IC Engine Misfire Detection Algorithm Generation using Genetic Programming

Richard J. Hampo
Bruce D. Bryant
Kenneth A. Marko
Ford Motor Company Scientific Research Laboratory
20000 Rotunda Drive, MD 2036
Dearborn, Michigan 48121-2053 USA

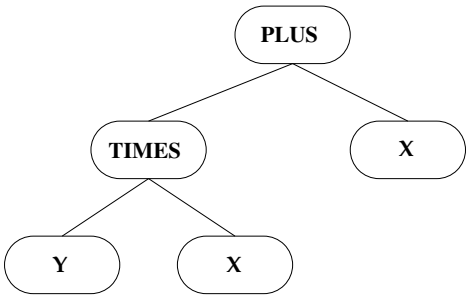
Keywords: Diagnostics, Genetic, Misfire

Misfire detection has been a topic of concern for automobile manufacturers in the United States since the California Air Resources Board (CARB) introduced legislation requiring it as a component of on-board diagnostics (OBD-II) for monitoring all vehicle systems which affect exhaust air quality. It was determined that engine misfires can accelerate deterioration of the catalytic converter - and subsequently degrade the quality of the exhaust air. The development of an efficient, accurate, and robust misfire detection algorithm has turned out to be a considerable challenge. OBD-II legislation requires that each cylinder firing be monitored for misfire over the broadest possible operating conditions. For an eight cylinder engine that runs at an average of 2000 RPM at 50 M.P.H. for 100,000 miles, over its lifetime, the misfire detector must make 960 million determinations of whether or not there was a misfire. The detector must catch as many misfires as possible, and thus minimize the type I, or false negative error. It must also minimize the number of false alarms, or type II errors. Legislation sets stringent limits on the tolerable rates of these two types of errors. The sheer number of determinations that need to be made reinforce the need for an accurate and robust detection algorithm.

A significant difficulty with misfire detection is that of instrumentation--the cost associated with instrumenting engines to directly observe combustion is prohibitive in mass production. Therefore, most practical misfire detection techniques deduce the information necessary to assess combustion from sensors already on the engine which are used for existing powertrain control and monitoring functions. These techniques generally monitor crankshaft position, intake manifold pressure, ignition voltage properties, and/or exhaust pressure. A broad spectrum of approaches to misfire detection have been discussed in the literature. These range from model-based methods which compare expected and observed torque estimations [1],[2], to empirical techniques which fit trainable or adaptable algorithms to the misfire detection task [3]. Other methods combine model-based and empirical techniques in an effort to exploit the strengths of each [4],[5]. In this paper we discuss a hybrid approach which seeks to identify an improved misfire detector by employing an empirical technique called Genetic Programming [6] to search the space of possible solutions which include permutations of the existing model-based and empirical approaches.

Genetic programming (GP) [6] is an extension of the Genetic Algorithm (GA) [7] paradigm. Essentially, both GP and GA exploit the efficient searching properties of crossover and mutation, similar to those found in evolutionary theory. GP operates on a population of candidate solutions. Individual members of the population are tested and ranked according to their performance on some fitness measure. The problem solutions that rank highest are selected to reproduce. Moreover, the probability that a solution will be selected for reproduction is proportional to its fitness. Crossover (depicted in Figure 2) drives the solutions to areas of the search space that maximize the performance of the members of the population. Mutation inserts the small amount of randomness needed to maintain genetic diversity and allow the population to continue to improve. This scheme amounts to an efficient searching of the space of possible solutions.

To use genetic programming, four decisions must be made. First, the input measurements that the solution can use must be identified. Second, a pool of possible operators, some of which may be problem specific, must be specified. These operators are the building blocks which GP will automatically combine to form solutions to the given problem. Third, a "fitness function," which measures how well each candidate so-



Result: F(X,Y) = X + (Y*X)

Figure 1: An example of a Genetic Program

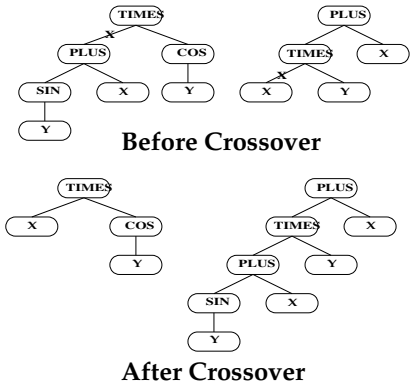


Figure 2: An example of Crossover Reproduction

lution solves the problem at hand, must be identified. Fourth, an acceptable final performance criteria must be specified, as in any empirical technique. In addition, some operational considerations must be made, such as limits on the population and maximum solution size.

