01 Jan 1996

Tuning Numeric Parameters to Troubleshoot a Telephone-Network Loop

Christopher J. Merz
Missouri University of Science and Technology, merzc@mst.edu

M. J. Pazzani

A. P. Danyluk

Tuning Numeric Parameters to Troubleshoot a Telephone-Network Loop

Christopher J. Merz and Michael J. Pazzani, University of California, Irvine
Andrea Pohoreckyj Danyluk, Williams College

THE NYNEX MAX EXPERT SYSTEM ANALYZES THE RESULT OF AN AUTOMATED ELECTRIC TEST ON A TELEPHONE LINE AND DETERMINES THE TYPE OF PROBLEM. HOWEVER, TUNING THE SYSTEM'S PARAMETER VALUES CAN BE DIFFICULT. THE OPTI-MAX SYSTEM CAN AUTOMATICALLY SET THESE PARAMETERS BY ANALYZING DECISIONS MADE BY EXPERTS WHO TROUBLESHOOT PROBLEMS.

OUR RESEARCH ADDRESSES THE challenge of learning to troubleshoot a telephone network using data provided by Nynex (the parent company of Nynex New England and Nynex New York, formerly New England Telephone and New York Telephone). Nynex has implemented a rule-based expert system, the Maintenance Administrator Expert (Max),[1] which determines a malfunction's location for customer-reported telephone troubles. In particular, Max troubleshoots the *local loop*, the part of the telephone network from the central office to the customer's premises. Max, as well as other systems generated from the original system, are an important part of the operations of Nynex New York and Nynex New England, the largest phone companies for those regions.

Like all expert systems, Max requires occasional maintenance to its knowledge base. In addition, many different sites in New York and New England use Max, and there are small differences in how each site diagnoses reported troubles. Max's designers have facilitated this customization via numeric parameters (indicating, for example, when a voltage is too high) that each site sets or that the designers adjust periodically to improve Max's performance. Our goal is to develop strategies to tune these parameters for improved performance on the examples. Because of the many troubles processed by Max and Max-related systems, a phone company can save a significant amount of money with even a small improvement in performance (as little as 1%).

In this article, we describe the troubleshooting problem and Max in more detail and discuss how we collected the data for the study. Then we describe Opti-Max, our approach for revising Max's parameters, and describe the results of using Opti-Max to improve these parameters.

The Nynex-Max expert system

Max is an expert system developed by Nynex Science and Technology for the high-level diagnosis of customer-reported telephone troubles. Max runs on Sun workstations and is implemented in Inference Corp.'s Automated Reasoning Tool (ART). Max, a moderately complex system comprising over 150 rules, diagnoses troubles in the local loop. Given information about the customer's line, Max determines roughly where the trouble lies and selects the type of technician that should address the trouble.

When Nynex customers have problems with their lines, they call a special number to report the trouble. A phone company representative takes information from the customer about the trouble's symptoms and creates a trouble report. At the same time, the representative initiates electrical tests on the line—AT&T's Mechanized Loop Test. The representative then attaches the data that the MLT gathers (for example, voltage readings) to the trouble report and sends it to a maintenance administrator for diagnosis. The MA also receives a rough diagnosis from a primitive rule-based system called the Screening Decision Unit. This unit makes its diagnosis on the basis of a *vercode*—a code that represents MLT's summary of its test results. The MA uses the information from the trouble report, MLT, and Screening Decision Unit to make a high-level diagnosis of the trouble. On the basis of this diagnosis, the MA determines to what part of the local loop to dispatch a repair technician: the central office, the cable, or the customer's home. The MA can also specify that a test should be performed on the trouble before it can be dispatched. (Phone company operations are constantly evolving; this gives one view that is adequate for understanding the Max expert system.)

Max performs the same role as the MA except that it also may send a trouble to a human MA when it is unable to make a diagnosis. (A human MA does not have the option to refer a trouble to another MA.) Therefore, while the human MA has four choices, Max has five possible outcomes for each case: dispatch to the central office, dispatch to the cable, dispatch to the customer's home, perform a test, or send to a human MA.

