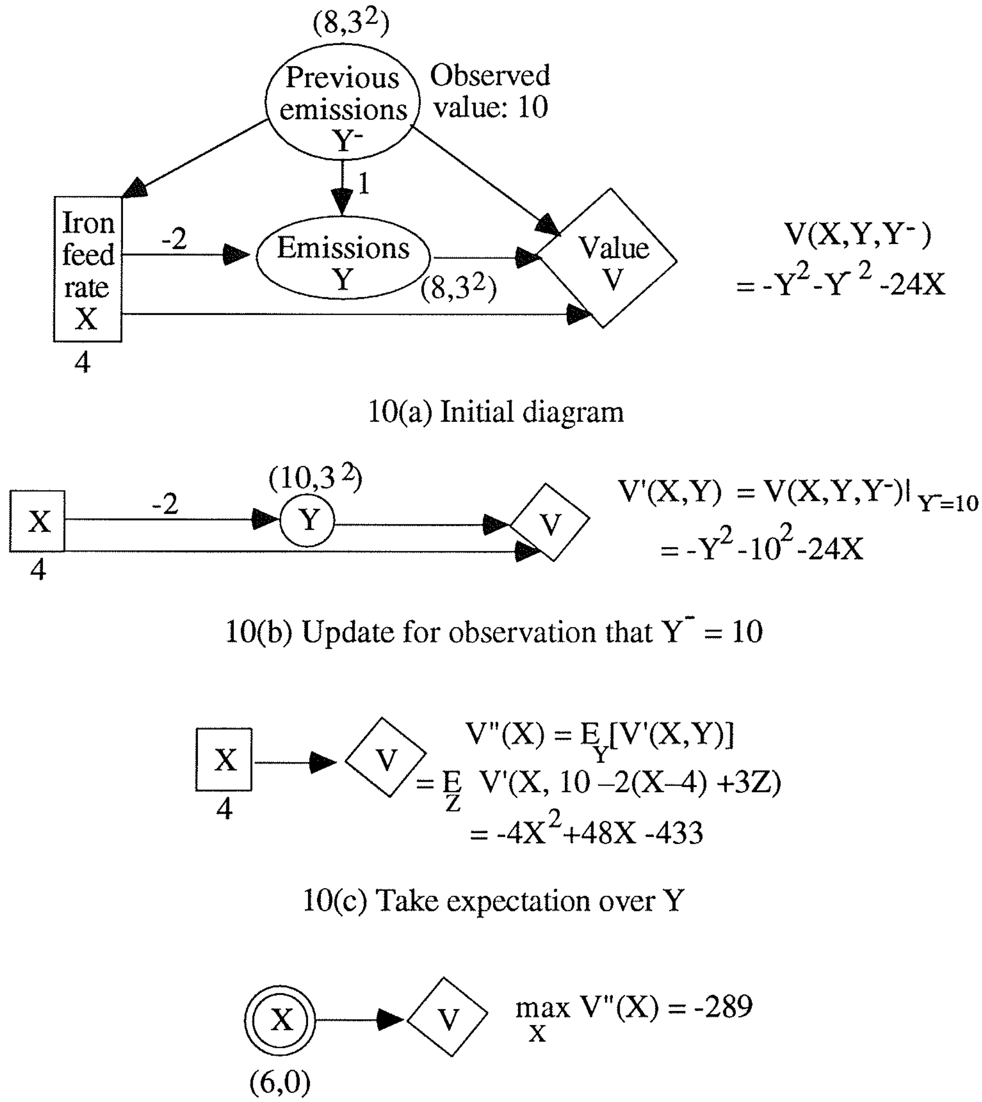
$$Y \mid X_1, ..., X_n = \mu + \sum_{i=1}^{n} b_i(X_i - EX_i) + \sqrt{v}\, Z,$$

where Z is independent standard Gaussian and is independent of all other variables. This expresses a linear regression of Y on $X_1, ..., X_n$, with error $\sqrt{v}\, Z$.

Figure 10 uses a GID to represent and solve a simple stochastic control problem. The example used is a simplified version of the subproblem faced each period for the Geysers application described in Section 3. Figure 10a shows the problem: to find the iron feed rate that minimizes the expected cost, or maximizes the negative expected value, of emissions plus iron supply costs. Value is assumed to decrease quadratically with current-period emissions (Y) and previous-period emissions (Y^-), and to decrease linearly with iron feed rate (X). The conditional mean of current emissions is assumed to increase linearly with previous emissions and decrease linearly with iron feed rate, and its conditional variance is assumed to be constant. We give the iron feed rate an initial reference value of 4, for convenience in assessing the conditional mean of emissions, which we assess to be 8 in this case. The GID model of emissions is: $Y = 8 + 1(Y^- - 8) - 2(X - 4) + 3\, Z$, where Z is standard Gaussian. We observe the previous emissions to be 10 before we make the iron feed decision.

