In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*
Los Altos, CA: Morgan Kaufmann, 1989.

Model-Based Monitoring of Dynamic Systems[*]

Daniel Dvorak[†]
AT&T Bell Laboratories
2000 N. Naperville Road
Naperville, Illinois 60566

Benjamin Kuipers
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712

Abstract

Industrial process plants such as chemical refineries and electric power generation are examples of continuous-variable dynamic systems (CVDS) whose operation is continuously monitored for abnormal behavior. CVDSs pose a challenging diagnostic problem in which values are continuous (not discrete), relatively few parameters are observable, parameter values keep changing, and diagnosis must be performed while the system operates.

We present a novel method for monitoring CVDSs which exploits the system's dynamic behavior for diagnostic clues. The key techniques are: modeling the physical system with dynamic qualitative/quantitative models, inducing diagnostic knowledge from qualitative simulations, continuously comparing observations against fault-model predictions, and incrementally creating and testing multiple-fault hypotheses. The important result is that the diagnosis is refined as the physical system's *dynamic* behavior is revealed over time.

Introduction

Process monitoring is a continuous real-time task of recognizing anomalies in the behavior of a dynamic system and identifying the underlying faults. This task is common in many industries (*e.g.,* electric power generation, chemical processing, etc.) and in medicine (*e.g.,* cardiac monitoring). In contrast to earlier work on diagnosis, process monitoring poses three special difficulties:

1. *Diagnosis must be performed while the system operates.* Process systems are designed for continuous operation and are capable of operating with multiple minor faults. Shutdown for diagnosis and repair is either costly (in industry) or impossible (in medicine).
2. *Few system parameters are observable.* All measurements come from sensors, which can be expensive and/or unreliable and/or invasive. Monitoring is typically based on a small subset of the system parameters, with limited opportunity to probe other parameters.
3. *The systems are dynamic.* The system exhibits time-varying behavior, parameter values vary over a continuous range, the system has state (*i.e.,* has integrated quantities), and feedback is common.

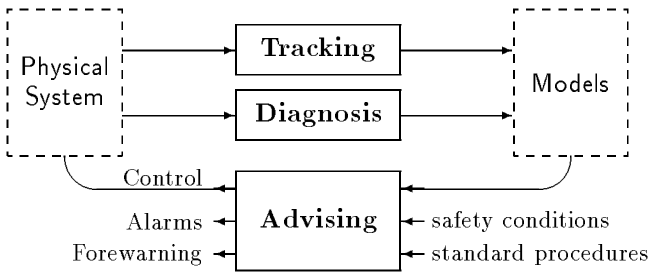

Figure 1: The 3 tasks of monitoring and control.

Automated process monitoring systems typically provide a set of alarms which are triggered whenever fixed thresholds are exceeded. A nuclear power plant, for example, can have over a thousand distinct alarms, and hundreds of them can be activated within a minute, as in a loss-of-coolant accident. In such situations, process operators may overlook relevant information, respond too slowly, panic when the rate of information flow is too great, and form incorrect mental models [Perrow, 1984]. The monitoring method described here is intended as an aid to help overcome these problems.

Model-based Monitoring

This paper describes MIMIC, a model-based method for monitoring dynamic systems in which the condition of

[*]This work has taken place in the Qualitative Reasoning Group at the Artificial Intelligence Laboratory, UT-Austin. Research of the Qualitative Reasoning Group is supported in part by NSF grant IRI-8602665 and by NASA grants NAG 2-507 and NAG 9-200.

[†]Supported by the AT&T Doctoral Support Program.

the physical system is represented (and repeatedly updated) in a dynamic qualitative model. The intent is to mimic the condition of the physical system in the model. Two tasks maintain the model, as shown in Figure 1. The *tracking* task advances the state of the model in step with observations from the physical system. The *diagnosis* task, upon identifying a particular fault, injects that fault into the current model so that the predictions of the model will continue to agree with actual observations. To be precise, MIMIC maintains a *set* of candidate models since a given behavior may be caused by one of several faults. Each candidate model represents a possible condition of the system (state and faults).