Max is a rule-based system that makes its diagnosis on the basis of the MLT results as well as other general information, such as the type of switching equipment for the customer's line. All maintenance locations throughout New York and New England use one general rule base, but it contains parameters that can be tuned for local conditions. The segment below is a simplified fragment of a rule. (In this segment, numeric parameters that Opti-Max can change are capitalized, variables whose values are set by the MLT are in small letters, and diagnoses are italicized.)

```
If (highest_measureZ >=
  THRESHOLD2)
    Dispatch to customer's
    home
Else If (highest_measureZ
  >= THRESHOLD3)
    Hold for further
    testing
Else
    Dispatch to cable
```

This rule tests voltage values against some parameter settings. The conditions test for high voltages. The classification produced by Max determines where to dispatch a repair person.

Max is currently in use throughout Nynex New York and New England. Since its deployment in 1990, Max has been modified and expanded for other related tasks in Nynex as well. Among its benefits, it is fast, consistent, and reduces the number of incorrect dispatches over the Screening Decision Unit, which is a primitive rule-based system. Some of Max's limitations are that it is not always correct and that the local parameters are difficult to tune properly.

Data used in this study

To evaluate how well Max performs and how much Opti-Max improves Max, Nynex collected 500 examples of troubles (from one geographical region, or site) that customers had reported. Max maintains a log of all troubles it has diagnosed, including all data used to describe the troubles. Max does not maintain logs forever, but these logs do provide a useful mechanism for collecting trouble descriptions. Nynex presented the 500 example descriptions to between two and four human maintenance and dispatch experts (essentially expert MAs), who then analyzed the troubles. (In fact, Nynex implemented Max's initial knowledge base from this type of expertise.)

TO GATHER THE EXPERT DIAGNOSES, WE CREATED AN ON-LINE SURVEY SYSTEM THAT WOULD LET THE EXPERTS EASILY CONSIDER TROUBLES AND INPUT THEIR DIAGNOSES.

To gather the expert diagnoses, we created an on-line survey system that would let the experts easily consider troubles and input their diagnoses. We established a system that displays one trouble at a time on the screen in the same format as trouble reports are displayed for an MA. Although this is not necessarily the clearest way to represent the trouble report, it is a format that the experts are very comfortable with. We asked our experts to diagnose each trouble. Alternatively, an expert might indicate that Max should send a case to a human MA. For troubles where four experts had provided diagnoses, we chose the answer given the majority of the time (if there was one) as the correct diagnosis. In all other cases, we constructed a set of acceptable answers, defined as any answer given by an expert. We consider it acceptable for an automated system to react like any expert would on a particular case.

Using this definition of acceptable, each case has an average of 1.84 acceptable diagnoses. Max gave a diagnosis matching the acceptable answers in 66.2% of the 500 examples. With five possible diagnoses, if Max were to guess randomly, it would be 36.8% accurate. Although getting 66.2% correct diagnoses may seem low, in other studies[1] Nynex has found that Max generally performs at least as well as experienced MAs. Some figures indicate a range of 80–95% accuracy, depending on the way accuracy is measured.[2]

In our study, conducted in a laboratory at the University of California, Irvine, we used only a subset of the complete Max knowledge base, which contributed in part to Max's performance. (The complete Max knowledge base hasn't been released to universities for research purposes because of its proprietary nature.) In general, though, Max does not operate at 100% accuracy, because it is a broad system that must be tuned for good site-dependent performance. In fact, the capability to tune parameters effectively has eluded the system's developers and maintainers.

We use the data as follows: We repeatedly divide the 500 cases into two sets, one with 400 examples and the other with 100. We give the 400 cases (the training set) to Opti-Max so that it can tune Max to agree with the expert diagnoses. And we use the 100 examples in the other set (the test set) to evaluate the tuned parameters.