To solve this problem, we first account for the evidence that Y^- is 10, by updating the mean of Y to $8 + 1(10 - 8) = 10$, and by updating the value function to $V'(X,Y) = V(X,Y,Y^-)|_{Y^-=10}$ (Figure 10b). Then we take the expectation of the value function over Y, giving $V''(X) = E_Y[V'(X,Y)]$. This allows us to remove Y from the diagram, since it is no longer needed to express any conditional distributions or functions (Figure 10c). Finally, we calculate the optimal decision as the maximum of the quadratic function $V''(X)$, which occurs at $X = 6$ (Figure 10d). Without the observation of high previous emissions, the optimal iron feed rate would have been only 5; however, the observation leads us to anticipate high emissions this period and to increase iron feed to 6 to abate these emissions.

Though this example is simple, it gives a qualitative picture of how our control procedure works. In more complex problems including the Geysers problem, some arcs must be reversed (via Bayes' rule), and optimal decisions must be determined as a function of their parents. Shachter and Kenley (1989) give the general, backward-recursive solution algorithm.



10(a) Initial diagram

10(b) Update for observation that $Y^- = 10$

10(c) Take expectation over Y

2(d) Optimize X

Figure 10. Example GID Solution

9.2. Derivation of parameters for linear iron feed rate GID

This worksheet shows the derivation of the parameters of the linear GID iron feed controller. The logarithmic parameters may be derived from these by the formulas given in section 3.3.3.

	unit5:unit6	
s	780	Steam flow, kph
f	4.4	Iron feed, gph (effective)
c	11:10	Iron concentration, ppm
i	390:420	H_2S in, ppm
o	8 : 6	H_2S out, ppm
a	0.0032:0.002	abatement factor
	2000 kp	water in system
	0.92	decay factor for blowdown
	0.72	decay factor for blowdown and evaporation note: $0.72(2000+780) \approx 2000$
	400	feed-to-concentration factor 10^6 ppm * 0.0004 kp/gal

$c = (0.92(2000)c+400f)(0.72(2000)+0.72s)^{-1}$
$\quad = (1840c+400f)(0.72(2000+s))^{-1}$
$o = asic^{-2}$

Solving for c in the 2d eqn gives $c = \sqrt{asi/o} = 11:9$ to make o=8, but the model keeps c at 11 : 10.

Solving for f from first eqn gives $f = 0.4\,c \approx 4$.

$\partial c/\partial c = 1840(0.72(2000+s))^{-1}\,|_{s=780} = 0.92$
$\partial c/\partial f = 400(0.72(2000+s))^{-1}\,|_{s=780} = 0.2$
$\partial c/\partial s = -(1840c+400f)(0.72(2000+s))^{-2}\,|_{c=11,f=4,s=780} \approx -0.005$
$\partial o/\partial i = asc^{-2}\,|_{a=0.0032:0.002,s=780,c=12,10} \approx 0.02$
$\partial o/\partial c = -2asic^{-3}\,|_{a=0.0032:0.002,s=780,i=390:420,c=12:10} = -1.5 : -1.26.$
but regression shows +0.2 : -1.2. I choose -0.9 : -1.2.
$\partial o/\partial s = aic^{-2}\,|_{a=0.0032:0.002,i=390:420,c=12:10} \approx 0.01$

As wet bulb temperature goes from 60 to 30 (either 50% or 100% ... avg is 71%), 187.5 less steam ((60-30)/120 %) evaporates. Linear arc wet.bulb -> amt.water is -6.

9.3. Results of test script manipulations

A test script of mishaps was prepared to test the model and the thermal diagnostician. In this script, the model is made to simulate a given mishap and then the diagnostician is told to investigate the system. The diagnoses given by the diagnostician in a typical test suite are given here.

Different test suites achieve slightly different results due to the noise in the system, but the predominant outcome is always that the bulk of the problems are diagnosed correctly. In a few case, the test suite is crafted to inform the system that the value of a general diagnostic variable is inappropriate because if this had not been done the diagnostician would likely find no problem in the system. This issue is discussed in section 3.4.3.

air leak	ok	look for air leak
aux-condenser prob.	ok (intermit)	fix aux condensers
high base steam rate	ok	turbines; cannot handle
high CWT	ok	unknown tower problem
fill problem	ok (intermit)	tower fill problem
hotwell temp too high	ok (?)	unknown tower problem
noncondensibles = 8	ok	identifies excess NCs
steam valves shut = T	ok	tells you to open the valves
TTD = 20	ok	unk. cause of condenser inefficiency
unseasonable = 15	ok	unk condenser inefficiency
water flow = 12000	ok	unk high backpressure
bad distribution = T	ok	fix tower distribution
blowdown = 500	?	unk. high backpressure
dirty condenser	ok	tell how much dirty condenser costs
insufficent motive steam	only if coaxed	identify motive steam of BP & TTD specified
recirculation	only if coaxed	identify recirc if BP, CWT
steam seal leak	ok	look for air leak/steam seal leak
throttle flow = 300	ok	unk cause of low steam
tower effectiveness = .6	ok	unk tower problem
turbine effectiveness = .6	ok	turbine: cannot fix
vent gas flow = 20	ok	vent gases high due to excess NCs
h2s emissions = 20	ok	raise feed rate from 6 to 9
iron concentration = 5	ok	raise feed rate from 9 to 13 & drop smoothly
iron content = .01	ok	raise feed from 7 to 10 & drop smoothly
iron concentration = 4	ok	emissions went way up
iron feed rate = 0	ok	it did it
main steam flow = 1100	ok	it did it
wet bulb = 30	ok	it did it

