

Case-based Adaptation for Automotive Engine Electronic Control Unit Calibration

Chi-man VONG*

*Department of Computer and Information Science, FST, University of Macau,
Macao*

E-mail: cmvong@umac.mo, Tel.: +853-83974476, fax: +853-28838314

Pak-kin WONG

Department of Electromechanical Engineering, FST, University of Macau, Macao

E-mail: fstpkw@umac.mo

Rui-he BAO

*Department of Computer and Information Science, FST, University of Macau,
Macao*

E-mail: ma56582@umac.mo

Abstract

The automotive engine performance is greatly affected by the calibration of its electronic control unit (ECU). The method for ECU calibration is traditionally done by trial-and-error. This traditional method consumes a large amount of time and money. To resolve this problem, case-based reasoning (CBR) is employed, so that an existing and effective ECU setup can be adapted to fit another similar class of engines. The adaptation procedure is done through a more sophisticated step called case-based adaptation (CBA). The CBA is an effective knowledge management tool, which can interactively learn the expert adaptation knowledge. The paper briefly reviews the methodologies of CBR and CBA. Then the application to ECU calibration is described via a case study. With CBR and CBA, the efficiency of calibrating an ECU can be enhanced. A prototype system has also been developed to verify the usefulness of CBR in ECU calibration.

Keywords: Case-based reasoning (CBR), Adaptation, Electronic control unit (ECU) calibration

1. Introduction

Modern automotive engines are controlled by the electronic control unit (ECU). The engine performance, such as power, torque, brake specific fuel-consumption and emission level, is significantly affected by the setup of control parameters in the ECU. Parameterization of electronic control unit (ECU) software is a major milestone in the development process for modern automotive engines. This process is known as ECU calibration or ECU tune-up. Fig. 1 shows a screenshot of calibration process in an ECU software. ECU calibration is engine dependent. In other words, an ECU setup is only valid on the same engine model. Traditionally, the ECU calibration is done by the vehicle manufacturer. However, in recent years, the programmable ECU (Fig. 2) and ECU read only memory (ROM) editors have been widely adopted by many performance cars. These devices allow the non-factory engineers to tune up their engines according to different add-on components and driver's requirements, and create business for aftermarket automotive industry.

Current practice of engine tune-up relies on the experience of the automotive engineer, who will handle a huge number of combinations of engine control parameters and carry out many engine testes on the dynamometer according to different combinations of engine control parameters. The relationship between the input and output parameters of a modern car engine is a complex multivariable nonlinear function, which is very difficult to be found [1]. Consequently, engine tune-up is usually done by trial-and-error method. This spends a large amount of time and

* Corresponding Author

money. In industrial practice, many automotive engineers like to tune up an engine by referring to an existing base map, which is obtained from the past setup of a similar engine or the same engine. The parameters of the base map are then adjusted to fit different performance requirements of the same engine, or even to fit another but similar engine. This practice exactly fulfills the working environment of Case-based reasoning (CBR), i.e., based on a retrieved similar case, adaptation is then performed to fit different new situations. So, a case-based reasoning and adaptation framework for computer-aided ECU calibration is proposed in this study.

2. Case-Based Reasoning

Case-based reasoning [2, 8, 11, 18] is a simple problem-solving paradigm that involves matching a current problem against similar problems that were successfully solved in the past. The process can be augmented by adapting solutions so that they can match the current problem more closely. There are many examples for CBR applications, e.g., an auto mechanic who fixes an engine by recalling another car that exhibited similar symptoms, a lawyer who advocates a particular outcome in a trial based on legal precedents or a judge who creates case law. Hence, CBR is a prominent kind of analogy making.

Since CBR is a kind of lazy-learning methodology, it does not induce any useful rule but stores all instances (cases) in a knowledge base called *case library*. When a new problem is entered to and then resolved by the CBR system, a case comprising this problem and its solution is also stored in the case library. So, CBR learns by accumulating cases. Maintenance of knowledge base in CBR system becomes easy and does not require re-compilation of the whole system [10, 15]. In theory, CBR has been formalized for purposes of computer reasoning as a four-step process [17, 5]:

- **Retrieve:** Given a target problem, CBR retrieves cases from case library that are relevant to solving the problem. A case consists of a problem, its solution, and, typically, annotations about how the solution was derived.
- **Reuse:** The solution is mapped from the previous case to fit the target problem. This may involve adapting the solution as needed to fit the new situation.
- **Revise:** Having mapped the previous solution to the target situation, test the new solution in the real world (or a simulation) and, if necessary, revise.
- **Retain:** After the solution has been successfully adapted to the target problem, store the resulting experience as a new case in memory.

In the following subsections, retrieve and reuse stages are described. Revise and retain stages are described in Section 4 together with the application to ECU calibration.

2.1. Similarity Function in Retrieval

In the retrieval stage of CBR, there are numerous different designs of a similarity function [11, 12]. However, a simple similarity function [17, 18] is usually employed to find the nearest neighbor for the current problem from the reference cases:

$$E (f^I, f^R, W) = \frac{\sum_{i=1}^n W_i \times \text{sim} (f_i^I, f_i^R)}{\sum_{i=1}^n W_i} \quad (1)$$

where W_i is the importance of i^{th} attribute of a case, sim is the similarity function for primitives, and f_i^I and f_i^R are the values for feature f_i in the input and retrieved cases respectively. The user could freely design the similarity function but mostly this function could employ the Euclidean distance function and turn Equation (1) to:

$$E (f^I, f^R, W) = \sqrt{\frac{\sum_{i=1}^n W_i \times \left[\frac{(f_i^I - f_i^R)}{\text{Range} (f_i)} \right]^2}{\sum_{i=1}^n W_i}} \quad (2)$$

In case symbolic features are encountered,

$$f_i^I - f_i^R = \begin{cases} 0 & \text{if } f_i^I = f_i^R \\ 1 & \text{if } f_i^I \neq f_i^R \end{cases} \quad (3)$$

where $Range(f_i)$ is the range of the i^{th} feature in the case. It is used to normalize the difference ($f_i^I - f_i^R$) and ensures the difference lies on the range (0, 1). For symbolic features, $Range(f_i) = 1$.

The retrieved case with the smallest value E is considered the most similar to the new case because the value E is now indicating the distance (difference) between the input case and the retrieved case. The similarity of the cases can also be expressed as percentage, using

$$Similarity(f^I, f^R) = [1 - E(f^I, f^R, W)] \times 100\% \quad (4)$$

2.2. Reuse

After a similar case is retrieved, it is reused to generate a new solution. The process is called *reuse* stage or *adaptation*. Adaptation methods in CBR [5, 16, 19] can be categorized according to the complexity of the problem domains, namely, substitutional adaptation, transformational adaptation and derivational replay. In ECU calibration, only substitutional adaptation, or specifically parameter adjustment, is employed because no modification on case schema is necessary. However, CBA is required as an assistant to adapt existing ECU parameters to fit a new engine or a new performance requirement.

3. Case-Based Adaptation (CBA)

Adaptation in CBR sounds easy but very difficult to implement because no general rules can cover all situations even in a very specific domain. The only way to acquire adaptation knowledge is only done by consulting human domain experts for their ways to handle different problems [13]. However, even the human domain experts may not answer precisely and accurately how the problems can be handled. In addition, there is no formal agreement on how to represent adaptation knowledge. With the emergence of CBA, the previous difficulties could be alleviated.

3.1. Mathematical Model of Adaptation

The adaptation knowledge in CBR is usually represented as rules. The adaptation rules specify, under a certain situation, how to modify the value of a feature of the case in order to generate the solution for the new problem. A prospective model of adaptation [3] is recently proposed as an *ordered sequence of adaptation rules* which is called **adaptation history**, in which adaptation rules are defined in terms of functions transforming one case into a successor case. In mathematical expression, an adapted case $C' = \alpha_m \oplus \dots \oplus \alpha_2 \oplus \alpha_1 (C)$, where C is the retrieved case and α_i is the adaptation rules or adaptation operators.