The purpose of monitoring is to determine the possible conditions of the physical system. The role of the *advising* task is to present this information to the operator and assist in interpreting it and making decisions about control actions. Since the models are predictive, they can be used to predict the effect of proposed control actions and forewarn of trends leading to undesirable conditions. This paper focuses on the tracking and diagnosis aspects, not on the advising aspect.

Dynamic Qualitative Models

Two main properties are required of the simulation technique used in MIMIC — it must reveal the time-varying (*i.e.,* dynamic) behavior of the system, and it must make explicit the behavioral distinctions important in diagnosis. We use the QSIM [Kuipers, 1986] method for qualitative simulation of dynamic systems. Just as modern control theory represents a dynamic system as a set of coupled first-order differential equations, QSIM represents a dynamic system as a set of coupled first-order *qualitative* differential equations. Simulated dynamic behavior is represented as a sequence of states, with alternate states representing a time point or time interval in the dynamic behavior.

By using a qualitative model rather than a numeric model, an infinite number of infinitesimally close numeric behaviors is reduced to a small number of qualitatively distinct behaviors. Although QSIM is fundamentally qualitative, it can exploit available quantitative information to refine its predictions [Kuipers and Berleant, 1987]. This capability proves to be very important in process monitoring because many sensors provide quantitative values and some faults can only be diagnosed by their subtle-but-quantitatively-noticeable effects.

Basic Cycle

MIMIC accomplishes tracking and diagnosis in a hypothesize-and-match cycle that combines associative and model-based reasoning. In effect, the associative component *proposes* fault hypotheses and the model-based component *disposes* of them. The cycle has four main steps, as shown in Figure 2:

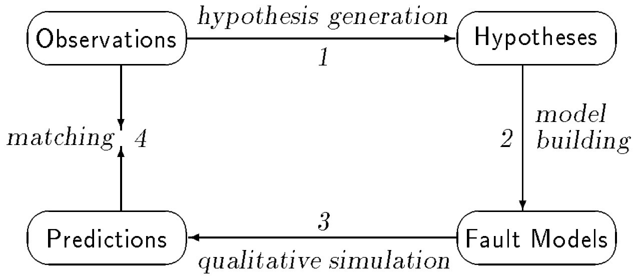

Figure 2: Hypothesize-build-simulate-match cycle.

1. *Hypothesis Generation.* Observations from the physical system may evoke fault hypotheses via a decision tree (the decision tree is generated beforehand, as described in section). The fault hypotheses are in the form of specific failure modes (such as a stuck pressure regulator or an abnormal setpoint) and are ordered by likelihood.
2. *Model Building.* Given a combination of one or more fault hypotheses, the corresponding QSIM fault model is instantiated by initializing setpoint variables and mode variables.
3. *Qualitative Simulation.* Each new fault model is first initialized from the observations that evoked its construction, thus establishing the initial state of the model. The model is then simulated incrementally as observations change, predicting the immediate successor states. QSIM generates qualitative values and quantitative ranges for each parameter. The ranges may be very precise (*e.g.,* [98.5 98.7]) or imprecise (*e.g.,* [0 ∞]) depending on the available quantitative knowledge.
4. *Matching.* A similarity function computes the similarity between the observations and a state of the model. The comparison is based on both qualitative and quantitative values. For similarities above a threshold, the model is retained as a plausible reflection of the physical system's condition. Below the threshold, the model is discarded.

Learning Diagnostic Knowledge

The knowledge used in MIMIC's hypothesis generation step is mechanically derived from the model of the dynamic system. The basic technique, as described in [Bratko *et al.*, 1986], induces fault diagnosis rules from the results of simulating qualitative fault models. This technique has been extended to the domain of continuous-variable dynamic systems, as detailed in [Lee and Dvorak, 1989]. Briefly, diagnostic knowledge is derived through five steps:

1. *Model Definition.* A model of the physical system is defined in terms of qualitative differential equations (QDEs), with some QDEs conditional on the operating mode of the component whose behavior they

model. For example, a pump will have one set of constraints associated with its normal mode of operation and another set associated with a "broken" mode.