Alternative sources of data

The lack of agreement among the experts motivated us to explore alternative data sources. (The experts were, on average, giving nearly two answers for each example description.) Data for this task are particularly problematic, and we considered various data sources. In addition, we explored various data-engineering approaches to try to extract useful information from this bad data.[3]

```
Given:
      Parameters - a set of numeric parameters
      Examples - a set of classified examples

Errors = Errors(Examples, Parameters)
Non_improvements = 0
ReviseLoop: Changed = False
      For each Parameter in Parameters
              OldValue = Parameter
              Parameter = Uniform(0, Parameter)
              NewErrors = Errors(Examples, Parameters)
              Effect = NewErrors - Errors
              If (Beneficial(Effect)) Then Changed = True
              Else If (Zero(Effect) And Random(.5) = True))
                      Then Errors = NewErrors
                      Else Parameter = OldValue
      For each Parameter in Parameters
              OldValue = Parameter
              Parameter = Uniform(Parameter, 2 * Parameter)
              NewErrors = Errors(Examples, Parameters)
              Effect = NewErrors - Errors
              If (Beneficial(Effect))
                      Then Changed = True
              Else If (Zero(Effect) And Random(.5) = True))
                      Then Errors = NewErrors
                      Else Parameter = OldValue
      For each Parameter in Parameters
              OldValue = Parameter
              Parameter = Uniform(.95 * Parameter, Parameter)
              NewErrors = Errors(Examples, Parameters)
              Effect = NewErrors - Errors
              If (Beneficial(Effect))
                      Then Changed = True
              Else If (Zero (Effect) And Random(.5) = True))
                      Then Errors = NewErrors
                      Else Parameter = OldValue
      For each Parameter in Parameters
              OldValue = Parameter
              Parameter = Uniform(Parameter, 1.05 * Parameter)
              NewErrors = Errors(Examples, Parameters)
              Effect = NewErrors - Errors
              If (Beneficial(Effect))
                      Then Changed = True
              Else If (Zero(Effect) And Random(.5) = True))
                      Then Errors = NewErrors
                      Else Parameter = OldValue

If Changed = True
      Then Non_improvements = 0
      Else Non_improvements = Non_improvements + 1
If (Non_improvements < Maxwander)
      Then GoTo ReviseLoop
      Else Return Parameters
```

Figure 1. The hill-climbing procedure for revising numeric parameter values. We use a value of 12 for `Maxwander`. `Uniform(X,Y)` returns a random number between X and Y. `Random(X)` returns `True` when a generated random number between zero and one is greater than X.

Despite problems, the task of tuning Max's parameters remains important to Nynex. Hence, we consider the expert-provided diagnoses to be a reasonable starting point for doing this tuning.

Machine learning approaches to tuning Max

We have investigated several learning approaches for the Max domain—for example, inductively learning a new knowledge base from data, modifying the existing knowledge base, and tuning the numeric parameter values. In this article, we focus on parameter tuning, so now we briefly compare the other approaches to this one.

In inductive learning, the task is to create a new knowledge base using data that describes troubles and their diagnoses. The attributes used in inductive learning are the attributes that describe the troubles in Max. The classes that an inductive learner predicts are the diagnoses that the experts indicate Max should predict. We can apply any number of inductive learning algorithms, including decision trees[4] and neural networks,[5] to this learning problem. In fact, Nynex has experimented with decision trees and search techniques for generating new rules.[6] From a technical point of view, there is a possible disadvantage in using these inductive learning algorithms: They ignore all the knowledge contained in the Max rule base and focus solely on trying to find predictive relationships in the data. Deductive learners that are able to use existing domain knowledge, such as that encoded in the Max knowledge base, learn more accurately, and require fewer training examples than purely inductive learners.[7]

However, from an organizational point of view, there is a more serious problem with inductive learners. They produce decision trees or neural nets, instead of rules in ART that can be easily integrated into the existing operational environment. Nynex has expended considerable resources integrating Max into the computational environment of its maintenance centers, and Nynex would have to repeat much of this work if it switched to decision trees or neural nets. In addition, Nynex has spent a lot of time having experts verify the rules in the Max rule base. This work would also have to be repeated. Furthermore, Nynex can customize Max to a specific site by simply changing the numeric parameters. The decision tree or neural net would probably vary considerably at each site, increasing the effort required to manually inspect the rule base for errors. Besides these financial- and time-investment issues, larger social issues concern the integration of Max into the phone company operations. Any modifications made to the system, even if clear improvements, are most wisely made in a manner that integrates neatly into the existing system.