3.2. Principle of CBA

When the problem domain is simple or well-understood, the set of adaptation rules α_i can be easily and automatically selected by the system to make effect on similar old case and to produce a new one. The selection of adaptation rules is done easily by comparing the conflicting differences between the new problem and the current retrieved case. In addition, the sequence of applying the adaptation rules is also important because they may not be commutative.

Although a mathematical model of adaptation is formulated, in real world problem domains, there is still no universal method for finding out the adaptation rules α_i and their corresponding sequence. Then the selection of adaptation rules α_i and their order can only be left to the user, based on his/her domain experience. Hence adaptation in such domains is highly experience-driven. To reduce the burden from the user to review the adaptation rules and choose the most appropriate ones, another source of adaptation knowledge is necessary to guide the system for the automatic selection of suitable adaptation rules. Since this selection process is based on domain

experience, another level of CBR could be established to augment the adaptation stage. This method of integrating two levels of CBR for augmenting adaptation is called *case-based adaptation (CBA)*.

In CBA, guidance for adaptation (both what features to adapt and how to adapt the features) is given by a previous case. The guidance supplying the adaptation information is called an *adaptation case*. An *adaptation case* contains meta-information for adaptation and is different from a *case* in CBR. When potential problems with solutions are identified, it is not always clear exactly what should be fixed to solve the problem. If blindly applying adaptation rules, the newly generated solution may not be qualified enough although it may still satisfy all the user specifications. However, previous experience to guide adaptation can promote novel adaptations leading to creative solution.

Domain expert, whose knowledge of how adaptation should be performed can be captured during the use of a CBR system, presumably provides adaptation cases. After initial installation of *adaptation case library*, as time passes by, the expert fills gradually the *adaptation case library* by repeatedly using the CBR system. This provides a better and more natural way to acquire domain knowledge that is not easy to capture into production rules via communication with experts. Hence the main objective of CBA is to provide a more convenient and consistent method to perform case adaptation in some complex problem domains such as ECU calibration. The overall operation of CBA is shown in Fig. 3 and it mainly consists of:

1. Retrieval stage of adaptation cases: it takes place from Fig. 3, Steps 1 to 4. Adaptation is done at Step 5.
2. Learning stage (of *case* and *adaptation case*): it takes place from Steps 6 to 9.
3. Maintenance stage: it is done (Step 6 & Step 9) for checking the consistency and redundancy when a learned *case* or *adaptation case* is to be added to the case libraries.

3.3. Structure of Adaptation Case

CBA can be viewed as another CBR for adaptation process. The purpose of this inner CBR is to assist the outer CBR system for adaptation. The case representation of the outer level of the CBR system is usually domain-dependent. So, there is no formal model of constructing case representation for a CBR system. However, for the inner level of the CBR system (CBA), the case representation [4, 7, 9, 20] can be generalized into a format that should contain at least the following information:

- I. Adaptation capability:** Functionality of the adaptation case, e.g., which features can be modified, which features can be added, deleted, etc.
- II. Previous successfully adapted case:** Reference to the past similar situation for successful adaptation, just simply an ID linking a normal case.
- III. Adaptation history:** Suggested rules for adaptation.

The first and the second parts of an adaptation case are used as retrieval index (Fig. 3, Step 2). To retrieve a similar adaptation case, the first step is to match the conflicting features of the current problem against the *capability* (Part I) of an adaptation case. If the adaptation case contains the knowledge to solve the conflicting features of the current problem, it is said that the adaptation case is capable to handle the adaptation stage for the current problem, and the adaptation case is selected as one of the candidates to provide decision support during adaptation.

The second step in retrieving a similar adaptation case (Fig. 3, Step 3) involves matching the *current retrieved case* against the *past case* successfully adapted (Part II of the adaptation case). The matching is done exactly the same as the retrieval using Eq. (4). If the past successfully adapted case referenced in the adaptation case is similar to the current retrieved case, then the third part of the adaptation case (adaptation history) is recommended to the user (Fig. 3, Step 4). Since the retrieved adaptation history is just a reference for adaptation, successful adaptation is not guaranteed. Then the user needs to refine the sequence. The user-refined adaptation history is captured as the result part of a new adaptation case in order to handle future similar adaptation process (Fig. 3, Step 7). The *current retrieved case* fills in the second part of the adaptation case while the third part is filled with the user-refined adaptation history. Based on this adaptation history, the list of functionalities for the adaptation case can be extracted, such that the first part of the adaptation case can be filled (Fig. 3, Step 8).

3.4. Matching Algorithm for CBA

The retrieval of adaptation case is based on two issues: capabilities to handle solution features and the similarity of base case to the current retrieved case. In finding capable adaptation case, there are three possible situations:

- a. **Exact match:**
Attributes (current retrieved case) = Attributes (adaptation case)
- b. **Similar and adaptable:**
Attributes (current retrieved case) \subseteq Attributes (adaptation case)
- c. **Similar but not adaptable:**
Attributes (current retrieved case) – Attributes (adaptation case) $\neq \emptyset$

Exact match means that the solution attributes necessary to adapt are exactly the same as the ones provided by the adaptation case. This is the ideal situation. The similar and adaptable situation happens when the adaptation case contains not only the features to adapt but also the adaptation capability for some unnecessary features. Some unwanted results will then be produced and human intervention is necessary to modify the set of adaptation rules. The last possible situation in matching of adaptation case is the similar but not adaptable one. When the adaptation case is not capable to adapt a required solution feature, it is *not* adaptable for the current situation. Invention (by human) of new adaptation rules may be necessary to overcome this *not-adaptable* problem. The matching algorithm is simply illustrated in Fig. 4, where ‘Attributes(X)’ is a set, ‘–’ is operator of set difference, |X| is operator of set cardinality.

3.5. Learning & Maintenance in CBR and CBA

In CBR, whenever adapting an existing solution to solve a “new problem”, a new case containing the “new problem” along with the adapted solution is generated. Then this new case is checked against the case library for consistency and redundancy. If an exact (or highly similar) case is found in the case library, then the new case is not added into the case library because the case library has already covered this new case. For a contradicting case, it is either removed or repaired based on a maintenance process, which belongs to the topic of *case base maintenance* [14] and is out of the scope of this paper. In the system implementation of this research, a contradicting case is ignored. After checking the case consistency and redundancy, the new case is added to the case library for future reuse (Fig. 3, Step 6). This is the learning and maintenance process of CBR – RETAIN. Similarly, CBA uses the same procedure for learning and maintenance (Fig. 3, Step 9).

Comparing to general learning methodologies, knowledge is not required to extract during learning stage. Learning in CBR means storing a new case. This process is much easier than the eager learning methods. Maintenance in CBR is also easier because this is done directly during learning stage where the eager learning methods cannot. So, less effort is necessary for knowledge base maintenance in CBR.