2. *Model Building.* Using this model definition, a model-building program instantiates the normal (fault-free) model, all single-fault models, and selected combination-fault models. The issue of *which* combination-faults to model is discussed in section .
3. *Qualitative Simulation.* Using QSIM, each model is simulated starting from each possible initial state, producing a total envisionment for each model.
4. *Construction of Training Set.* Training instances are formed from the states of the total envisionment using the qualitative magnitude and qualitative direction-of-change of each observable parameter. Each instance is tagged with the fault combination embodied in the model that generated the state. Collectively, these instances form the training set.
5. *Induction.* The training set is compressed by an inductive learning program to a smaller body of operational diagnostic knowledge in the form of a decision tree. The induction algorithm is similar to ID3 [Quinlan, 1986], but exploits three additional sources of knowledge: the observability of each parameter, the *a priori* probabilities of faults, and the historical probabilities of behaviors.

The resulting decision tree is used to classify observations from the monitored system, yielding fault hypotheses. The classification procedure ranks the resulting fault hypotheses by likelihood, thus allowing MIMIC to focus attention on the most probable faults. The procedure also produces a ranked set of manually measurable parameters whose values, if measured, could reduce the number of hypotheses. MIMIC presents this information to the system operator.

The learning procedure described above can consume a large amount of computer time, but it is performed only once, outside of the real-time monitoring cycle. As Pearce [1988] has demonstrated, this approach to knowledge acquisition provides more complete coverage of faults than the traditional knowledge engineering approach.

Multiple-Fault Diagnosis

The number of fault models that need to be constructed depends on the characteristics of the system being modeled and the importance of detecting multiple faults. In some domains, single-fault diagnosis is adequate, but in general, multiple faults are common in complex continuous-running systems. However, complete multiple-fault diagnosis is combinatorially explosive and therefore unrealistic for real-time monitoring of large systems. As a middle approach, MIMIC uses a method for incrementally constructing and testing multiple-fault hypotheses. The key ideas are described below.

1. MIMIC does *continuous* monitoring, repeatedly reading the sensors. We assume that faults usually occur one-at-a-time with respect to the sampling rate, so any unpredicted behavior will normally be due to a single additional fault (or a single repair). Thus, MIMIC usually only needs to deal with *one* new fault at a time.
2. Single-fault diagnostic knowledge triggers fault hypotheses whenever any of the manifestations of a single fault are present. Many faults *don't* interact, so they can be recognized even in the presence of other faults.
3. Double-fault diagnostic knowledge captures the manifestations peculiar to every pair of *interacting* faults. So, if a pair of faults interact in a way that obscures either or both of the individual faults, this knowledge can detect both faults.
4. The hypothesize-and-match algorithm described earlier is capable of injecting any number of faults into a model. When a fault is hypothesized it may be injected into models that already embody one or more faults. Thus, multiple-fault hypotheses are incrementally constructed and tested.
5. Hypotheses are combined in a beam search based on the similarity function described earlier. Let N be the maximum number of models to be tracked, T be the set of models currently being tracked, and F be the newly proposed fault hypotheses. At each cycle then, MIMIC retains the N best models from $T \cup (T \times F)$, ranked by similarity value.

Tracking a Model

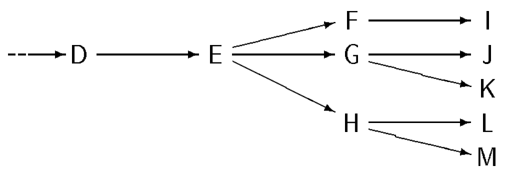

Figure 3: Tracking through a behavior graph.

When a fault model is first constructed, an attempt is made to initialize it from current observations. If the observations are not consistent with any state of the fault model, then the model is discarded. If there is at least one consistent initialization then the model becomes a member of the set of candidate models, and all of its consistent initializations (there may be more than one) are added to the "tracking set". The tracking set is a set of models, each in a state consistent with the most recent observations.

