

# Genetic Algorithms for Automated Tuning of Fuzzy Controllers: A Transportation Application

Piero P. Bonissone, Pratap S. Khedkar, Yuto Chen

Information Technology Laboratory  
General Electric Corporate Research and Development,  
K1 5C32A, 1 Research Circle, Schenectady, NY 12309, USA  
email: bonissone@crd.ge.com

## Abstract

*We describe the design and tuning of a controller for enforcing compliance with a prescribed velocity profile for a rail-based transportation system. This requires following a trajectory, rather than fixed set-points (as in automobiles). We synthesize a fuzzy controller for tracking the velocity profile, while providing a smooth ride and staying within the prescribed speed limits. We use a genetic algorithm to tune the fuzzy controller's performance by adjusting its parameters (the scaling factors and the membership functions) in a sequential order of significance. We show that this approach results in a controller that is superior to the manually designed one, and with only modest computational effort. This makes it possible to customize automated tuning to a variety of different configurations of the route, the terrain, the power configuration, and the cargo.*

## 1. Introduction

### 1.1. Problem Description

We propose a system, composed of a *cruise planning* and a *cruise control* module, that will automate the controls of a freight train. This system will be applicable during most of the train journey, except for the initial and final transients, i.e. the train starting and stopping. In this paper we will focus on using a GA to improve the performance of the on-line controller.

A freight train consists of several hundred heavy masses connected by couplers. Each coupler may have a dead zone and a hydraulically damped spring. This implies that the railcars can move relatively to each other while in motion, leading to a

train that can change length by as much as 50–100 feet. Handling of the locomotive controls has a direct effect on these inter-car coupler dynamics and the forces and distances therein (which are termed *slack*). Couplers are subjected to two types of forces which may lead to breakage of the coupler, the brake pipe, and the train – *static* forces, which exceed a certain threshold, and *dynamic* forces, such as impulse impacts, which may snap the coupler. In addition, violation of speed limits and excess acceleration/breaking may also lead to derailments and severe cargo damage. Smooth handling while following a speed target is therefore absolutely imperative. Usually this handling is provided by an experienced train engineer.

### 1.2. Summary of Approach

This paper focuses on the module responsible for on-line tracking of the externally supplied speed profile to drive the train smoothly and without violating the speed constraints. The profile is the reference provided according to railroad operating rules and requirements. The controller uses a fuzzy logic PI to minimize tracking error, while providing a smooth ride and insuring that no part of the train<sup>1</sup> will exceed the posted speed limit. The Fuzzy Proportional Integral (FPI) controller uses the tracking error and its change-in-error to recommend a change in the control output. This is used to modify the current control settings if feasible. The control outputs are the notch and brake settings (which control the acceleration of the train). If allowed, the FPI will track the reference profile accurately.

The computation of the error used by the FPI

<sup>1</sup>It is important to remember that the typical freight train can be as long as 2.0 miles and requires up to four locomotives to pull it.

incorporates a lookahead to properly account for the train inertia. The algorithm predicts the future velocity of the train and incorporates not only the current error, but also the future predicted error, since the future reference speed is known from the profile. Finally, the PI is tuned (off-line) using GA's that modify the controllers most sensitive parameters: scaling factors (SF) and membership distributions (MF).

This system has multiple purposes: providing a certain degree of train handling uniformity across all crews, enforcing railroad safety rules (such as insuring that the posted speed limits are never exceeded by any part of the train), maintaining the train schedule within small tolerances, operating the train in efficient regimes, and maintaining a smooth ride by avoiding sudden accelerations or brake applications. This last constraint will minimize damage due to poor slack-handling, bunching, and run-in.

In the next section we will summarize related work in the problem domain, and prior attempts of tuning Fuzzy Controllers (FCs) via GAs. Then, we will describe our system's architecture, the tuning of the FC, and some selected results. Finally, we will conclude with an assessment of the current system and possible future research.