Theory revision systems, such as Either (Explanation-Based and Inductive Theory Extension and Refinement)[8] and FOCL (First-Order Combined Learner),[7] overcome some of the technical shortcomings of inductive learners. In particular, they take as input both a set of examples and the expert system's rule base, and they produce as output a modified set of rules. Experimental analysis typically shows such systems to be more accurate than inductive methods, when applicable. We have experimented with FOCL on Max data and the Max rule base.[9]

To do this, we translated the Max rule base by hand into Prolog, and then FOCL revised the Prolog rules. The revisions involved specializing some rules by adding extra conditions to them, deleting some rules that were incorrect, inducing some new rules, and deleting some conditions to rules.

FOCL cannot directly change a numeric parameter. Instead, it might start with a condition, such as `Voltage > 17.5` and `Resistance > 2,400`; delete a condition to produce `Voltage > 17.5`; and add a condition with induction to create `Voltage > 17.5` and `Resistance > 2500`. However, nothing prevents FOCL from adding a condition involving a different attribute, such as `Voltage > 17.5` and `Type-of-service = PBX`. Although we demonstrated that FOCL was more accurate than an inductive learner, many of the organizational problems remain with FOCL. In particular, FOCL produces a different set of rules per site, rather than a different set of parameters.

Parameter tuning for Nynex Max

Parameters are the only mechanism used to customize Max for different sites. Here, we explore an automated approach to setting these parameter values. We have implemented a system called Opti-Max, which takes as input the following items:

- Training examples, each consisting of two components: (1) a set of 21 numeric and symbolic variables that describe items such as the type of telephone equipment that the customer uses and the AC and DC voltages and resistances between pairs of wires, and (2) a set of acceptable diagnoses for the particular case. We obtained these diagnoses from experts in the area of local loop maintenance.
- The rule base that Max uses.
- Initial parameter values.

Opti-Max returns a set of revised parameter values that Max may use. The goal of the revision process is to reduce the number of times that Max disagrees with one of the experts. Such a parameter revision system has an advantage over other learning approaches in that it does not require changing Max or the environment where it currently operates.

The problem Opti-Max solves is essentially a function-optimization problem. The function to be optimized here is a complex function defined by symbolic rules. If we described the function and knew the derivatives of the function, we might be able to optimize it with standard mathematical methods such as gradient descent. Because this is not the case, we have explored various artificial intelligence search methods. Although the methods we employ are not guaranteed to find a globally optimal solution, we have shown experimentally that they can improve the existing parameters and help customize Max for a new location.

The methods we've tried include steepest ascent hill-climbing search, genetic algorithms,[10] and simulated annealing.[11] Opti-Max is a greedy hill-climbing search method. In previous experiments, we showed that an

OPTI-MAX RETURNS A SET OF REVISED PARAMETER VALUES THAT MAX MAY USE. THE GOAL OF THE REVISION PROCESS IS TO REDUCE THE NUMBER OF TIMES THAT MAX DISAGREES WITH ONE OF THE EXPERTS.

earlier version of a greedy hill-climbing search method performed at least as well as the other more complex search methods on a related problem but required much less time to revise the parameters.[12]

Figure 1 represents the Opti-Max algorithm. Basically, this algorithm randomly adjusts a parameter by setting it to random numbers between 0 and its current value, setting the parameter between its current value and twice its value, setting it to a value between its current value and 95% of that value, and setting it to a value between its current value and 105% of that value. For each, we evaluate how changing the parameter affects the training cases. If changing the parameter value reduces the number of mistakes made by Max, the algorithm changes the parameter value. If the change has no effect, then the parameter is changed 50% of the time. Otherwise, the algorithm restores the parameter to its current value. The algorithm repeats this process of cycling through all parameters, changing them to a random value, and evaluating the effect of the change until it has cycled through all parameters 12 times in a row without reducing the number of errors that Max makes.