Tracking is the process of using the observations to follow a path through the behavior graph of a model (a "partial envisionment"). Consider the fragment of behavior graph in Figure 3. If a model is currently in state E, then a new set of observations is compared (using the similarity function) to the parameter values

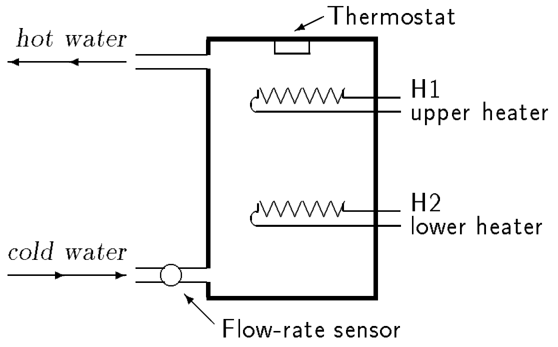

Figure 4: Electric water heater.

of state E. If the match is above a threshold, then the model remains in state E. Otherwise, the immediate successor states of state E are generated (via incremental simulation) and each of these states is compared with the observations. If, say, the match with state G was above the threshold, then the model is retained with its state now set to G. If none of the successor states match (up to some limited "distance" from E), then the model is discarded. The limited distance is needed to jump over instantaneous states in the envisionment that fall between consecutive observations.

Observations may include independent parameters, *i.e.*, parameters which are under external control and whose values thus cannot be predicted. When an independent parameter changes value, tracking must reinitialize the states in the tracking set using the current observations and values of history variables (rather than looking in vain for a compatible successor state).

It is possible that tracking could discard *all* the candidate models. This condition could arise if either: (1) some type of fault was overlooked in the description of possible faults, or (2) the diagnostic knowledge failed to propose one or more faults of a combination fault. In such a case MIMIC (1) alerts the operator that the physical system is behaving abnormally but cannot be diagnosed, and (2) displays the fault hypotheses evoked by the current observations.

Example

To illustrate MIMIC at work, let's consider the electric water heater shown in Figure 4. It has a single thermostat which controls whether or not power is applied to the two heating elements (on-off control). Raw sensor information comes from a temperature sensor near the thermostat, from a flow-rate sensor on the cold-water inlet, and from the electric terminals of the heating elements. In a real monitoring situation we would want to diagnose a variety of possible faults such as defective heating elements, a stuck thermostat, a faulty flow-rate sensor, and loss of electrical power. However, to keep this example simple, we'll consider only the possibility of defective heating elements.

The water heater is modeled in QSIM according to the laws of thermodynamics that relate heat capacity, heat flow, thermal resistance, and temperature. In the normal (fault-free) model all the components of the water heater (tank, heating elements, thermostat, flow-rate sensor) operate according to their intended design. In a fault model, a faulty component operates according to a failure mode (such as a heating element that generates no heat when power is applied).

Table 1 summarizes an example of monitoring the water heater, showing how monitoring progresses over eight moments in a series of observations[1]. For each moment, the table shows the quantitative sensor readings and three sets maintained inside MIMIC. The water heater begins in a state where the water in the tank is hot, the heating elements are off, no water is flowing, and there is a slow temperature loss. These readings are consistent with the normal model. Now, someone starts to draw water for a bath. A high flow rate is measured but all other readings remain the same. Since water flow is an independent variable, MIMIC reinitializes every tracked model (just the normal model in this case) to reflect the change. Since the normal model is consistent with the new values, it is retained.

As time continues, the temperature inside the tank drops because of the cooler inlet water. These readings are consistent with the current state of the normal model and evoke no fault hypotheses, so no change occurs to the tracking set. At moment 3 the temperature drops to the point where the heating elements turn on (as observed on a voltage sensor), These readings are also consistent with the normal model, so the state of the model is updated accordingly.

At moment 4 the temperature continues to drop. Although this observation is *qualitatively* consistent with the normal model, it is inconsistent with the associated *quantitative* ranges. In effect, the model is saying that for this flow rate, tank capacity, heating rate, and inlet temperature, the water temperature should not be dropping so fast. Thus, the tracking task discards the normal model. At the same time, the readings are classified by the decision tree as being suggestive of three possible faults — a bad upper heating element, a bad lower heating element, or both heating elements bad (denoted bad-H1, bad-H2, and bad-H1,H2)[2]. This causes three fault models to be built. Each model is successfully initialized, so MIMIC is now tracking three models.

The water flow stops at moment 5 (somebody turned off the faucet). With this change in an independent parameter, MIMIC reinitializes the three models. At moment 6, the temperature is observed to be rising.