The GP paradigm operates as follows: It constructs a population of candidate solutions in the form of tree-like structures whose nodes are mathematical operators and whose leaves are inputs. Figure 1 illustrates one such example "tree." This example uses the inputs X and Y, and the operators TIMES and PLUS to form a function F(X,Y) = (X TIMES Y) PLUS X. GP applies the fitness function to each candidate solution in the population, then ranks them according to fitness. GP then creates a new population by reproducing members of the original population. Reproduction is accomplished mainly through crossover, where randomly-selected branches of two candidates are swapped to yield two new candidates. Another form of reproduction is also used which directly copies members of the original population into the new population. A small percentage of the new population is generated by mutation, where members of the original population are altered slightly before being copied into the new population. The fitness function is then applied to the new population, and the process of reproduction/fitness testing is repeated until a candidate solution meets the final performance criteria.

The strength of GP over other empirical techniques such as Neural Networks, Regression Analysis, and even Genetic Algorithms, is that it is not restricted in the form of the final solution. The only limitations on the form are in the pool of available operators and inputs (and, of course, reasonable computing limitations). In addition, the fitness function and performance criteria are not limited by constraints of mathematical construction, as is the case with Neural Networks. The only limitation on the fitness function is that it must be able to objectively reflect the degree to which candidate solutions solve the problem at hand, based on the available information.

Intuition about the problem at hand is useful in choosing the pool of operators and inputs. The trees in Figures 1 and 2 use the data X and Y as inputs and the familiar arithmetic functions as operators. These operators plus some basic functions from mathematics and trigonometry are often very useful in applying GP, however, any knowledge of the form of the final solution should inspire the developer to include related functions. For example, a control system problem would be more easily solved if the derivative and integral functions were included in its pool of operators. This principle may also be applied to the selection of available inputs. For example, we know that in the misfire detection problem, certain inputs such as intake manifold pressure are used by other approaches, so we would provide these inputs to GP as well. In addition, processed input data such as crankshaft speed and acceleration have been shown to be useful in other approaches, so we also would include them in the pool of inputs. However, since each addition to the pool of available operators and inputs essentially adds another dimension to the space of possible solutions which GP must search, it is wise to limit them to only those which may help in finding a solution. Judicious choice of operators and operands is important to a successful GP implementation.

To apply GP to the misfire detection problem, the CARB imposed regulations on misfire were studied to

determine an appropriate fitness measure and termination criteria. Detector errors (number of missed calls) was chosen as the fitness function. Induced engine misfire data was already in our possession, and this data was split into three data sets; A *training* set, a *testing* set, and a set of all the *other*s. The training set was chosen to cover the speed/load plane, with some data taken at steady state, and other data taken during transient conditions. The training data was the only data used by the GP algorithm during its development. After good solutions were found, they were tested on the *testing* data and the *other* data. Based on the CARB regulations, a total error rate of 2.2 % was determined to be acceptable performance, with 2.0% being the limit on false negatives (missed calls) and 0.2% being the limit on false positives (false alarms).

For this problem, the inputs RPM (incremental crankshaft velocity), ACCEL (incremental acceleration), DACCEL (local deviation from ACCEL), and LOAD (volumetric efficiency) which are calculated from measured engine signals were chosen. These higher level inputs were chosen rather than the raw data inputs because our experience indicates that the information necessary to detect misfire is contained in them, in a much more efficient format than, for instance, mass air flow or crankshaft position versus time.

The set of operators used included ADD, SUB, MUL, DIV, SQ, and SQRT which are ordinary mathematical operations, and a three position median function, SMEDIAN. Further, since the effect of a misfire can be seen for several firings after it occurs, operators like MIN, MAX, AVE, and TMEDIAN calculate the minimum, maximum, average and median of an input over the current, previous two, and subsequent two firings. Finally, a derivative operator, FD, is used to calculate the forward derivative of an input. These operator and input selections encode our a-priori information about the problem. For instance, we know that DACCEL is, by itself, a fairly reliable misfire detector. However, we also know that it is not good enough over some engine operating conditions, namely high speed, low load. Therefore, RPM and LOAD seem necessary for misfire detection. Also, since the effects of a misfire at one firing can be seen for several engine firings later, the MIN, MAX, AVE, and TMEDIAN operators seem necessary to isolate misfires.