H2S emissions = 20	ok	it did it
iron content = .01	ok	it did it
main steam h2s = 600	ok	emissions went up

9.4. Results of the emissions exogenous-shock test

Since temperature, iron feed rate, steam flow and emissions are measured by online sensors at the Geysers, we made these values from the simulation available to the controller, and we report them here. For the other two crucial parameters, amt.water and iron.conc. (=amt.iron/amt.water), the actual value in the simulation is reported, followed by the value inferred by the GID controller after observing the other values.

Logarithmic control responds to the 3 ppm test: (actual/inferred)

time	temp	feed	steam	amt.water	iron.conc.	emissions
1am	41.	8.	762.	2155./2133.	14./ 13.	5.
2	39.	8.	799.	2187./2220.	3./ 9.	127.
3	38.	20.	742.	2182./2279.	6./ 9.	35.
4	43.	21.	799.	2182./2288.	9./ 10.	11.
5	46.	16.	750.	2130./2247.	10./ 12.	4.
6	49.	9.	755.	2078./2211.	11./ 11.	14.
7	54.	10.	810.	2035./2149.	12./ 12.	9.
8	54.	10.	753.	1975./2084.	12./ 12.	7.
9	54.	8.	728.	1914./2017.	13./ 13.	6.
10	56.	8.	671.	1832./1929.	13./ 14.	3.
11	54.	6.	728.	1805./1889.	13./ 13.	10.
12pm	57.	8.	785.	1791./1843.	14./ 13.	7.
1	54.	8.	842.	1835./1826.	14./ 13.	7.
2	50.	8.	836.	1895./1835.	13./ 13.	10.

Linear control responds to the 3 ppm test:

time	temp	feed	steam	amt.water	iron.conc.	emissions
4am	38.	7.	793.	2207./2222.	14./ 14.	4.
5	38.	12.	736.	2189./2387.	3./ 4.	97.
6	44.	71.	793.	2176./2370.	15./ 15.	1.
7	48.	8.	809.	2153./2346.	15./ 14.	9.
8	47.	11.	866.	2177./2330.	15./ 14.	4.
9	52.	11.	872.	2162./2286.	15./ 15.	3.
10	56.	6.	815.	2085./2219.	15./ 14.	11.
11	54.	8.	778.	2022./2149.	15./ 15.	3.
12pm	58.	5.	721.	1925./2053.	15./ 15.	3.
1	60.	2.	664.	1812./1939.	14./ 14.	4.
2	56.	2.	721.	1770./1864.	13./ 14.	1.
3	52.	6.	720.	1763./1834.	13./ 14.	8.
4	47.	9.	707.	1781./1825.	14./ 15.	1.
5	46.	6.	650.	1773./1822.	13./ 14.	6.

9.5. Computer-generated reasoning trace

Here is the output that the thermal diagnostic component produces while it is diagnosing a one particular mishap: a dirty condenser. This example is chosen only because the trace is reasonably long. The sort of text generated by investigations of other mishaps is similar.

```
TDIAG recommends:
Check backpressure.
In the current simulation,
the result of this would be:
Backpressure is 5.3.


In light of the following
facts at unit 5:
Backpressure is too high.
TDIAG recommends:
Calculate TTD.
In the current simulation,
the result of this would be:
TTD is 24.


In light of the following
facts at unit 5:
Backpressure is too high.
TTD is too high.
TDIAG recommends:
Check vent gas flow.
In the current simulation,
the result of this would be:
Vent gas flow is 9.2.


In light of the following
facts at unit 5:
Backpressure is too high.
TTD is too high.
Vent gas flow is too high.
TDIAG recommends:
Determine air content of vent
gas flow.
In the current simulation,
the result of this would be:
Condenser air flow is 1.0.


In light of the following
facts at unit 5:
Backpressure is too high.
TTD is too high.
Vent gas flow is too high.
Condenser air flow is ok.
TDIAG recommends:
Do a temperature traverse on
the condenser.
In the current simulation,
the result of this would be:
Temperature traverse
indicates dirty condenser.


Dirty condenser is costing
229. thousand dollars a year.
```