## 2. Prior Work

### 2.1. Prior Work: Problem Domain

As described above, the handling of freight trains involves a multi-body problem and proper slack management, without sensors for most of the state. This leads to a much more complex problem, which cannot be solved by the simpler schemes used by the cruise controllers for other vehicles, such as cars, trucks, boats etc.

Current locomotives are equipped with a very simplistic cruise control that uses a linear Proportional Integral (PI) controller, which can be used only below speeds of 10 mph. This PI controller is primarily meant to be used for uniform loading, yard movement etc. and does not prescribe braking action. Furthermore, the technology used does not consider slack or distributed dynamics in any way, and is inappropriate for extended trains at cruising speeds over general terrain. To test and tune our fuzzy controller, we have used a simulator developed in-house, based on work done at GE and the Association of American Railroads.

### 2.2. Prior Work: GA and Fuzzy

Many researchers have explored the use of genetic algorithms to tune fuzzy logic controllers. Reference [3] alone contains an updated bibliography of 295 papers combining GA with fuzzy logic, of which at least half are specific to the tuning and design of fuzzy controllers by GAs. For brevity's sake we will limit this section to a few contributions. These methods differ mostly in the order or the selection of the various FC components that are tuned (termsets, rules, scaling factors).

Early work in this area was due to C. Karr [6, 5, 7], who uses GAs to modify the membership functions in the termsets of the variables used by the FCs. Karr uses a binary encoding to represent three parameters defining a membership value in each termset. The binary chromosome is the concatenation of all termsets. The fitness function is a quadratic error calculated for four randomly chosen initial conditions.

Herrera, Lozano, and Verdegay [4] directly tune each rule used by the FC. They use a real encoding for a four-parameter characterization of a trapezoidal membership value in each termset. Each rule is represented by the concatenation of the membership values used in the rule antecedent (state vector) and consequent (control action). The population is the concatenation of all rules so represented. A customized (max-min arithmetical) crossover operator is also proposed. The fitness function is a sum of quadratic errors.

Kinzel, Klawon and Kruse [8] tune both rules and termsets. They depart from the string representation and use a (cross-product) matrix to encode the rule set (as if it were in table form). They also propose customized (point-radius) crossover operators which are similar to the two-point crossover for string encoding. They first initialize the rule base according to intuitive heuristics, use GAs to generate better rule base, and finally tune the membership functions of the best rule base. This order of the tuning process is similar to that typically used by self-organizing controllers [2].

Lee and Takagi also tune the rule base and the termsets [9]. They use a binary encoding for each three-tuple characterizing a triangular membership distribution. Each chromosome represents a TSK-type rule [11], concatenating the membership distributions in the rule antecedent with the coefficients of the linear consequent. Also of interest to the reader is the approach taken by Surman, Kanstein, and Goser [10], who modify the usual quadratic fitness function by addition an entropy term describing the number of activated rules.

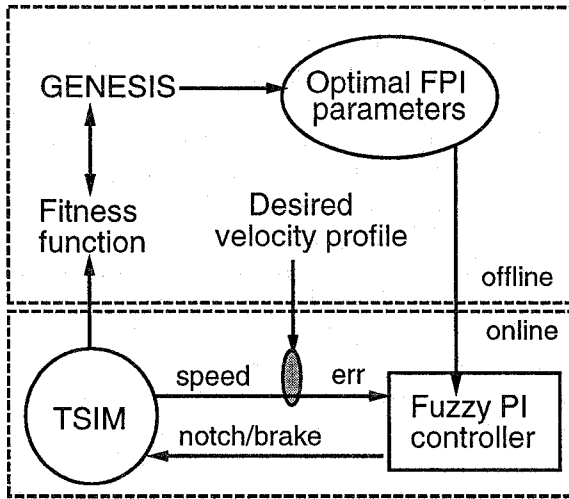


Figure 1. System schematic for using a GA to tune a fuzzy train controller.

In our approach, which will be described later, we follow the tuning order suggested by Zheng [12] for manual tuning. We begin with macroscopic effects, by tuning the FC state and control variable *scaling factors*, while using a standard uniformly spread termset and a homogeneous rule base. After obtaining the best scaling factors, we proceed to tune the *termsets*, causing medium-size effects. Finally, if additional improvements are needed, we tune the *rule base* to achieve microscopic effects.