4. Application to ECU Calibration

In modern automotive engines, a lot of engine performance is affected by the control parameters in the ECU, such as power performance, idle speed performance, emission performance, etc. As the scope of the problem domain is very wide, this project selects an engine ECU calibration for aftermarket power performance tune-up to demonstrate the effectiveness of the CBR and CBA methods. The engine power performance is usually expressed as a power curve against speeds. The engine power at specific engine speed is mainly determined by the engine torque at wide open throttle and the formula is shown in Eq. (5) [6].

$$HP_r = \frac{2\pi \times r \times 9.81 \times T_r}{746 \times 60} \quad (5)$$

where HP_r : Engine horsepower at the corresponding engine speed r (Hp)
 T_r : Engine torque at the corresponding engine speed r (kg-m)
 r : Engine speed (RPM : Revolution per minute)

An example of power performance data of an engine output horsepower and torque against speeds is shown in Fig. 5. For aftermarket tuning, the user has a lot of freedom to adjust the power of a car according to their needs. Eq. (6) shows the goal of engine power performance tune-up $P(\mathbf{x})$ for aftermarket tuning, which tries to maximize the engine torque at specific speed subject to various user-defined weights.

$$\text{Maximize } P(\mathbf{x}) = \sum_{r \in \Gamma} w_r T_r(\mathbf{x}) \quad (6)$$

where w_r is the user-defined weight of the torque function $T_r(\mathbf{x})$ at the corresponding engine speed r . The range of each weight is $[0, 10]$. From the operating characteristics of the internal combustion engine, it is impossible to obtain maximum engine torque across all engine speeds; a trade-off among engine speeds is necessary (i.e. preference to low speed torque or medium speed torque or high speed torque). Therefore, the user-specific weights have to be introduced. In ECU calibration, each calibration map is usually divided the engine speed discretely with interval 500 RPM (See Fig. 1 again). Hence $r \in \Gamma = \{1000, 1500, 2000, \dots, 8500 \text{ RPM}\}$. In Eq. (6), the torque function $T(\mathbf{x})$ is equivalent to the torque produced by an ECU setup \mathbf{x} at wide open throttle, and the following ECU parameters are selected in this case study for demonstration purpose:

$$\mathbf{x} = \langle I_r, t_r, J_r, d, vt \rangle$$

I_r : Ignition spark advance at the corresponding engine speed r (degree before top dead centre)

t_r : Fuel injection time at the corresponding engine speed r (millisecond)

J_r : Timing for stopping the fuel injection at the corresponding engine speed r (degree before top dead centre)

d : Ignition dwell time at 15V (millisecond)

vt : VTEC changeover speed (2500 – 7000 RPM, interval: 500 RPM)

According to the above information and goal, the case definition of the current case study can be constructed as $\mathbf{C} = (\mathbf{w}, \mathbf{x})$. In terms of CBR jargon, \mathbf{w} is the problem part while \mathbf{x} is the solution (retrieved) part. For illustration, an example of well-known HONDA B-series VTEC engine is used. Suppose an automotive engineer has already calibrated a HONDA B16A engine based on different engine torque requirements (i.e., weight vector \mathbf{w}), he/she produces a base map (i.e., ECU setup \mathbf{x}). All of these cases (i.e., (\mathbf{w}, \mathbf{x})) are stored in a case library \mathbf{CB} (Later, we denote a case library containing ECU setups for B16A by \mathbf{CB}_{B16} , while for another HONDA B-series engine B18C by \mathbf{CB}_{B18}). When a HONDA B18C engine needs ECU calibration, CBR can be applied to reuse cases and adaptation knowledge from \mathbf{CB}_{B16} . The reason is that B16A and B18C engines are actually very similar in hardware configuration. Their ECU setups are also similar in a way. The following subsections describe the four CBR stages applied to solve the ECU calibration problem.

4.1. Retrieval

Suppose an ECU setup \mathbf{y} for a Honda B18C engine is required under a new user-input torque requirement $\mathbf{z} \in [1, \dots, 10]^{16}$. Where 1 to 10 are the importance for the engine torque at the corresponding engine speed, and there are totally 16 engine speeds (i.e., 1000, 1500, ., 8500). The weights \mathbf{z} can be considered to be the problem part of a case. The similarity function for current application is specified in Eq. (2), where $f^I = \mathbf{z}$, $f^R = \mathbf{w}$, and $W = [1, 1, 1, \dots, 1]$ because all features are equally important in current application. The ranges of all features in \mathbf{z} , \mathbf{w} are the same (i.e., within $[1, 10]$) so the ranges of features in Eq. (2) are constant. It should be noted that $\mathbf{w} \in [1, \dots, 10]^{16}$, because they specify the weights of 16 different engine speeds. Using Eq. (4) to compute the similarities for \mathbf{z} against every \mathbf{w} in the case library \mathbf{CB}_{B18} , the most similar case $c_{best} = (\mathbf{w}_{best}, \mathbf{x}_{best})$ is retrieved (Fig. 6), if any, where \mathbf{x}_{best} means the most similar ECU setup. However, if no similar case exists in \mathbf{CB}_{B18} , then \mathbf{CB}_{B16} is selected to be the most suitable case, but adaptation is needed to adapt a B16A ECU setup to fit the B18C engine.

4.2. Reuse

Assume no similar case can be found in \mathbf{CB}_{B18} , but a similar case can be found in \mathbf{CB}_{B16} . The retrieved case $c_{best} = (\mathbf{w}_{best}, \mathbf{x}_{best})$ only works for a B16A engine, where \mathbf{w}_{best} are most similar to the input problem \mathbf{z} . For the B18C engine, the solution \mathbf{y} is derived from \mathbf{x}_{best} by applying expert adaptation knowledge according to the differences between \mathbf{w}_{best} and \mathbf{z} . For example, if $\mathbf{w}_{best, 2000} = 4$ and $\mathbf{z}_{2000} = 3$, they differ by $tq_diff_{2000} = \mathbf{z}_{2000} - \mathbf{w}_{best, 2000} = -1$. Based on these differences with

respect to all engine speeds, then the solution $\mathbf{y} = \text{CBA}(\mathbf{x}_{best}, tq_diff_r)$ with respect to r can be obtained. In the following, the adaptation knowledge for handling tq_diff_r is described.

4.2.1. Adaptation Knowledge / Rules

Adaptation knowledge is in fact a list of individual rules called *adaptation rules*, which contains a sequence of adaptation operators as illustrated in Table 1. This rule governs the modification of the solution part \mathbf{x}_{best} of the retrieved case. In this case study, only substitutional adaptation, or specifically parameter adjustment, is employed because no modification on case schema is necessary. Table 2 specifies the structure of such an adaptation operator (e.g., op21, which means that a value of 10 is added to the feature t_{2000}).

For retrieval efficiency and management, the adaptation knowledge for one application (e.g., B16A engine to B16B engine, or B16A engine to B18C engine, etc.) is usually stored in a single table independent of other cases and other applications. Moreover the expert users can modify the table as necessary.

4.2.2. Adaptation Cases

Although adaptation rule is defined, the burden of choosing and performing appropriate adaptation still leaves to human users. CBA can aid this problem as discussed in previous section. According to the definition in Section 3.3, the three parts of an adaptation case can be defined for current case study (Table 3). The first part ‘‘Adaptation capability’’ contains engine speed and torque difference together. The second part contains just a case ID. In the first adaptation case, for example, case #120 has been successfully adapted to fit a new problem, using the operators {op21, op33, op45, ...} to fix the torque difference of +1 under 1000RPM. Although the sequence of operators can be simplified to adaptation rules, it is still recommended to use adaptation operators because sometimes an adaptation rule may contain unwanted side-effects on the features.

4.2.3. Procedure of CBA for ECU calibration

When the torque requirements \mathbf{z} is specified from user, the most similar case $c_{best} = (\mathbf{w}_{best}, \mathbf{x}_{best})$ is retrieved. By simply comparing \mathbf{z} and \mathbf{w}_{best} , a list of torque weight difference can be easily calculated, e.g., $\mathbf{z} - \mathbf{w}_{best} = [tq_diff_{2000}, tq_diff_{2500}, \dots, tq_diff_{8500}] = [1, 3, \dots, 0]$. Each tq_diff_r , except equals to 0, requires performing a case-based adaptation. The matching procedure is simple and straightforward as described in Section 3.4. The corresponding adaptation rules for tq_diff_r are then returned.

