Automation of PID Autotuner Design for Complex Systems

G. Zhou and J. D. Birdwell

Dept. of Electrical and Computer Engineering
The University of Tennessee
Knoxville, TN 37996-2100, U. S. A.

Abstract

This paper describes a method to automatically design autotuners for control systems using simulated annealing and decision trees. Simulated annealing is used to select a set of tuning rules that optimize an objective function defined by a set of system performance criteria over the controller's parameters. Simulated annealing updates its intelligence to select a best tuning rule for a given circumstance. The training set is then processed by a machine learning algorithm to produce a decision tree. This decision tree will be the autotuner for a set of plants. Since the procedure starts with a plant that is a black box, it is applicable to any type of system. The autotuner is intended for use with systems of moderate complexity and is evaluated by tuning controllers for two plants with nominal models but uncertain parameters. The intended application is to groups of similar plant/control loops in process control system which must now be periodically hand-tuned.

Keywords: Artificial Intelligence, Automation, Autotuner, Decision Tree, Intelligent Control, Machine Learning, PID, Simulated Annealing

1 Introduction

This paper describes a method to automatically design autotuners for control systems, particularly proportional-integral-derivative (PID) controllers, using simulated annealing (SA) and an inductive inference machine learning method [1].

A predefined set of modification rules to change the parameters of a PID controller (percentage increase or decrease of a parameter) are specified before designing of the autotuner. The percentage can be specified according to the nature of the system.

The performance measurement of the closed loop system is specified before the designing stage. The performance is a numeric representation of the system performance. It can be a minimum integrated error or a weighted vector of system measurements. The objective of the Autotuner is to adjust the controller parameters to optimize performance.

A simulation plant is chosen to be the real-world plant which will be using the autotuner. The characteristics of the plant is unknown to the simulated annealing process. But some outputs from the system at different situations are provided to the process. Noises are also added to the testing plant to evaluate the robustness of the autotuner.

The simulated annealing method selects a modification rule and the performance improvement by measuring the closed loop output of the system. The objective of the SA process is to minimize an objective function defined by a set of system performance criteria over the controller's parameters.

A set of initial controllers that can maintain the closed loop stability is chosen at the beginning of the SA process. There is a probability density function associated with the modification rule set. The probability is the likelihood that a particular rule will be selected to modify the controller. The probability of

each rule being selected is uniform for an initial controller and is modified during successive steps.

During the SA process, modifications are made by the SA optimization procedures applied to the selected set of initial controllers. Each application of the SA optimization procedure makes a number of modifications to an initial control system. At each modification step, a modification rule is randomly selected from the pre-selected set based on the probability density function. An effective modification is one which improves the objective function. When a modification rule is effective, the probability that the applied rule will be selected for future modification steps is increased. A bad modification is defined as a tuning action that has caused system performance to deteriorate. In this case, the probability that the applied rule will be selected for future modification steps is decreased.

A set of example systems is extracted randomly from a class of systems that can be represented using the nominal model with parameters ranging within the uncertain scope. A set of initial controllers is chosen for each example system. A set of training examples is automatically generated from a simulated annealing optimization process applied to all the system/controller pairs. The training examples are represented with encoded data which are quantitative representations of the system performance with specific controllers.

The example controller/system pairs along with the associated effective tuning rules are collected. The attributes of each example include the current controller parameters, quantitative representations of the system performance, and the effective rule used to tune the system. All the example systems are used to construct the decision tree autotuner.

Training examples collected during the simulated annealing process are further processed by the machine learning algorithm to construct the autotuner, which is a decision tree. The DT will take a set of input in order to make a decision, that is to select a rule for a closed loop operating point. The input is the output measurements of the closed loop system performance.

The constructed decision tree functions as an expert system which works by evaluating the system performance and automatically making modifications to the controller.

The most salient feature of the proposed method is that it does not require human experience on the tuning process; neither does it force the system to operate near the critical point [3, 5]. Tuning rules are obtained by automatically learning from perturbed systems. Data are collected by testing a set of tuning rules with these selected example systems. The test procedures are optimized by the SA procedures. The SA optimization process is a stochastic learning process that starts with ignorance of the system and effects of the controller modifications on the system performance. Perturbations to the control systems are restricted to maintain closed loop stability. Both the learning procedure and the application procedure optimize the same objective function. A nominal system model is used, and each of the parameters is restricted to a certain range. The constructed decision tree autotuner is intended for application to the class of systems defined by the nominal model and uncertainty structure. The method is intended to be used to construct autotuners for systems of moderate complexity or when other existing methods are inapplicable. The intended application is to groups of similar plant/control loops in process control system which must now be periodically hand-tuned.