This parameter sensitivity order can be easily understood if we visualize a homogeneous rule base as a rule table: a modified scaling factor affects the entire rule table; a modified term in a termset affects one row, column, or diagonal in the table; a modified rule only affects one table cell.

### 3. System Architecture

The overall scheme for the proposed train handling is shown in Figure 1. It consists of an FPI closing the loop around the train simulation (TSIM), using only the current velocity as its state input. This speed and lookahead are compared with the desired profile to generate a predicted near-term error as input to the FPI, which outputs control actions back to the simulator. Offline, a GA uses the setup for evaluating various FPI parameters.

The simulator — TSIM, is an in-house implementation, combining internal data with physical/empirical models as described in TEM (Train Energy Model) [1]. TEM was developed by the As-

sociation of American Railroads. TSIM simulates an extended train with arbitrary detailed makeup, over a specified track profile. The version of TSIM used here does not simulate a stretchable train for the sake of computational efficiency.

For the purpose of this study, GENESIS (GENetic Search Implementation System) has been adopted as a software development tool. It has been developed by John J. Grefenstette to promote the study of genetic algorithms for function optimization. The user must provide only a “fitness” function which returns a corresponding value for any point in the search space.

#### 3.1. Fitness Functions

To study the effects of different objectives, we minimize three fitness functions :

$$\begin{aligned}
 f_1 &= \sum_i |notch_i - notch_{i-1}| \\
 &\quad + |brake_i - brake_{i-1}| \\
 f_2 &= \sum_i |v_i - v_i^d| \\
 f_3 &= w_1 \frac{\sum_i |notch_i - notch_{i-1}|}{K_1} \\
 &\quad + w_2 \frac{\sum_i |v_i - v_i^d|}{K_2}
 \end{aligned}$$

$v^d$  denotes the desired velocity and  $i$  is a distance or milepost index.  $f_1$  captures throttle jockeying,  $f_2$  captures speed profile tracking accuracy, and  $f_3$  combines a weighted sum of the two.

#### 3.2. Fuzzy PI Controller

A fuzzy logic PI controller is described by a Knowledge Base (KB) composed of a *rule set*, *termsets*, and *scaling factors* (SF). The rule set maps linguistic descriptions of state vectors  $[e, \Delta e]$  into incremental control actions  $\Delta u$ ; the termsets define the semantics of the linguistic values used in the rules; and the scaling factors determine the extremes of the numerical range of values for both the input and output variables.

The relationship between output variable,  $u$ , and control error,  $e$ , can be expressed approximately as follows [12]:

$$\begin{aligned}
 \frac{\Delta u(t)}{S_u} &\approx \frac{\Delta e(t)}{S_d} + \frac{e(t)}{S_e} \\
 u(t) &\approx \frac{S_u}{S_d} \cdot e(t) + \frac{S_u}{S_e} \cdot \int e(t) \\
 &\quad -S_e \leq e(t) \leq S_e \\
 &\quad -S_d \leq \Delta e(t) \leq S_d \\
 &\quad -S_u \leq \Delta u(t) \leq S_u
 \end{aligned}$$

where  $S_e$ ,  $S_d$  and  $S_u$  are the scaling factors of error, change of error, and incremental output variable, respectively.

On the other hand, a conventional PI controller has  $u(t) = K_p e(t) + K_i \int e(t) dt$  where  $K_p$  and  $K_i$  are the proportional and integral gain factors, respectively. Comparing the above equations, we obtain

$$K_p \approx S_u/S_d, \quad K_i \approx S_u/S_e \cdot (1/dt)$$

## 4. Tuning the Fuzzy PI

### 4.1. Testbed and Design Choices

For comparison purposes, twelve tests have been conducted by taking a cross-product of : the scaling factor values before and after GA tuning, the membership function parameter values before and after GA tuning, and the 3 fitness functions. The tests are designed to demonstrate that GAs are powerful search methods and are very suitable for automated tuning of FPI controllers. GAs are able to come up with near-optimal FPI controllers within a reasonable amount of time according to different search criteria. In addition, the tests are also designed to demonstrate that we should tune parameters in the order of their significance. That is, we should tune scaling factors first since they have global effects on all the control rules in a rule base. Tuning membership functions will only give marginal improvements for a FPI with tuned scaling factors. In the following, we present testbed set-up and design choices in brief.