In this case study, the adaptation rule for a tq_diff_r is just to manipulate the ECU features with respect to the engine speed r . For example, for tq_diff_{2000} , adaptation rule = {op21, ...} only modifies the features $I_{2000}, t_{2000}, J_{2000}$, etc., but not doing anything to the features $I_{3000}, t_{4000}, J_{5000}$, etc. Hence the adaptation rules are commutative. Finally, the target solution \mathbf{y} may be obtained by applying these adaptation rules subject to tq_diff_r over all engine speeds, e.g., $\mathbf{y} = (\{op78, \dots, op120\}) \oplus \dots \oplus (\{op21, \dots, op19\}) \oplus (\mathbf{x}_{best})$, where \oplus means applying adaptation operators. However, the operators themselves inside the adaptation rule are ordered, i.e., $\{op21, \dots, op19\} \neq \{op19, \dots, op21\}$. Moreover, the retrieved adaptation rules are just for the user’s reference. They may not exactly match the current torque requirements \mathbf{z} . There are even some conflicts within adaptation rules. That is the reason why human revision on the adaptation rules is required, as discussed next.

4.3. Revise

Although the adapted solution \mathbf{y} for torque requirements \mathbf{z} is obtained as mentioned in previous subsection, the torque requirements \mathbf{z} may not be actually satisfactory. Even worse, the automatically adapted solution is far from satisfactory. At this moment, user intervention is necessary to modify the *retrieved* adaptation rules according to his/her expertise. When performing adaptation, there are at most 16 retrieved adaptation rules for tq_diff_r because of $r \in \Gamma = \{1000, 1500, 2000, \dots, 8500\}$. Each of these 16 adaptation rules needs user revision which can be done in several ways:

1. changing the sequence order of the adaptation operators;

2. inserting or removing an operator from the adaptation knowledge;
3. both 1) and 2).

If the existing operators are incapable to perform the necessary adaptation, the users can also create new adaptation operators.

4.4. Retain

A new solution $\mathbf{y} = \text{CBA}(\mathbf{x}_{best}, tq_diff_r)$ is revised and produced automatically or manually in previous stages. This new solution along with the torque requirements \mathbf{z} is considered as a new case $\mathbf{C}_{new} = (\mathbf{z}, \mathbf{y})$ for ECU setup for a B18C engine. If this setup \mathbf{y} is verified to perform well by dynamometer test (Fig. 7), the case should be retained for future reuse. In addition to the case \mathbf{C}_{new} , the adaptation during the CBA procedure (if any) should also be retained as well. In this case study, every time CBA may produce at most 16 adaptation cases according to tq_diff_r . If $tq_diff_r \neq 0$, the corresponding adaptation case needs checking for retain. The adaptation case contains the following information:

$$\mathbf{C}_{adapt} = \langle r, tq_diff, ID(\mathbf{x}_{best}), \{adaptation\ rule\} \rangle$$

where r is engine speed, tq_diff is weight difference, $ID(\mathbf{x}_{best})$ is the case # in the case library \mathbf{CB} , and $adaptation\ rule$ is the sequence of adaptation operators used. For example, $\mathbf{C}_{adapt} = \langle 2000, +1, 120, \{op23, op56, \dots\} \rangle$. Before retaining the case, \mathbf{C}_{new} should be checked that it does not appear elsewhere (at least not similar to any existing case) in the case library \mathbf{CB}_{B18} to ensure no redundancy and inconsistency. Similarly, \mathbf{C}_{adapt} is also required to check in its own adaptation case library for retain as well. The similarity checking can be done with the similarity function in Eq. (4). No matter whether there is any similar case in \mathbf{CB}_{B18} or not, the result will show to the user to judge if the case should be retained or not. Hence, high-quality cases and adaptation cases for B16A-to-B18C setup could be accumulated through this way.

5. Experiments and Results

A complete ECU calibration for a new engine model usually takes one to two years. As a matter of fact, the experienced engineer in HONDA B16A engine still spends at least half year to tune up a similar model B18C engine, because a lot of trial-and-error on ECU setups and dynamometer tests are still required. In addition, ECU calibration is similar to planning and design domains where explicit knowledge is hard to explain and managed. Therefore the learning curve of ECU calibration for a novice automotive engineer may be two to three years. Under these two issues, some experiments have been carried out to show that CBA can contribute to:

- 1). greatly reduce the necessary time with adaptation from an existing engine model to a new model;
- 2). enhance knowledge management of ECU calibration, so that novice engineers can learn better and in shorter time from the CBR system.

To accomplish the experiments, a prototype system has also been implemented using Borland C++ compiler under MS Windows XP, which provides the basic graphical user interface and database management system to construct and link a database, which is regarded as case library. The case libraries are stored in *dbf* format. Then many commercial database management systems can browse and even modify the case libraries without launching the prototype system. The whole concept and workflow of the experiments are illustrated in Fig. 8. The following subsections explain the steps in the figure.

5.1. Preparation of Case Library

To prepare a case library \mathbf{CB}_{B16} , a B16A engine is run on a dynamometer for ten times. Every time, its ECU setup \mathbf{x}_j is adjusted according to a different \mathbf{w}_j . Then, an expert automotive engineer is asked to use the CBR prototype system (Figs. 9 -15) to perform adaptation with another set of requirements \mathbf{w}_i . His/her adaptation reuses the cases from \mathbf{CB}_{B16} . During his/her use, a set of new cases \mathbf{x}_i for B16A are produced and stored in \mathbf{CB}_{B16} (Fig. 8, Step 1). The ways he/she adapted from \mathbf{x}_j to \mathbf{x}_i under \mathbf{w}_i on the same engine are also recorded in another independent database, Adaptation Case Library. The recorded information includes adaptation cases (Fig. 8, Step 2) and new generated adaptation operators (Fig. 8, Step 3). As illustrated in Fig. 8, the adaptation cases are saved in a table called "B16_B16", which means adaptation from B16A to B16A itself because

they are also valid under this kind of adaptation. However, the adaptation operators are universal over the application, so they can be put in a single table.

5.2. Experiment Workflow of Case-Based Adaptation

5.2.1. Retrieval & Reuse

It should be noted that currently \mathbf{CB}_{B18} is still empty, which means the only data source is \mathbf{CB}_{B16} . Then the objective of CBA is to adapt a best matching B16A ECU setup \mathbf{x}_{best} to fit B18C under user-specific torque requirements \mathbf{z} . To retrieve \mathbf{x}_{best} , \mathbf{z} is employed to match against \mathbf{w}_i in \mathbf{CB}_{B16} (Fig. 8, Step 4). Its corresponding weight \mathbf{w}_{best} , and case ID, $ID(\mathbf{x}_{best})$, are used for searching the most appropriate adaptation case (Fig. 8, Step 5) from B16_B16. Then a set of adaptation rules *Rules* is retrieved.

5.2.2. Revise & Retain

Currently the best matching ECU setup \mathbf{x}_{best} , and appropriate adaptation rules *Rules* are retrieved. They are passed to the engineer for checking. If no more revision is required, the adaptation can be automatically applied. For any human revision on the adaptation rules, the procedure is illustrated in Fig. 8. In the course of revision, the engineer uses the CBR system to revise the sequence of the adaptation operators, or even produce new adaptation operators as necessary. After the appropriate revision and confirmation by the engineer, a new generated case \mathbf{y} for B18C is produced and saved in \mathbf{CB}_{B18} (Fig. 8, Step 6). The two pieces of adaptation information are respectively saved to the Adaptation Case Library (Fig. 8, Step 3 & Step 7). One point to note is that the adaptation case is now from B16A to B18C. So, it has to be saved separately in another table called B16_B18. In the future, if there are adaptation cases from B18C to B18C itself, a table called B18_B18 must be created for their storage.