This paper is organized as follows. Section 2 describes the objective function and the pre-selected modification rules defined for the autotuner. Section 3 describes the simulated annealing process designed to generate the training set and illustrates its application to two nominal plant models. Section 4 describes how the training examples are generated using plausible perturbations to the nominal models and how the decision tree works. Section 5 describes experimental results using the autotuner. Section 6 summarizes this research with a discussion of the merits and restrictions of this approach.

2 Objective and Rules of the Autotuner

The objective function of the autotuner is defined as the integral of the absolute error signal:

$$\text{Obj} = \int_0^T \mid e(t) \mid dt \tag{1}$$

where the error $e(t)$ is the difference between an ideal response and the actual response. The input is a set point change followed by a noise signal that is a weighted telegraph signal, and T is a specified time period.

The PID controller used in this research has the form:

$$\text{C}(s) = (k_p + k_d s + \frac{k_i}{s})(\frac{1}{\tau_f s + 1}) \tag{2}$$

where k_p, k_i, k_d, and τ_f are the proportional gain, integral gain, derivative gain, and the filter time constant, respectively. The four parameters are to be tuned. Tuning rules for the autotuner are defined to modify the four controller parameters; typical rules are exhibited in Table 1. The first rule is an alarm rule. The intent is to fire this rule whenever the system becomes

Table 1: Typical modification rules for the autotuner

Rule #	Para.	% of change	Action
0	-	-	Turn off
1	k_p	-15	Cont.
2	k_i	-15	Cont.
3	k_d	-15	Cont.
4	τ_f	-15	Cont.
5	k_p	+15	Cont.
...	Similar rules for k_i, k_d, and τ_f	Different percentage of changes	Cont.
25	-	no-change	Cont.

unstable or could become unstable. Corrective actions take place when the emergency alarm rule fires. In an industrial application many alarm rules would probably be used. The *no-change* rule is designed to fire whenever the system is working in a desirable manner. Other rules are fired automatically based upon current system performance and controller parameters until satisfactory performance is achieved. In Table 1, column 1 lists numbers assigned to the rules, column 2 lists the controller parameter to be modified, column 3 lists the percentage of changes to be made to the particular parameters, and column 4 is the action to be taken after the rule is fired, where *Cont.* means the autotuner continues to monitor the system performance and make modifications, and *Turn off* means the controller needs to be disconnected.

3 The Simulated Annealing Optimization Procedures

The method updates the controller parameters and computes the probability density function which governs the selection of the next modification rule. The rate of annealing, α, is defined as the rate at which the probability of acceptance of bad modifications is decreased. The probability of acceptance at the n^{th} step is $\exp(-\alpha n)$. The intent of this nonzero and monotonically decreasing rate is to avoid being caught in local minima found by the procedure, especially in the early stages. The total number of modifications allowed is limited by a number n_{max}. The method is described in more detail in [3].

The simulated annealing procedure is illustrated by tuning initial controllers for two 5^{th} order systems. The two systems have the following nominal models:

$$P_1(s) = \frac{s^2 + 30s + 300}{s^5 + 10s^4 + 45s^3 + 120s^2 + 200s + 300} \quad (3)$$

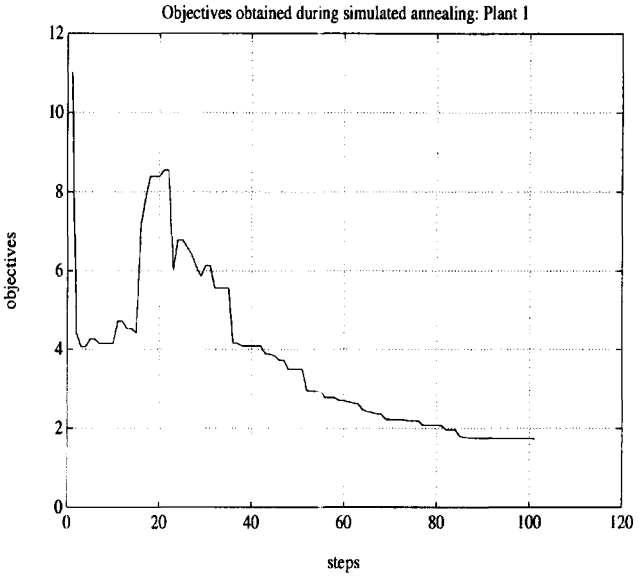

Figure 1: Objectives obtained during SA optimization: Plant #1

$$P_2(s) = \frac{s^2 + 5s + 15}{s^5 + 10s^4 + 45s^3 + 122s^2 + 199s + 145} \quad (4)$$

Plants $P_1(s)$ and $P_2(s)$ will be referred as "Plant #1" and "Plant #2", respectively.