• **Train Simulator Parameters:** All testing for the automated tuning of FPI was done using TSIM. Two track profiles were used: an approximately 14 mile flat track and an approximately 40 mile piece of actual track from Selkirk to Framingham over the Berkshires. The train was nearly 9000 tons, and about a mile long, with 4 locomotives and 100 loaded railcars. An analytically computed velocity profile which minimizes fuel consumption was used as the reference.

• **FPI Controller Parameters:** The standard termset used in the FPI is {NH, NM, NL, ZE, PL, PM, PH }. where N = Negative, P = Positive, H = High, M = Medium, L = Low, and ZE = Zero. Initially, the terms are uniformly positioned trapezoids overlapping at a 50% level over the normalized universe of discourse. Since the controller is defined by a nonlinear control surface in  $(e, \Delta e, \Delta u)$  space, we need three termsets in all, one for each of  $e, \Delta e, \Delta u$ . The design leads to a symmetric controller, which is not always a good assumption. In this case, the

GA tuning will automatically create the required asymmetry.

• **GENESIS parameters:** The GA parameters are : Population Size = 50, Crossover Rate = 0.6, Mutation Rate = 0.001. In addition, all structures in each generation were evaluated, the elitist strategy was used to guarantee monotone convergence, gray codes were used in the encodings, and selection was rank based.

• **Tuning of Scaling Factors (SF):** Each chromosome is represented as a concatenation of three 3-bit values for the three floating point values for the three FPI scaling factors  $S_e$ ,  $S_d$  and  $S_u$ . They are in the ranges:  $S_e \in [1, 9]$ ,  $S_d \in [0.1, 0.9]$ ,  $S_u \in [1000, 9000]$ . The intent is to demonstrate a GA's ability for function optimization even with such a simple 9-bit chromosome.

• **Tuning of Membership Functions (MF):** When tuning membership functions (MFs), a chromosome is formed by concatenating the 21 parameterized MFs. Since each MF is trapezoidal and overlap degree=0.5 is maintained between adjacent trapezoids, this partitions the universe of discourse into intervals which alternate between being cores of an MF, and overlap areas. The core of *NH* and *PH* extend semi-infinitely to the left and right respectively (outside the [-1,1] interval). These intervals, denoted by  $b_i$  are 11 in number for 7 MF labels. In general,  $\#(b) = 2 \times \#(MF) - 3$ , where  $\#(b)$  is the number of required intervals and  $\#(MF)$  is the number of membership functions. Each chromosome is thus a vector of 11 floating point values. Since the universe is normalized,  $\sum_{i=1}^{11} b_i \leq 2$ . For the present study, each  $b_i$  is set within the range of [0.09, 0.18] and five bits are used to represent a chromosome for GA tuned membership functions. If  $\sum_i b_i$  exceeds 2, this implicitly reduces the number of effective MF's, thus providing partial structure optimization as well.

## 5. Simulation results

### 5.1. SF : Manually Tuned vs. GA

The GA was started with an initial set of scaling factor values, and run with each of the fitness functions  $f_1, f_2, f_3$  for the flat track and unit train. The results after four generations are shown in Table 1.

Throttle jockeying is almost eliminated in  $f_1$  by decreasing both  $K_p$  and  $K_i$  simultaneously. There are no big differences in fuel consumption between the four runs. These results will show further improvement when a more dynamic fuel consumption model which models transients is incorporated into

the simulator.