[1]The numeric values shown in Table 1 are from a numeric simulation of the water heater in which the lower heater is burned out.

[2]In a more detailed example, other hypotheses would also be proposed, such as a faulty temperature sensor and a faulty flow meter.

Moment	0	1	2	3	4	5	6	7
Synopsis	temp hot	flow starts	temp dropping	heater on	temp still dropping	flow stops	temp rising	heater off
Time (min.)	0.0	1.0	2.0	2.4	2.7	3.0	13.0	27.7
Flow (liters/min.)	0	30	30	30	30	0	0	0
Temp. (deg. C)	64.9	64.9	61.4	58.9	57.1	55.9	60.0	66.0
Power (on or off)	off	off	off	on	on	on	on	off
New fault hypotheses	none	none	none	none	bad H1 bad H2 bad H1,H2	none	none	none
New fault model(s)	none	none	none	none	bad H1 bad H2 bad H1,H2	none	none	none
Tracked model(s)	normal	normal	normal	normal	bad H1 bad H2 bad H1,H2	bad H1 bad H2 bad H1,H2	bad H2	bad H2

Table 1: Diagnosing the water heater from dynamic behavior.

This observation is qualitatively inconsistent with the bad-H1,H2 model, so this model is discarded. The observed temperature is then compared to the quantitative predictions of the two remaining models. Because the observed temperature exceeds the range predicted by the bad-H1 model, that model is discarded. The predictions of the one remaining model, bad-H2, are compatible with the observations, so the model is retained. This model continues to track future readings, thus emerging as the sole fault hypothesis.

Discussion

The water heater example shows how, with few observable parameters, MIMIC can diagnose a system by observing its dynamic behavior. In general, the speed at which a diagnosis can be narrowed depends on the number of monitored parameters and the dynamic activity of the system. With more monitored parameters and more system activity, there are more opportunities to falsify hypotheses.

As a diagnostic method, MIMIC can generate both false positives and false negatives. False positives are common because the relatively small number of observed parameters cannot, in a single reading, discriminate among all the possible faults. Thus, all but one of the candidates will be a false positive. However, with additional readings that reveal the system's dynamic behavior, the set of candidates can be reduced, sometimes to a single candidate.

False negatives can arise either because some type of failure was overlooked in the knowledge acquisition phase or because a combination of three or more faults interacted in a way that obscured one or more of the individual faults. The former case is solved by adding the new failure type and rerunning the diagnostic learning procedure. The latter case is more difficult. One approach is to learn diagnostic knowledge for more than just single and double faults. In the general case this is impractical since the number of fault combinations is exponential in the number of concurrent faults. However, in some domains (such as in medicine) the number of *realistic* combination faults may be tractable because many of the combinations are physiologically impossible or medically uninteresting [Bratko *et al.*, 1986].

Limitations

MIMIC assumes that faults occur one-at-a-time with respect to its sampling rate. This assumption may be violated in the case of a catastrophic event (such as an explosion) or cascading faults; these are real conditions that MIMIC does not address. Also, MIMIC cannot guarantee that it will recognize any combination faults beyond the combinations included during learning. In practice, it *will* diagnose many such non-learned combination faults, but the faults *may* interact in a way that obscures the manifestations of some subset of those faults.

Related Work

Measurement interpretation is a significant part of the job of monitoring, and this aspect of MIMIC shares some of the ideas set forth by Forbus [1986]. In particular, MIMIC's notion of *tracking* combines two elements of Forbus' approach: that of finding temporally adjacent states in the envisionment which correspond to temporally adjacent measurements, and jumping over short gaps where consecutive measurements have missed an intervening instantaneous state in the envisionment.

Mimic is similar to Premon [Doyle *et al.*, 1989] in that both use a qualitative model of the physical system to perform monitoring. However, Premon is concerned with monitoring a correctly functioning system in a changing environment. It uses a single fault-free model in a predict-plan-sense cycle for dynamic adjustment of alarm thresholds and for deciding which sensors to focus attention on (where the number of sensors is large). This work is complementary to Mimic's focus on diagnosis, and raises the possibility of combining the techniques to perform diagnostic monitoring on very large systems.