An initial controller is randomly generated for each of the above two plants. The simulated annealing optimization method is used to find an optimum controller for each of the two plants, starting from the chosen initial controller. The objective is to minimize the objective function defined in Section 2. In both of the simulated annealing procedures, the annealing rate α is 0.05, and the maximum number of modifications n_{max} made for the initial controller is 100. The desired objective, obj_{des}, is 0.5. The initial probability density function of the rule set is uniform. The results of the optimization for Plant #1 is shown in Figure 1. The results of the optimization for Plant #2 is shown in Figure 2.

Figures 1 and 2 demonstrate the objectives obtained during the 100 successive modifications guided by simulated annealing for the two plants, respectively. It is observed that the objective function value decreases substantially for both procedures. It is also observed that in the early stages bad modifications are accepted more often than in later stage.

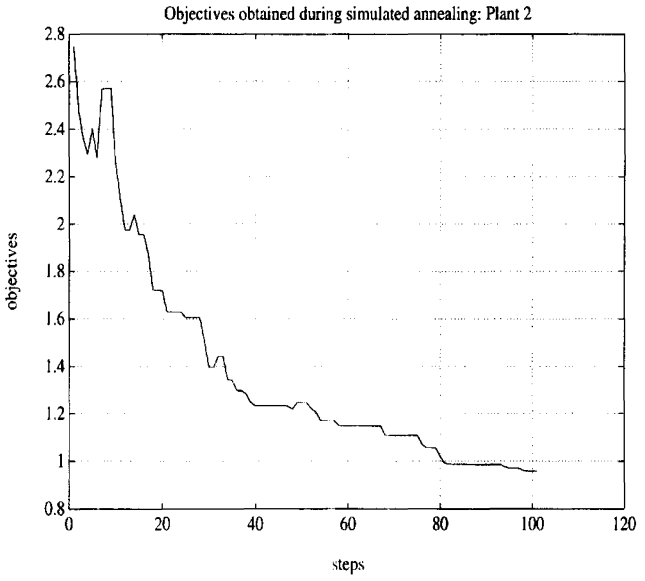

Figure 2: Objectives obtained during SA optimization: Plant #2

4 Training Examples and the Decision Tree

The training examples for the machine leaning method are generated by the simulated annealing procedures. The simulated annealing procedure is applied to a set of randomly selected initial controllers for each of the two test plants. The nominal models of the two plants can be represented by the following form:

$$P(s) = \frac{s^2 + b_1 s + b_0}{s^5 + a_4 s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0} \qquad (5)$$

In order to design an autotuner to work with uncertainties of the plant, the nominal parameters, b_0, b_1, a_0, a_1, a_2, a_3, and a_4, are perturbed in a random fashion during the training process. The uncertainties of parameters are restricted by assuming the actual value of the parameter is within 15% of the nominal value.

For the nominal plant, n perturbations are made. This leads to n systems:

$$\{(A_i, B_i, C_i, D_i)\}_{i=1}^{n} \qquad (6)$$

where A_i, B_i, C_i, D_i are the parameter matrices if the system is described by the following state equations:

$$\dot{\mathbf{x}} = A_i\mathbf{x} + B_i\mathbf{u} \qquad \mathbf{y} = C_i\mathbf{x} + D_i\mathbf{u} \qquad (7)$$

For each of the above perturbed systems, m initial controllers are randomly chosen that maintain the closed-loop stability. The system and controller pairs are represented as follows:

$$\{(A_i, B_i, C_i, D_i), \{\mathrm{PID}_{ij}\}_{j=1}^{m}\}_{i=1}^{n} \qquad (8)$$

Each of the above system/controller pair is modified by applying the simulated annealing procedure. If the maximum number of modification steps of the simulated annealing is n_{max}, n_s ($\leq n_{max}$) system/controllers are obtained for each optimization process. During the modification steps guided by the simulated annealing procedure, all the good modifications are recorded and encoded to construct the training set. The attributes of each training example include the controller parameters and the quantitative representations of the system performance. A decision tree is created based on the training set.