In previous versions of a hill-climbing search, we tried steepest-ascent hill climbing. In this approach, we changed only the parameter whose value most decreased the number of errors made on each cycle through the parameters. However, this approach required many more cycles through the parameters than our current approach does. In addition, we tried hill-climbing operators that added or subtracted a fixed amount to each parameter value (for example, 2.5%) and increased this amount if no change had a beneficial effect. However, this approach was not as accurate as the more probabilistic approach we currently use.

Simulated annealing and genetic algorithms offer a potential advantage over our current approach. In particular, if there is an interaction among parameter values such that changing one parameter value is not beneficial unless another parameter value also changes, then the hill-climbing approach will not necessarily find such an improvement. However, this problem has not arisen in our experimental studies, where our current approach is at least as accurate as the other methods yet requires considerably less time.

Our approach changes a parameter fifty percent of the time if the change has no effect. We added this step to allow exploration of some feature interactions. We design the operators in our hill-climbing search (which test parameters at random values rather than predetermined discrete differences) on the basis of our experience with more complex, time-consuming search methods. Furthermore, our experience with genetic algorithms that can make both large and small changes to parameter values influenced our decision to explore both large changes (up to 100% of the parameter value) and small changes (within 5% of the parameter value).

Experimentation and results

We ran three experiments that demonstrate the effectiveness of the hill-climbing approach summarized in Figure 1. In each experiment, we split the data into 30 random partitions of testing and training data. We trained the search methods on subsets of the training partition (from 100 to 400 examples in increments of 100). Then we evaluated the method's effectiveness by observing how