Initially, a fitness function which was inversely related to the number of errors each candidate solution made on a set of known misfire data was used. However, we recognized that this function did not address the issue of robustness, since, although it considered data over the entire operating range of a vehicle, we only used data from a small number of vehicles, and this data was collected at only one point in time over the vehicles' useful lives. Therefore, we augmented the fitness function to include Separation Factor (SF), the degree to which misfire calls were geometrically distinguishable from proper firing calls. Robustness is addressed in this way by reducing the potential for sensitivity of the misfire detector's accuracy to parameter variations. Separation Factor (SF) is defined as follows:

$$SF \triangleq \frac{\overline{F_{miss}(\cdot)} + \overline{F_{no\,miss}(\cdot)}}{\sigma_{F_{miss}(\cdot)} + \sigma_{F_{no\,miss}(\cdot)}}$$

$\sigma_{F_{miss}(\cdot)}\ refers\ to\ the\ standard\ deviation\ of\ the\ function\ F(\cdot)\ for$
$the\ examples\ of\ misfire.\ Similarly,\ \sigma_{F_{no\,miss}(\cdot)}\ refers\ to\ the\ standard$
$deviation\ for\ cases\ of\ no\ misfire.\ The\ overbars\ denote\ the\ mean\ value.$

In an effort to maximize accuracy and robustness, we employed several fitness functions using various combinations of the error rate and separation factor. Generally, the best performance was achieved with a fitness function that was inversely related to the number of wrong answers plus 20 times the inverse of the separation factor SF. The performance of a few of the algorithms found is detailed in the table along with the performance of some other misfire detectors for comparison. The target error rates are those for a detector that would meet the CARB requirements. Data Set #1 was used to train the Neural Network-based misfire detector and was also used in the GP fitness function. It contains data from the entire operating range of a vehicle. Data Sets #2 and #3 were not used to train the Neural Network, nor were they used in the GP fitness function. #2 contains data over the entire operating range. #3 contains data more representative of actual driving conditions, hence the better performance on this set.

The problem of diagnosing faults in complex machinery has been addressed from the standpoint of developing solutions using empirical data, guidance from experts in the field, and engineering judgement.

Sample Misfire Detection Results				
Algorithm	Errors	Type I	Type II	SF
Target Error Rates	2.20 %	2.00%	0.20 %	3.5
Results: Data Set #1 (Training)				
Neural Network Trained by Back-Propagation	0.49 %	0.093 %	0.40 %	3.15
GP # 1	0.19 %	0.036 %	0.015%	2.62
GP # 2	0.14 %	0.026 %	0.12 %	2.04
Results: Data Set #2 (Testing)				
Neural Network Trained by Back-Propagation	0.048 %	0.005%	0.043 %	4.51
GP # 1	0.024%	0.014 %	0.010 %	3.32
GP # 2	0.024 %	0.005 %	0.019%	2.03
Results: Data Set #3 (Other)				
Neural Network Trained by Back-Propagation	0.46 %	0.0 %	0.46 %	4.72
GP # 1	0.17 %	0.006 %	0.17 %	4.63
GP # 2	0.042 %	0.010%	0.042 %	3.69

Table 1: Misfire Detector Test Results

Genetic Programming has provided an excellent framework for producing a solution to a difficult problem, efficiently incorporating all available resources. Solutions provided by GP are valuable for the following reasons:

● The solutions have an explicit representation which permits direct insertion into conventional processors.

● The solutions have shown how the performance of conventional algorithms may be improved.

● Identification and reduction of the explicit form of the solutions provides an opportunity for simple parametric optimization on similar mechanical systems, and offers an opportunity for experts to "learn" from the forms of the solutions discovered by GP.

● GP offers a rational means to "benchmark" performance of diagnostic algorithms against an impartial generator of high performance algorithms.

Genetic programming applied to the diagnostic classification problem provided a viable means of obtaining explicit solutions which met the performance objectives identified. We expect that GP will become a routine tool for use in algorithm development and performance benchmarking.

References:

[1] G. Rizzoni, "Diagnosis of Individual Cylinder Misfire by Signature Analysis of Crankshaft Speed Fluctuations", SAE 890884, 1989

[2] W. Ribbens and G. Rizzoni, "Applications of Precise Crankshaft Position Measurement for Engine Testing, Control and Diagnosis", SAE 890885, 1985

[3] K. Marko, J. James, J. Murphy, J. Dosdall "Automotive Control System Diagnostics Using Neural Networks for Rapid Pattern Classification of Large Data Sets", Proceedings of the International Neural Network Conference, Washington DC, June 1989

[4] J. James, J. Dosdall, K. Marko "Misfire Detection in an Internal Combustion Engine", US patent # 5,044,194, September 1991

[5] J. James, J. Dosdall, K. Marko "Misfire Detection in an Internal Combustion Engine", US patent # 5,044,195, September 1991

[6] J. Koza "Genetic Programming" MIT Press, 1992