The decision tree is the autotuner to be applied to the system. After the decision tree is constructed, the simulated annealing procedure will no longer be used to modify the controllers. The decision tree will provide tuning rules to modify the parameters of the controller successively until satisfactory performance is achieved or a maximum specified number of modification steps have been processed. The input to the decision tree is a vector whose parameters are values of the current controller parameters and the performance indices obtained by this controller. The output of the decision tree is a tuning rule to be applied to the system. The decision tree works from its root node and goes to one of the partitions by comparing a particular attribute with a threshold until a leaf node is reached. The leaf node gives a class label which is the rule label in the modification rule table described by Table 1.

5 Experimental Results

The decision trees constructed from the training examples are evaluated by tuning a set of initial controllers for the plants. The set of initial controllers are selected by the same random generator that generated training examples for the machine learning method.

The autotuner continues the tuning process until a desired value of the objective function is reached or the maximum number of allowable tunings is reached. In this experiment, the desired objective value obj_{des} is 1.0 and the maximum number of tuning steps is 50.

Figures 3 and 4 show the system responses before and after the application of the autotuner to the initial controllers for Plant #1, and Plant #2, respectively.

To compare the results with widely used available methods, the Ziegler-Nichols' tuning [5] method and Murrill's ITAE method are used to design two PID

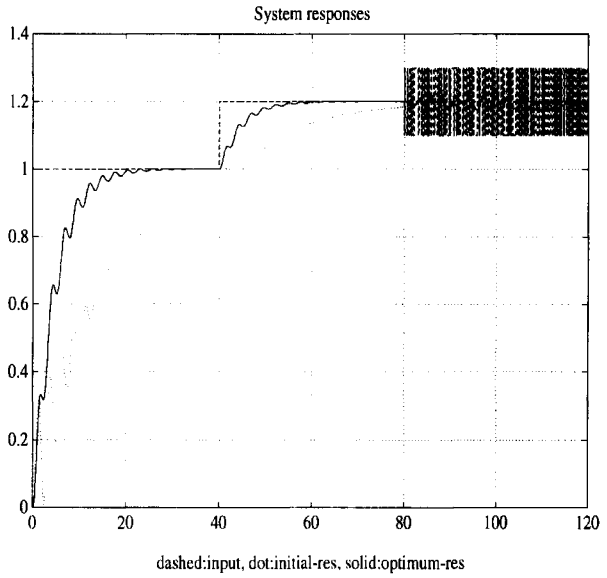

Figure 3: Performance before and after application of the autotuner, Plant #1

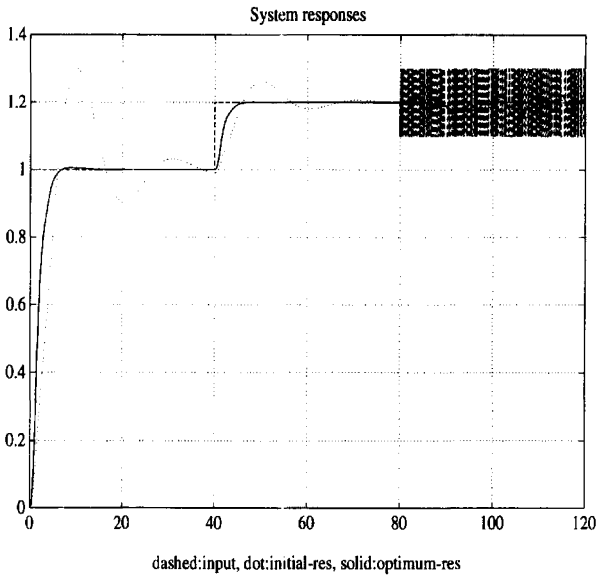

Figure 4: Performance before and after application of the autotuner, Plant #2

controllers for the systems. The ITAE method is inapplicable to Plant #1. The results of application of the decision tree autotuner, ITAE method, and Ziegler-Nichols' tuning method for Plant #2 is shown in Figure 5.

To evaluate the decision tree autotuner for the application of more systems, the following experiments are performed: (1) application of the autotuners to a set of 100 randomly generated initial controllers for the two nominal Plant #1 and Plant #2, referred as Nplant #1, and Nplant #2, respectively. (2) application of the autotuner to a set of 100 randomly generated initial controllers for the two systems when the parameter uncertainty ranges are 15%. The systems are referred as Uplant #1 and Uplant #2, respectively.

To evaluate this approach for possible wider applications, the following tree switching experiments are performed: (1) The decision tree autotuner constructed for Nplant #2, referred as NTree #2, is applied to NPlant #1, and the decision tree autotuner constructed for NPlant #1, referred as NTree #1, is applied to NPlant #2. (2) The decision tree autotuner constructed for UPlant #2, referred as UTree #2, is applied to UPlant #1, and the decision tree autotuner constructed for UPlant #1, referred as UTree #1, is applied to UPlant #2.