Mimic differs markedly from model-based troubleshooters such as GDE [de Kleer and Williams, 1987] in that it (a) uses fault models and (b) does *not* use dependency tracing or constraint suspension [Davis, 1984]. Fault models are necessary in Mimic because of the need to track a faulty system's behavior over time. Faults are specifically modeled (rather than suspending constraints) in order to get reasonably detailed predictions of behavior. Dependency tracing, although extremely useful in domains such as digital logic circuits, provide little diagnostic power in a system of constraints among continuous variables having feedback loops. The problem is that dependency tracing in such systems often returns *all* constraints as suspects because: (a) all parameters of a constraint usually affect the result, and (b) output parameters often feed back to input parameters.

The high-level design of Mimic is similar to the "generate, test and debug" (GTD) paradigm [Simmons and Davis, 1987] in that both use associational knowledge to generate plausible hypotheses and model-based reasoning to evaluate them. The core idea in GTD is of "debugging almost right plans" whereas in Mimic it is of "debugging almost right models".

A number of expert systems have been built which share the same operational goal as Mimic — that of relieving some of the burden of monitoring from process operators [Dvorak, 1987]. Mimic focuses solely on determining the condition of the physical system, but most of these expert systems have the broader scope of trying to advise the operator on corrective actions. Escort [Sachs *et al.*, 1986], an exemplar of this group, gets its knowledge of faults and anomalies and corrective actions through the usual process of codifying human expertise in rules; Escort does not encode a predictive model of the physical system as Mimic does.

Implementation Status

Prototypes of Mimic and its companion learning algorithm (named DynaLearn) are implemented in Common Lisp and have been run on a Symbolics 3670.

Conclusions and Future Work

This paper has presented a technique for online diagnosis (*i.e.*, monitoring) of continuous-variable dynamic systems. The key elements of the design are: (1) representation of continuous-variable dynamic systems in dynamic qualitative/quantitative models, (2) induction of diagnostic knowledge from model simulations, (3) tracking fault-model predictions against observations, and (4) incremental creation of multiple-fault hypotheses. The important result is that Mimic exploits the system's dynamic behavior for diagnostic clues.

Work is progressing on a hemodynamic model of the human cardiovascular system with an expert cardiologist. The complete model will have 4 state variables, about 50 parameters, and several regulatory mechanisms (negative feedback loops). As a model of realistic complexity and scale, this will better reveal limitations of the Mimic design and areas for improvement.

References

Ivan Bratko, Igor Mozetič, and Nada Lavrač. *Expert Systems: Automating Knowledge Acquisition*, chapter Automating synthesis and compression of cardiological knowledge. Addison-Wesley, 1986.

Randall Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24(3):347–410, December 1984.

Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, April 1987.

R. J. Doyle, S. M. Sellers, and D. J. Atkinson. A focused, context-sensitive approach to monitoring. In *Proceedings of the 1989 International Joint Conference on Artificial Intelligence*, August, 1989.

Daniel Dvorak. Expert systems for monitoring and control. Technical Report AI87-55, Department of Computer Sciences, The University of Texas at Austin, May 1987.

Wan-Yik Lee and Daniel Dvorak. Learning diagnostic knowledge for dynamic systems. Technical Report AI89-xx (forthcoming), Department of Computer Sciences, The University of Texas at Austin.

Kenneth D. Forbus. Interpreting measurements of physical systems. In *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, pages 113–117. August 1986.

Benjamin Kuipers. Qualitative Simulation. *Artificial Intelligence*, 29(3):289–338, September 1986.

Benjamin Kuipers and Daniel Berleant. Using incomplete quantitative knowledge in qualitative reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 324–329. August 1988.

D. A. Pearce. The induction of fault diagnosis systems from qualitative models. In *Proceedings of the*

Seventh National Conference on Artificial Intelligence (AAAI-88), pages 353–357. August 1988.

Charles Perrow. *Normal Accidents*. Basic Books, Inc., New York, 1984.

J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81–106. 1986.

Paul A. Sachs, Andy M. Paterson, and Michael H. M. Turner. Escort – an expert system for complex operations in real time. *Expert Systems*, 3(1):22–29, January 1986.

Reid Simmons and Randall Davis. Generate, test and debug: combining associational rules and causal models. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 1071–1078. August 1987.