## 5.2. Tuning SF vs. MF with $f_1$

Four sets of tests were conducted for the comparison of significance in tuning scaling factors (SF) vs. tuning membership functions (MF) with respect to fitness function  $f_1$ . The results are shown in Table 2. The initial set of scaling factors is  $[S_e S_d S_u] = [5.0, 0.2, 5000]$ . After 4 generations of evolution, the set of GA tuned SF is  $[9.0, 0.1, 1000]$ . As shown in Table 2, the fitness value was dramatically reduced from 73.20 to 15.15 for the GA tuned SF with respect to  $f_1$ . Substantially smoother control results.

On the other hand, GA tuned MF only reduced throttle jockeying a little bit after 20 generations of evolution. It is indicated by the the small reduction in fitness value from 73.20 to 70.93. From the above observations, we can conclude that tuning SF is more cost-effective than tuning MF. The little improvement by tuning MF's leads to an asymmetric shift in the membership functions, such that the ones on the left of ZE get shifted more than the ones on the right. This is due to the slightly asymmetric calibration of the notch and brake actuators. A +5 (notch) may not lead to the exact tractive force forward that a -5 (brake) leads to in the opposite direction. As a result, the best FPI responds to negative error slightly differently than to positive error.

Next, we demonstrate that tuning membership functions only gives marginal improvements for a fuzzy PI controller with tuned scaling factors. we used GA to optimize FPI's MF with respect to  $f_1$ , while using the GA tuned SF values of  $[9.0, 0.1, 1000]$ . After 10 generations of evolution, the fitness value was further reduced from 15.15 to 14.64. The change in smoothness and the MF values is minimal. After observing the simulation results, we conclude that tuning membership functions alone will only provide marginal improvements for a fuzzy PI controller with fine tuned scaling factors.

## 5.3. Tuning SF vs. MF with $f_3$

We further verified our arguments stated in the last sub-section with the following four sets of tests conducted with respect to fitness function  $f_3$ , which balances both tracking error and control smoothness. The results are shown in Table 3.

Recall that the initial set of scaling factors is  $[5.0, 0.2, 5000]$ . After 4 generations of evolution, the GA came up with a set of SF  $= [3.3, 0.9, 5571]$ , with

respect to  $f_3$ . As shown in Table 3, the fitness value was reduced from 1.000 to 0.817 while doing this. On the other hand, GA tuned MF only reduced the fitness value from 1.000 to 0.942 after 20 generations of evolution, confirming the claim that tuning SF is more cost-effective than tuning MF.

We proceeded to experiment with MF tuning with tuned SF  $= [3.3, 0.9, 5571]$ . This time there were no significant improvements in fitness after 10 generations.

Table 4 summarizes all the 12 tests run on the flat track. In addition, we present the final performance graphs in Figure 2 for the more complex piece of real track with the same train. It shows substantial improvement in control accuracy and smoothness.

## 6. Conclusions and Future Work

We have presented an approach for tuning a fuzzy KB for a complex, real-world application. In particular, we used genetic algorithms to tune scaling factors as well as membership functions, and demonstrated that the approach was able to find good solutions within a reasonable amount of time under different train handling criteria. In addition, we showed that all parameters do not need to be treated as equally important, and that sequential optimization can greatly reduce computational effort by doing scaling factors first. Additional improvement was shown by the good performance of fairly coarse encodings. The scalability of the approach enables us to customize the controller differently for each track profile, though it does not need to be changed for different train makeups. In this way, we can produce customized controllers offline efficiently. Future work will also focus on automatic generation of velocity profiles for the train simulator by using genetic algorithms for trajectory optimization subject to "soft" constraints.

## References

- [1] William F. Drish, Jr. Train energy model - technical manual. Technical Report SD-040, Association of American Railroads, March 1992.
- [2] D. Burkhardt and P. Bonissone. Automated Fuzzy Knowledge Base Generation and Tuning. In *Proc. 1st IEEE Conf. on Fuzzy Systems*, pages 179-188, San Diego, CA, USA, 1992. IEEE.
- [3] O. Cordon, H. Herrera, and M. Lozano. A classified review on the combination fuzzy logic-genetic algorithms bibliography. Research Report DECSAI 95129, Department of Computer Science and AI, Universidad de Granada, Granada, Spain, 1995.