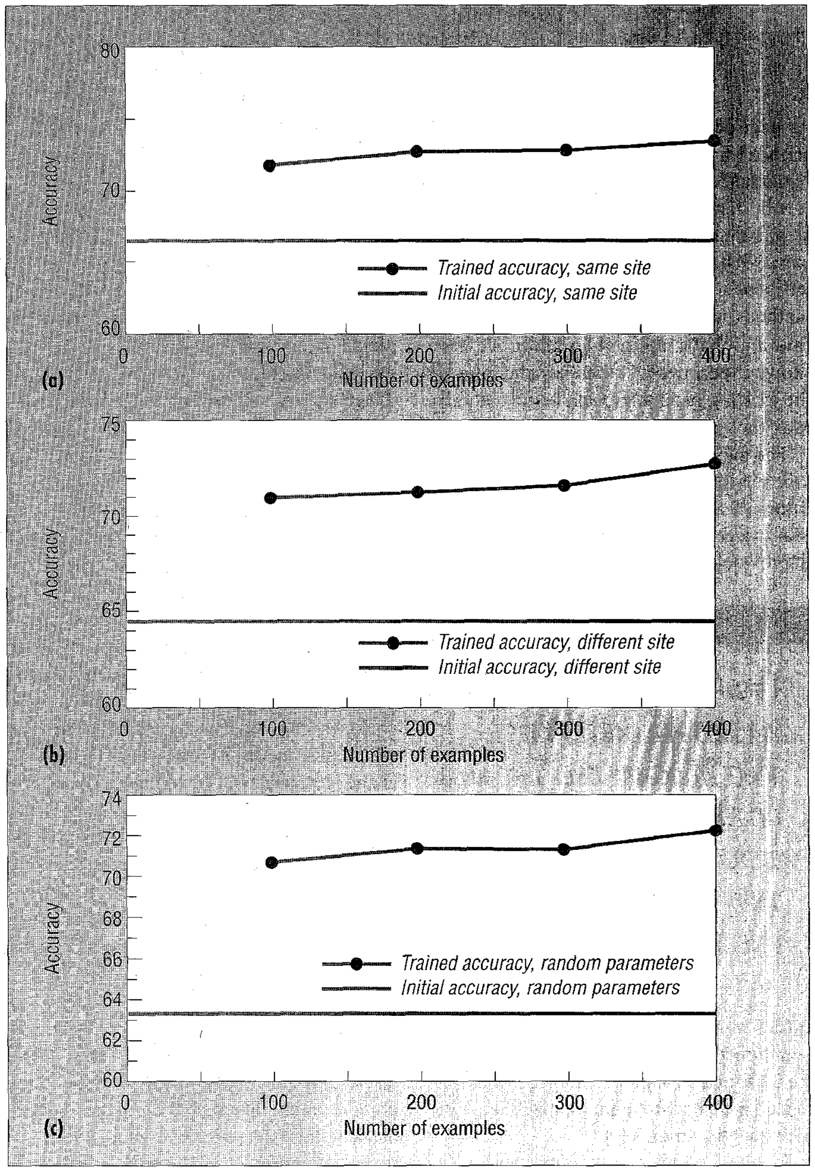

Figure 2. The results of using Opti-Max to (a) revise the parameter values currently in use to agree with expert diagnoses; (b) revise the parameter values currently in use at one site to agree with expert diagnoses at a different site; (c) revise the randomly set parameter values to agree with expert diagnoses.

well Max performed on the examples in the test partition. We averaged these results to generate the plots in Figure 2. Each experiment passed different initial-parameter values into Opti-Max. The three possible initial parameter configurations were

- *Same-site setting.* The actual parameters that Max uses are at the same site as where Nynex collected the training and test data. This tests Opti-Max's capability to fine-tune Max in an operational setting by using expert feedback.
- *Different-site setting.* The actual parameters that Max uses are at a different site than where Nynex collected the data. This tests Opti-Max's capability to customize Max to a new site, starting with a different site's parameters.
- *Random setting.* We chose random values for each parameter from a uniform distribution with a range from 75% of that parameter's value to 125% of that value. This tests Opti-Max's capability to tune the system, starting with reasonable but erroneous parameter settings.

Figure 2a plots Max's average accuracy using the parameters that Opti-Max returns after training, when the initial parameters are those Max is currently using at the same site where Nynex collected the data. Under these conditions, the initial accuracy of Max was 66.2%. By tuning the parameters to emulate expert decisions, Opti-Max increased the accuracy of Max to 71.7% with 100 examples and to 73.1% with 400 examples. This demonstrates that Opti-Max can improve Max by making it agree with expert diagnosticians.

Figure 2b plots the average accuracy of Max using parameters that Opti-Max tunes when the initial parameters are the parameters that Max is currently using at a site that differs from the site where Nynex collected the data. In this experiment, we used the same data as in the previous experiment, but the initial parameter values were different. Here, Max's initial accuracy was 64.5%. By tuning the parameters to emulate expert decisions, Opti-Max increased the accuracy of Max to 71.0% with 100 examples and to 72.7% with 400 examples. This demonstrates that Opti-Max can customize Max to a new site by emulating the expert decisions at that site.

Finally, Figure 2c plots average accuracy of Max using parameters that Opti-Max tunes when we assign the initial parameters as random numbers within 25% of the values currently used. Under this condition, Max's initial accuracy was 63.2%. By tuning the parameters to emulate expert decisions, Opti-Max increased the accuracy of Max to 72.0% with 400 examples. This demonstrates that Opti-Max can customize Max when the parameter values are set to random, but reasonable, values.

WE HAVE DEMONSTRATED A parameter-tuning strategy, based on a hill-climbing search, for the Nynex Max trouble-screening expert system. Max currently has a general set of rules that Nynex customizes for local optimization by setting parameter values. While customizing the rule base is extremely valuable, it is currently not very effective. The parameters are difficult to tune, because they may interact in ways that are not obvious when one looks at individual rules. Tuning them with a more global perspective would involve tracking

IEEE EXPERT
INTELLIGENT SYSTEMS
& THEIR APPLICATIONS

their interactions in all rules. Furthermore, Nynex needs to tune the parameters differently at many sites, making the task particularly time-consuming. Our search method tunes parameters to bring Max's behavior closer to that of experts. This automated approach reduces the time required of those who maintain the system. Furthermore, it tunes Max in a manner that is consistent with its current implementation; it does not modify the Max system itself.