5.3. Experiments

Firstly, a B18C engine with original (factory) ECU is run on the dynamometer to obtain a standard torque curve for performance comparison. In order to tune up the B18C engine, a MoTeC M4 programmable ECU (Fig. 2) is used in the experiments. Then, three different torque requirements over the engine speeds, i.e., $\mathbf{z}_i = \langle z_{1000,i}, z_{1500,i}, \dots, z_{8500,i} \rangle$, $i=1$ to 3, are specified. By repeating the experiments (Fig. 8, Steps 4-7) under different \mathbf{z}_i , appropriate cases \mathbf{x}_{best} and adaptation rules *Rules* from \mathbf{CB}_{B16} are retrieved and suggested to the engineer for revision, i.e., the adapted solutions for B18C, $\mathbf{y}_i = CBA(\mathbf{x}_{best}, \mathbf{z}_i)$, are produced, $i = 1$ to 3. These three adapted solutions \mathbf{y}_i along with \mathbf{z}_i are retained in \mathbf{CB}_{B18} for future reuse. Each ECU setup in the solution \mathbf{y}_i is entered to the programmable ECU and then the engine is run on the dynamometer one by one for evaluation. Consequently, three different torque curves are produced for different torque requirements \mathbf{z}_i .

5.4. Evaluation of Results

The results can be evaluated in two issues: 1) engine performance and 2) tune-up time. Comparing to the B18C standard curve, the three different torque curves for \mathbf{y}_i are not much worse as shown in Fig. 16. On some engine speed ranges, \mathbf{y}_i even produce better performance than the factory ECU setup. It shows the effectiveness of CBA for ECU calibration. Secondly, every time \mathbf{x}_{best} and *Rules* are retrieved, the engineer was just given one hour for revision and confirmation of the adapted solution \mathbf{y} in each experiment. Therefore the qualities of \mathbf{y}_i are not optimal but still good enough. The engineer also claims that by having more time (e.g., several days), the adaptation rules may be fully revised and hence \mathbf{y}_i may give even better performance over the factory ECU setup, which takes a couple of months to tune up the power performance optimally. So, the time can be greatly reduced from months to days.

6. Conclusions

This paper describes the techniques of CBR and CBA applied in automotive ECU calibration. Although CBR was proposed and continually formulated since 25 years ago, a good and feasible adaptation technique for general case had not been established because adaptation knowledge is



highly domain-specific and experience-driven. Hence, adaptation is traditionally a difficult task. In the research, case-based adaptation has been successfully applied to alleviate the difficulty. From the perspective of automotive ECU calibration, the application of CBR and CBA is a new attempt. To verify the proposed methodologies, a prototype CBR & CBA system has been implemented. A case study shows that the CBR and CBA system can shortly recommend a sub-optimal setup, which is close to an optimal setup done by a car manufacturer. Additionally, the system is embedded with learning ability to improve itself. Learning takes place when the system is being used because the user is required to create new operators and modify the sequence of adaptation, which can later be reused. The more the use of the CBA system, the rich the knowledge can capture. Furthermore, the CBR and CBA system can be viewed as an effective knowledge management tool. From Fig. 8, after an expert automotive engineer uses the CBR system for ECU calibration, his/her knowledge is logged. Next time, a novice automotive engineer may reuse his/her knowledge for ECU calibration. This is extremely valuable for a company because much domain knowledge cannot be explicitly specified and explained. With the CBR system, the domain knowledge can be saved for other engineers as a reference. Especially for novice engineers, the CBR system can act as a mentor while they are learning and hence also shorten their learning curve. Moreover, the proposed CBR and CBA methodologies are generic, they can be applied to the other ECU calibration problems and engineering applications.

References

- [1] Li GY. Application of Intelligent Control and MATLAB to Electronically Controlled Engines. Publishing House of Electronics Industry (In Chinese).
- [2] Armengol E. Usages of Generalization in Case-Based Reasoning. In Proceedings of International Conference on Case-Based Reasoning 2007 (ICCB2007); 2007, p. 31-45.
- [3] Bergmann R, Wilke W. Towards a New Formal Model of Transformational Adaptation in Case-Based Reasoning. In Proceedings of European Conference on Artificial Intelligence 1998 (ECAI '98); 1998, p. 43-52.
- [4] Burke EK, Qu R, Petrovic S, MacCarthy B. Structured cases in case-based reasoning - reusing and adapting cases for timetabling problems. Knowledge-Based Systems; 13:(2-3), p. 159-165.
- [5] Gebhardt F, Voß A, Gräther W, Schmidt-Belz B. Reasoning With Complex Cases. Kluwer Academic Publishers; 1997.
- [6] Pulkrabek WW. Engineering Fundamentals of the Internal Combustion Engine, 2nd Edition. Pearson Prentice Hall; 2004.
- [7] Kelbassa H. Optimal Case-Based Refinement of Adaptation Rule Bases for Engineering Design. In Case-Based Reasoning Research and Development: 5th International Conference on Case-Based Reasoning, ICCBR 2003, LNAI 2689; 2003, p. 201-215.
- [8] Kolodner J. Case-Based Reasoning. Morgan Kaufman Publication, San Mateo, CA, U.S.A.; 1993.
- [9] Lieber J. Application of the Revision Theory to Adaptation in CBR: the Conservative Adaptation. In Case-Based Reasoning Research and Development: 7th International Conference on Case-Based Reasoning, ICCBR 2007, LNAI 4626; 2007, p. 239-253
- [10] Nilsson M, Sollenborn M. Advancements and Trends in Medical Case-Based Reasoning: An Overview of System and System Development. In Proceedings of the 17th International FLAIRS Conference; 2004, p. 178-183.
- [11] Pal S, Shiu S. Foundations of Soft Case-based Reasoning. Wiley, John & Sons, Incorporated; 2004.
- [12] Pan R, Yang Q, Li L. Case retrieval using nonlinear feature-space transformation. Advances in Case based Reasoning, LNAI 3155; 2004, p. 361-374.
- [13] Seifert C. Case-Based Reasoning by Human Experts. In Case-Based Reasoning Research and Development: 6th International Conference on Case-Based Reasoning, ICCBR 2005, LNAI 3620; 2004, p. 4-5.
- [14] Shiu SCK, Yeung DS, Sun CH, Wang XZ. Transferring case knowledge to adaptation knowledge: an approach for case-base maintenance. Computational Intelligence: an International Journal 2005; 17 (2): 295 – 314.
- [15] Thibault A, Siadat A, Martin P. A Framework For Using A Case Based Reasoning System Applied To Cost Estimation. In Proceedings of IEEE Conference on Cybernetics and Intelligent Systems; 2006, p. 1-6.
- [16] Veloso M. Planning and learning by analogical reasoning. Springer-Verlag, Reading; 1994.

- [17] Watson I, Marir F. Case-based Reasoning: A Review. Knowledge Engineering Review 1994; 9 (4): 382-419.
- [18] Watson I. A case study of maintenance of a commercially fielded case-based reasoning system. Computational Intelligence: an International Journal 2001; 17(2): 387-398.
- [19] Weber B, Wild W. CBRFlow: Enabling Adaptive Workflow Management through Conversational Case-Based Reasoning. In Advances in Case based Reasoning, LNAI 3155; 2004, p. 434-448.
- [20] Zhang Y. Case-Based Reasoning Adaptation for High Dimensional Solution Space. In Case-Based Reasoning Research and Development: 7th International Conference on Case-Based Reasoning, ICCBR 2007, LNAI 4626; 2007, p. 149-163.