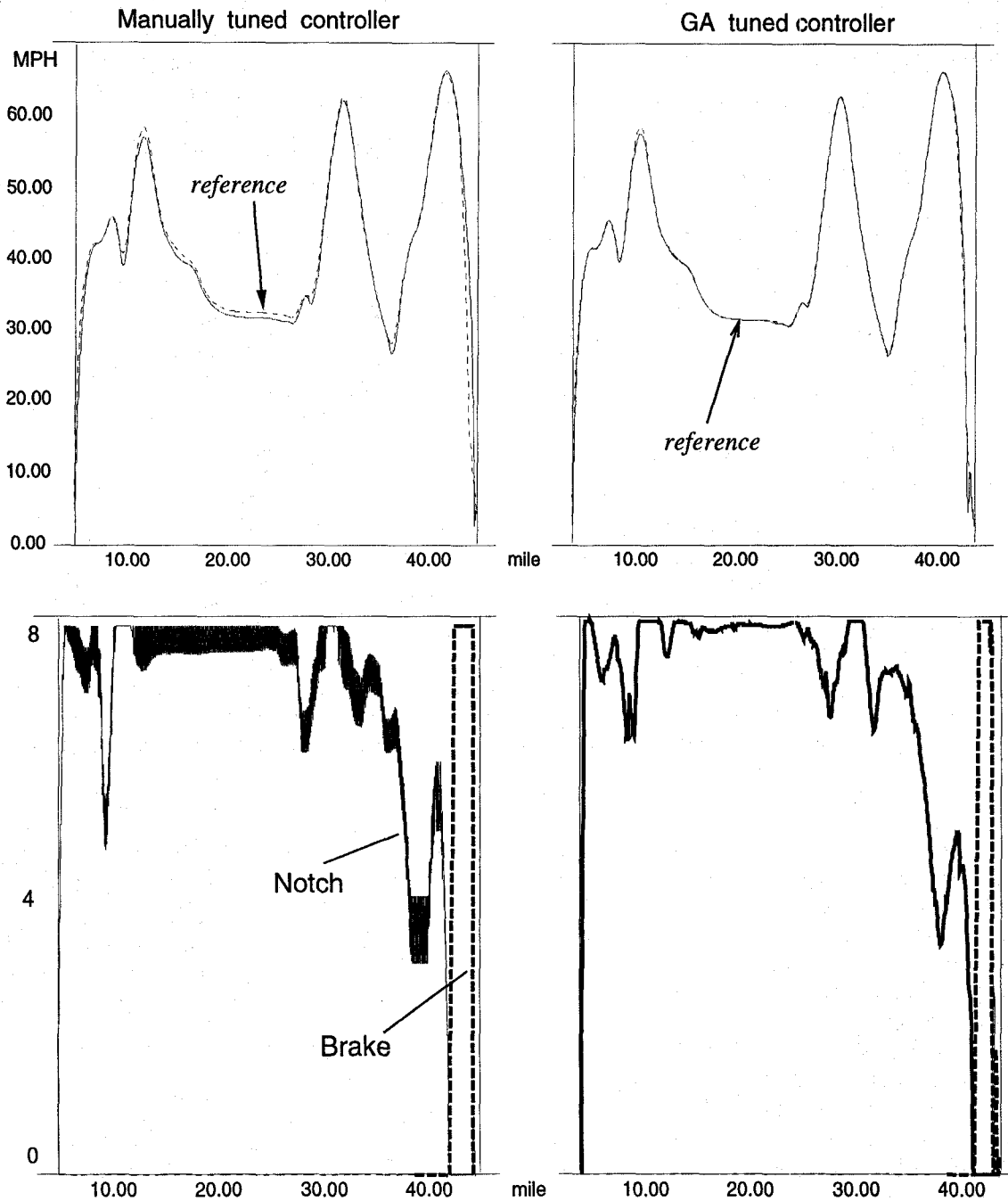


Figure 2. The two figures on the left show reference tracking performance and control outputs before GA tuning. After tuning, the two figures on the right show vastly improved tracking and smoothness of control. The track was a section of Selkirk—Framingham track, and the simulated train was the 9000 ton, mile-long, loaded unit train.