The experimental results are all shown in Table 2. In the table, the category "success" is defined as tunings that lead to the desired objective within the specified number of modification steps, the category "improved" is defined as tunings that lead to improvement of the system performance, the category "unchanged" is defined as tunings that leave the value of the objective function unchanged after the allowable maximum number of tunings, and the category "worse" is defined as tunings that increase the value of the objective function after the allowable maximum number of tunings.

Table 2: Table of experimental results

	Suce.	Impr.	Unch.	Wors.
NPlant#1	66	31	3	0
Nplant#2	90	10	0	0
Uplant#1	56	38	6	0
Uplant#2	88	5	7	0
NTree#2 to NPlant#1	57	36	7	0
NTree#1 to NPlant#2	75	15	10	0
UTree#2 to UPlant#1	50	36	14	0
UTree#1 to UPlant#2	77	11	12	0

The systems being controlled are generated by randomly perturbing the parameters of two nominal plant models. From the experimental results it is shown that decision tree autotuners constructed in this way can be applied to a broad class of systems.

6 Summary

Experimental results show that this approach is able to construct autotuners for some specific processes. The parameters of the processes are allowed to change within a certain range. For systems whose closed loop

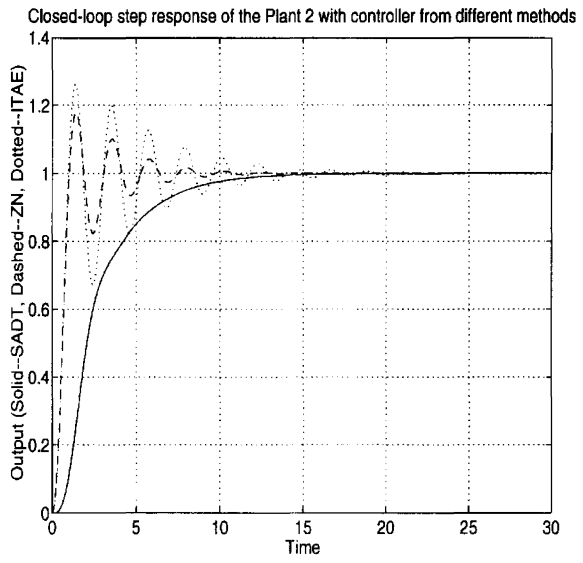

Figure 5: Comparison of the three methods for Plant #2

dynamics with specific controllers can be measured and whose controllers can be modified, the training set can be obtained by direct application of the simulated annealing procedure to the controlled plant. The autotuner design process does not require an exact system model or other knowledge about the system.

Data are recorded from the measured system performance changes caused by the applied controllers and modifications made to the controllers. For systems that can not be disturbed on line and for which simulated annealing process is not suitable, other types of data collected during normal operations can be used as a training set to construct the autotuner. For systems whose nominal model and for which the uncertainty ranges of the parameters are available, simulation of the design process similar to the procedures described by this paper can be used to construct an autotuner. For systems with some available models, but with no reliable parameter ranges available, the method described in [2] can be used to construct autotuners by computer simulation.

Since the number of the controller parameters and the forms of the controllers are not a crucial factor to this approach, the method can be applied to tune other types of controllers with few modifications. However, when the controller has a large set of modifiable parameters, and the process has a large set of changing parameters, the size of the decision tree autotuner can be undesirably large, and tuning results will be affected.

Acknowledgements

This material is based upon research work supported by The University of Tennessee, College of Engineering, Measurement and Control Engineering Center.

References

[1] Quinlan, J. R., "Induction of decision trees," *Machine Learning*, Vol. 1, pp. 81-106, 1986.

[2] Zhou. G. and J. D. Birdwell, "PID autotuner design using machine learning," *Proceedings of the 1992 IEEE Symposium on Computer-Aided Control System Design*, pp. 173-179, March, 1992.

[3] Zhou, G., *Automatic Design of Autotuners for PID Controllers*, Ph.D. Dissertation, Dept. of Electrical and Computer Engineering, The University of Tennessee, Knoxville, August, 1993.

[4] Zhou, G., and J. D. Birdwell, "Fuzzy logic-based PID autotuner design using simulated annealing", *Proceedings of the 1994 IEEE/IFAC Symposium on Computer-Aided Control System Design*, pp. 67-72, March, 1994.

[5] Ziegler, J. G. and N. B. Nichols, "Optimum settings for automatic controllers," *Trans. ASME, Vol.* **65**, pp. 433-444, 1943.