Figures and Tables

96081300 / General / Advanced Tuning - Default Map				ECU Connect V4.20						
	RPM	5500		TP	100.0 %	Eng Temp	93 °C			
	Effcy	100.0		MAP	102.0 kPa	Air Temp	40 °C			
	Load	102.0		Aux U	100.0	Diag Errors	0			
	Lambda	0.88		Bat U	14.4 U					
F U E L	Pulse W	8.5 mSec		Duty Cycle	78 %	INJ Time	215 deg			
IGNITION	Advance	43.0 BTDC		Dwell	2.9 mSec	LA Ctrl	OFF			
Fuel Main			< % of IJPU >	Trim	0.0 %	Lambda Was	0.90			
Eff \ RPM		3000	3500	4000	4500	5000	5500	6000	6500	7000
100		39.0	43.0	46.0	49.5	52.5	* 55.5	56.5	56.5	55.0
90	*	37.0	39.5	42.5	45.5	48.5	* 50.5	51.5	51.5	50.5
80		34.0	36.5	39.0	41.5	44.0	* 46.5	47.0	46.5	45.5
70		31.5	33.5	35.5	37.5	39.5	41.0	* 49.5	41.5	41.0
60		29.0	30.5	32.0	* 35.0	35.5	* 41.0	* 38.5	37.0	36.0
50		26.5	27.0	28.0	* 36.0	31.0	32.0	32.0	32.0	31.0
40		24.0	24.0	24.5	25.5	26.5	* 35.5	27.0	27.0	26.5

F1-Help F3-Diag F5-Ign F6-EOI F9-Func PgUp/Dn-Adj Enter-Set Esc-Screen/End

Fig. 1. A screenshot of fuel map calibration in an ECU



Fig. 2. Engine tune-up using programmable ECU

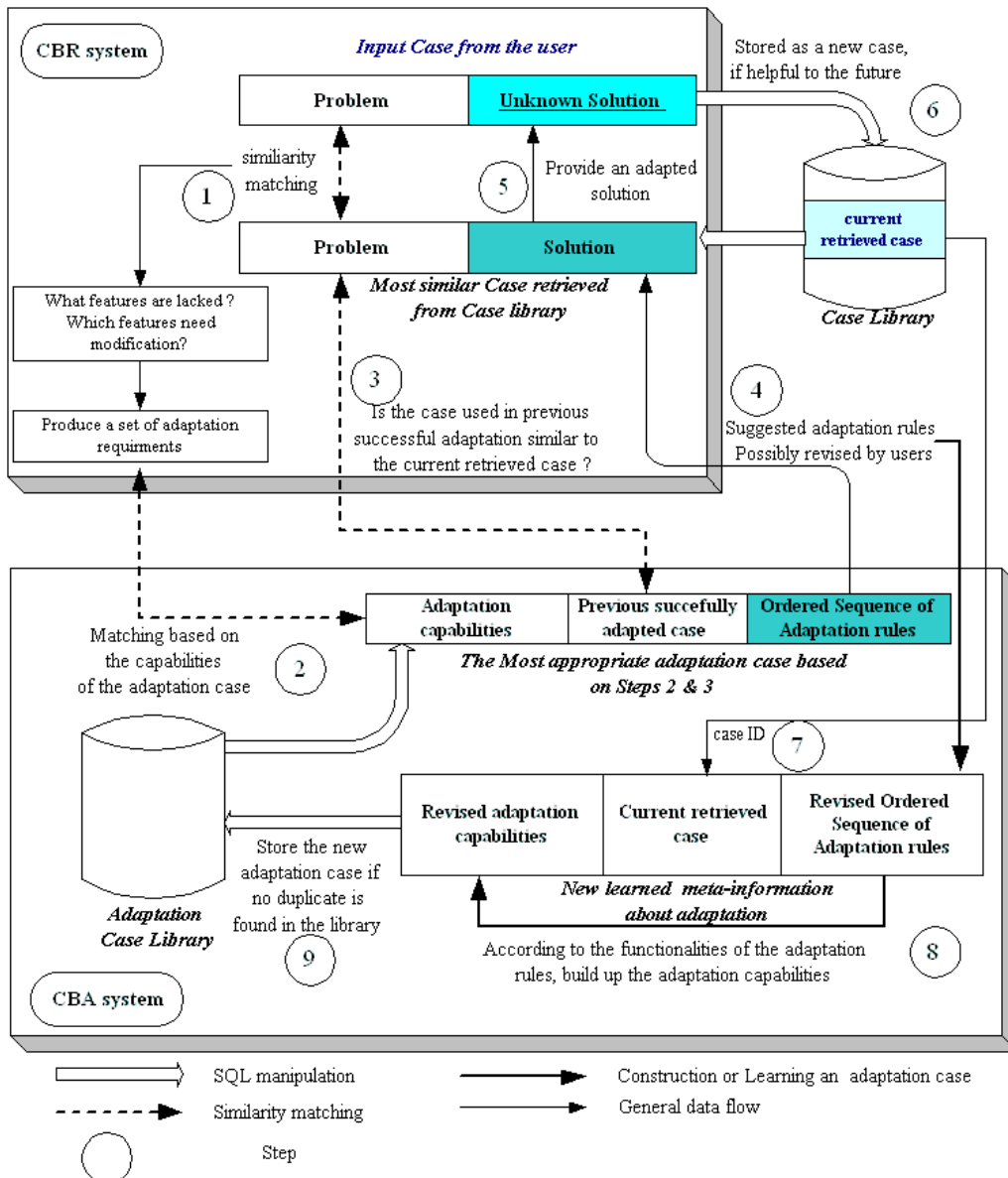


Fig. 3. Workflow of CBR & CBA

```

if  $Attributes(current\ retrieved) - Attributes(adaptation\ case) = \emptyset$ 
then find the adaptation cases such that
     $Min(|Attributes(adaptation\ case) - Attributes(current\ retrieved)|)$ 
    {If the result is 0, it means the adaptation case is an exact matching.}
else find the adaptation cases such that
     $Min(|Attributes(current\ retrieved) - Attributes(adaptation\ case)|)$ 
    {Find out adaptation cases covering as many resolving features as possible.}