Scaling factors	Time (min)	Journey (mile)	Fuel (gal)	Fitness
Initial [5.0,0.2,5000]	26.51	14.26	878	$f_1 = 73.2, f_2 = 227, f_3 = 1$
GA wrt. $f_1$ [9.0,0.1,1000]	27.76	14.21	857	73.2 $\rightarrow$ 15.2
GA wrt. $f_2$ [6.7,0.1,4429]	25.81	14.12	875	227 $\rightarrow$ 213.1
GA wrt. $f_3$ [3.3,0.9,5571]	27.26	14.27	875	1.0 $\rightarrow$ 0.82

**Table 1. Results after 4 generations with different fitness functions.**

Description	Time	Journey	Fuel	Fitness	Generation
Initial SF, initial MF	26.51	14.26	878	73.20	0
GA tuned SF, initial MF	27.76	14.21	857	15.15	4
Initial SF, GA tuned MF	26.00	14.18	879	70.93	20
GA tuned SF, GA tuned MF	28.26	14.12	829	14.64	10

**Table 2. Results using  $f_1$  with different parameter sets.**

Description	Time	Journey	Fuel	Fitness	Generation
Initial SF w/initial MF	26.51	14.26	878	1.000	0
GA tuned SF w/initial MF	27.21	14.35	871	0.817	4
Initial SF w/GA tuned MF	26.26	14.18	871	0.942	20
GA tuned SF w/GA tuned MF	27.26	14.35	872	0.817	10

**Table 3. Results using  $f_3$  with different parameter sets.**

Description	$f_1$	$f_2$	$f_3$
Initial SF w/initial MF	73.2	227	1.00
GA tuned SF w/initial MF	15.2	213	0.82
Initial SF w/GA tuned MF	70.9	201	0.94
GA tuned SF w/GA tuned MF	14.6	204	0.82

**Table 4. Summary of all simulation results on the flat track.**

- [4] F. Herrera, M. Lozano, and L. Verdegay. Tuning fuzzy logic control by genetic algorithms. *Int. Journal Approximate Reasoning (IJAR)*, 12(3/4):299–315, 1995.
- [5] C.L. Karr. Design of an adaptive fuzzy logic controller using genetic algorithms. In *Proc. Int. Conf. on Genetic Algorithms (ICGA '91)*, pages 450–456, San Diego, CA., USA, 1991.
- [6] C.L. Karr. Genetic algorithms for fuzzy controllers. *AI Expert*, 6(2):27–33, 1991.
- [7] C.L. Karr. Fuzzy control of ph using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, 1:46–53, 1993.
- [8] J Kinzel, F. Klawoon, and R. Kruse. Modifications of genetic algorithms for designing and optimizing fuzzy controllers. In *Proc. First IEEE Conf. on Evolutionary Computing (ICEC'94)*, pages 28–33, Orlando, FL., USA, 1994.
- [9] M. Lee and H. Takagi. Integrating design stages of fuzzy systems using genetic algorithms. In *Proc. 2nd IEEE Conf. on Fuzzy Systems*, pages 612–617, San Francisco, CA, USA, March 1993.
- [10] H. Surmann, A. Kanstein, and K. Goser. Self-organizing and genetic algorithms for an automatic design of fuzzy control and decision systems. In *Proc. EUFIT'93*, pages 1097–1104, Aachen, Germany, 1993.
- [11] T. Takagi and M. Sugeno. Fuzzy Identification of Systems and Its Applications to Modeling and Control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15(1):116–132, 1985.
- [12] Li Zheng. A Practical Guide to Tune Proportional and Integral (PI) Like Fuzzy Controllers. In *Proc. 1st IEEE Conf. on Fuzzy Systems*, pages 633–640. IEEE, March 1992.