References

1. H. Rabinowitz et al., "Nynex Max: A Telephone Trouble Screening Expert System," in *Innovative Applications of Artificial Intelligence 3*, R. Smith and C. Scott, eds., AAAI Press, Menlo Park, Calif., 1991, pp. 213–230.
2. E. Wolin, *Proc. Second Int'l Conf. Information and Knowledge Management*, 1993, p. 729.
3. F.J. Provost and A.P. Danyluk, "Learning from Bad Data," in "Working Notes for Applying Machine Learning in Practice: A Workshop at the 12th Int'l Machine Learning Conf.," Tech. Report AIC-95-023, Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, Washington, D.C., 1995.
4. J.R. Quinlan, "Induction of Decision Trees," *Machine Learning*, Vol. 1, No. 1, 1986, pp. 81–106.
5. D. Rumelhart, G. Hinton, and R. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, D. Rumelhart and J. McClelland, eds., MIT Press, Cambridge, Mass., 1986, pp. 318–362.
6. A.P. Danyluk and F.J. Provost, "Small Disjuncts in Action: Learning to Diagnose Errors in the Local Loop of the Telephone Network," *Proc. 10th Int'l Machine Learning Conf.*, Morgan Kaufmann, San Francisco, 1993, pp. 81–88.
7. M. Pazzani and D. Kibler, "The Utility of Knowledge in Inductive Learning," *Machine Learning*, Vol. 9, No. 1, June 1992, pp. 57–94.
8. D. Ourston and R. Mooney, "Theory Refinement Combining Analytical and Empirical Methods," *Artificial Intelligence*, Vol. 66, No. 2, 1994, pp. 311–344.
9. M. Pazzani and C. Brunk, "Finding Accurate Frontiers: A Knowledge-Intensive Approach to Relational Learning," *Proc. Nat'l Conf. Artificial Intelligence*, AAAI Press, Menlo Park, Calif., 1993, pp. 328–334.
10. D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Mass., 1989.
11. P. van Laarhoven and E. Aarts, *Simulated Annealing: Theory and Applications*, Kluwer Academic, Boston, 1989.
12. C. Merz and M. Pazzani, "Parameter Tuning for the Max Expert System," *Proc. Sixth Int'l Conf. Tools with Artificial Intelligence*, IEEE CS Press, Los Alamitos, Calif., 1994, p. 12.

Christopher J. Merz is a PhD student in the Department of Information and Computer Science at the University of California, Irvine. His primary research interests are machine learning and neural networks. He received a BS in computer science from the University of Missouri, St. Louis, and an MS from the University of Missouri, Rolla. Readers can contact Merz at the ICS Department, Univ. of California, Irvine, CA 92717; cmerz@ics.uci.edu.

Michael J. Pazzani is a professor and chair of the Department of Information and Computer Science at the University of California, Irvine. He received a PhD in computer science from the University of California, Los Angeles, in 1987 and his MS and BS in computer science from the University of Connecticut in 1980. He serves on the editorial board for the *Machine Learning J.* and the *J. Artificial Intelligence Research*. He is the author of *Learning Causal Relationships* (Lawrence Erlbaum Associates, 1990). Readers can contact Pazzani at the ICS Department, Univ. of California, Irvine, CA 92717; pazzani@ics.uci.edu; http://www.ics.uci.edu/~pazzani.

Andrea Pohoreckyj Danyluk is an assistant professor of computer science at Williams College. From Nov. 1990 to June 1994, she was an employee of Nynex Science and Technology. Her research interests are in machine learning, particularly in the analysis of the effects of data error on learning algorithms. She received a PhD in computer science from Columbia University in 1992 and an AB from Vassar College in 1984. She is a member of the AAAI, ACM, and IEEE. Readers can contact Danyluk at the Dept. of Computer Science, Williams College, Bronfman Science Center, Williamstown, MA 01267; andrea@cs.williams.edu.