```

Fig. 4. Matching Algorithm for CBA

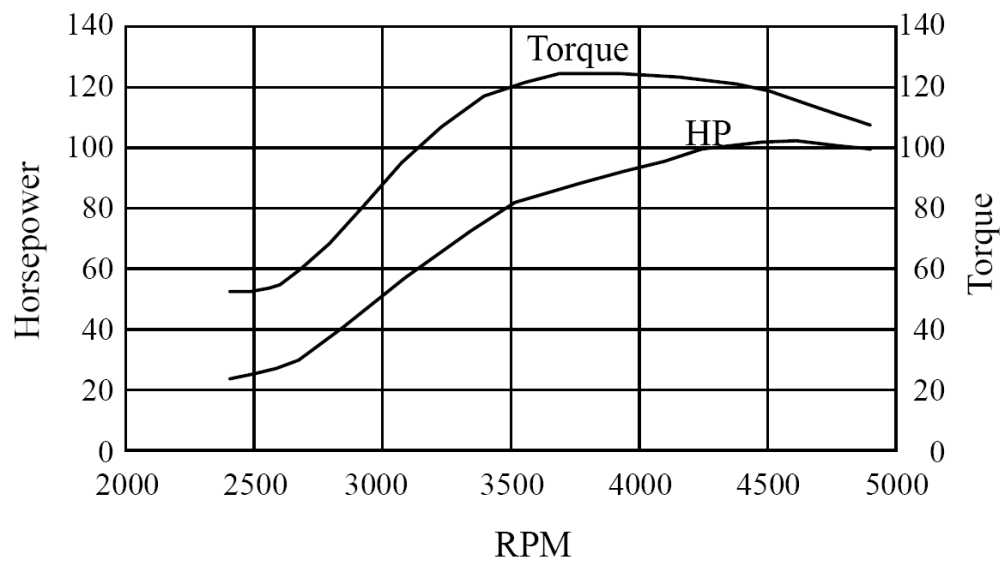


Fig. 5. Example of automotive engine power and torque curves

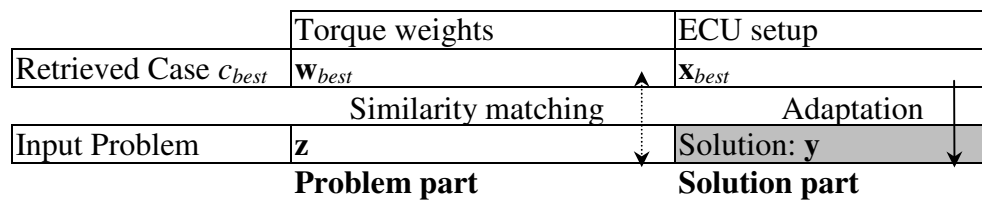


Fig. 6. Retrieval and adaptation for ECU setup



Fig. 7. Car engine performance data acquisition on a dynamometer

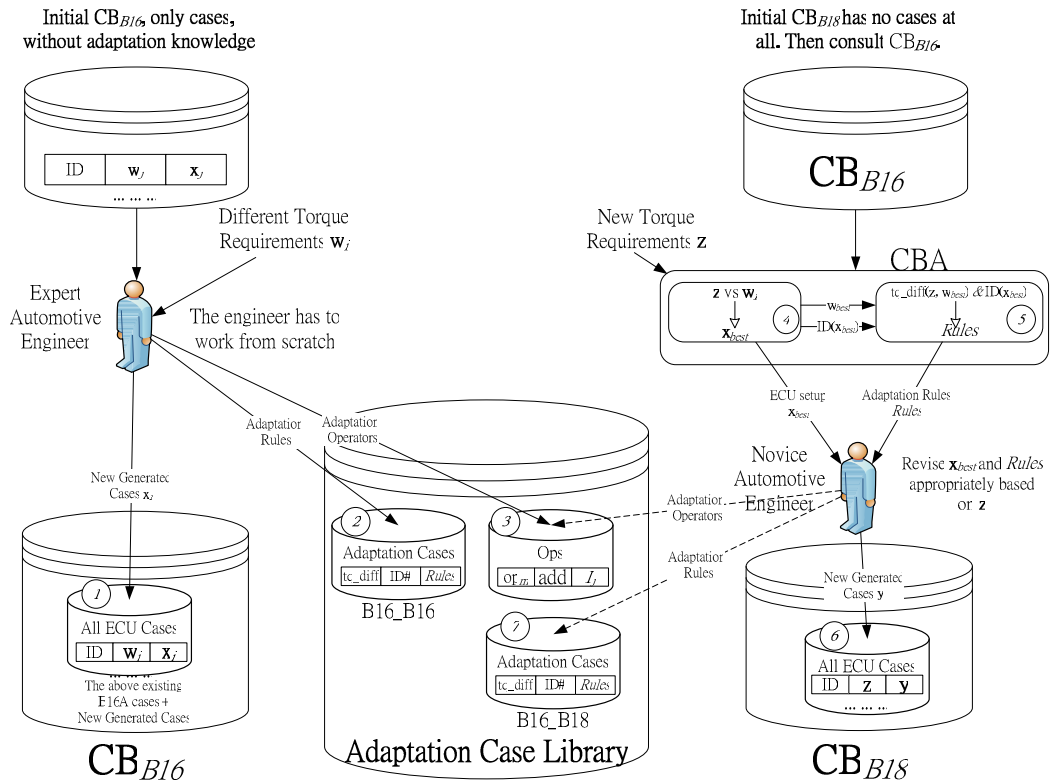


Fig. 8. Workflow of CBR & CBA for ECU calibration with a case study illustrating an adaptation from B16A ECU setup to B18C ECU setup

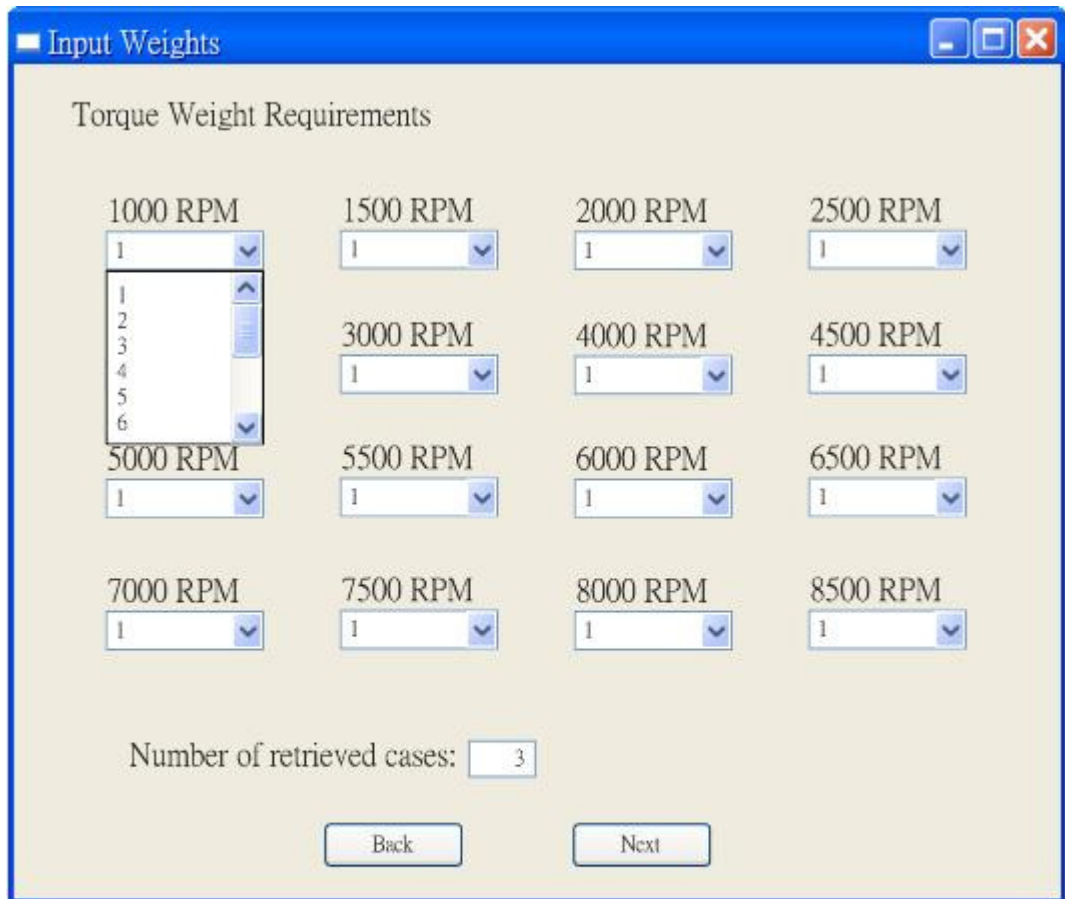


Fig. 9. Dialogue for entering engine torque weights in the prototype system

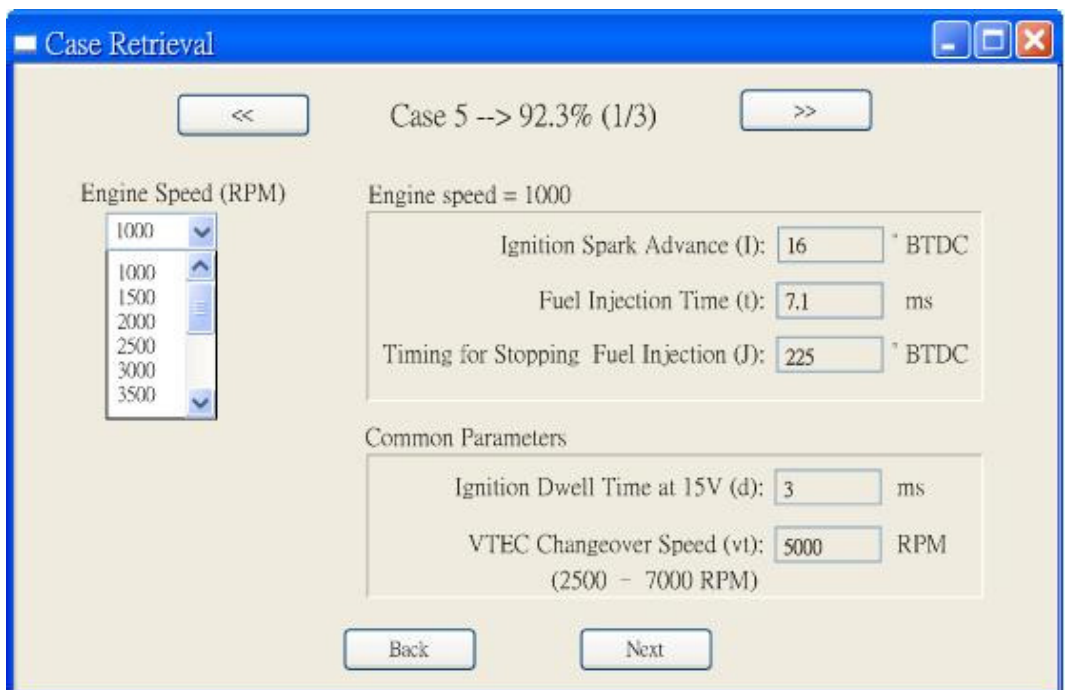


Fig. 10. Retrieval of similar case

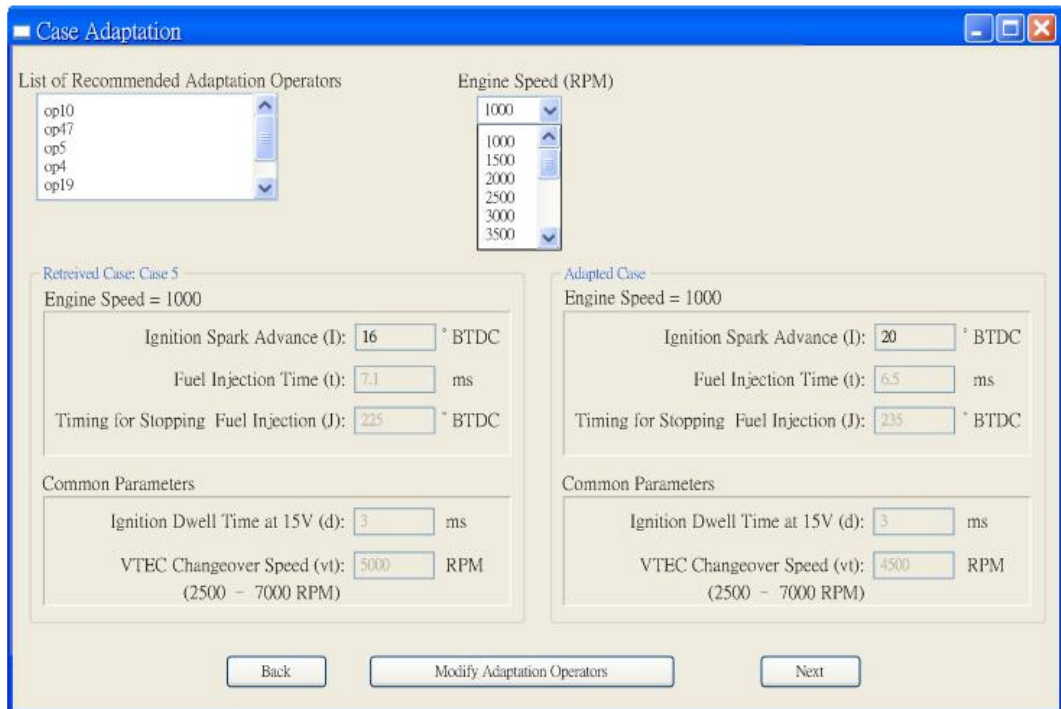


Fig. 11. Adaptation rules recommended from the prototype system

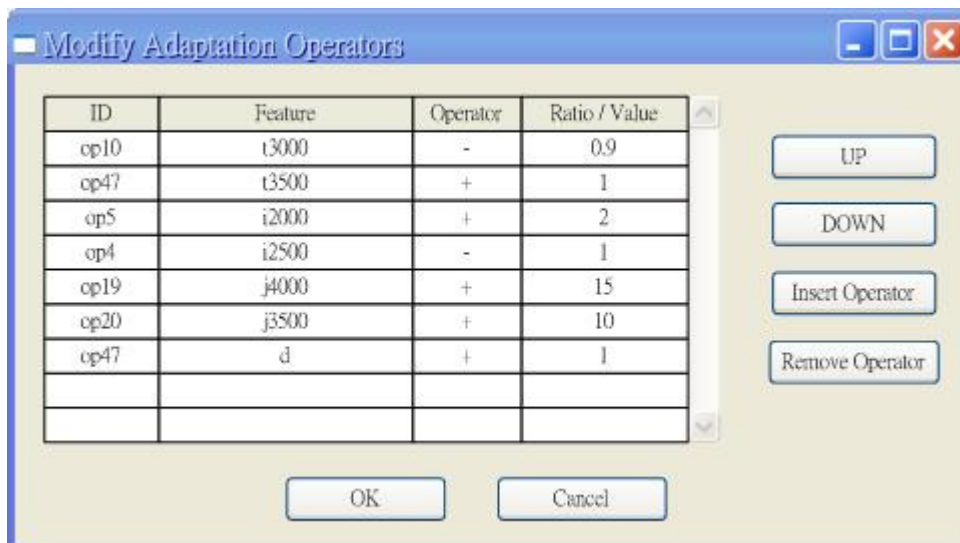


Fig. 12. Modifying adaptation operators

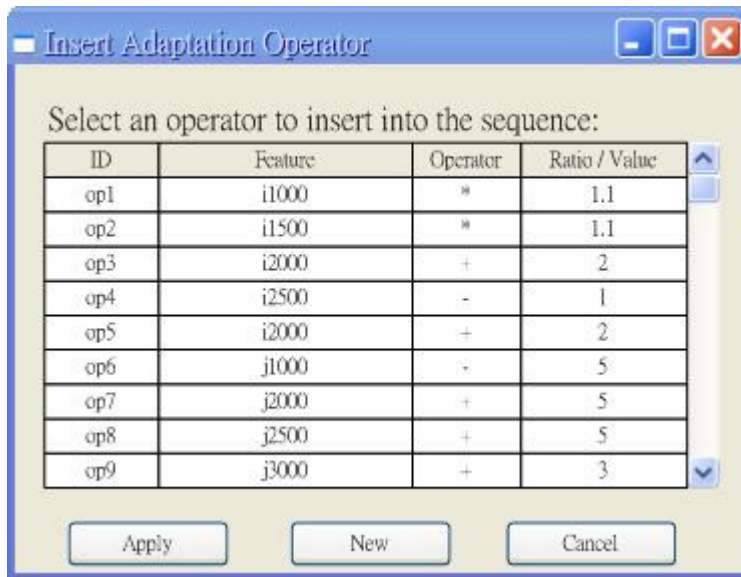


Fig. 13. Inserting adaptation operators



Fig. 14. Creating new operator

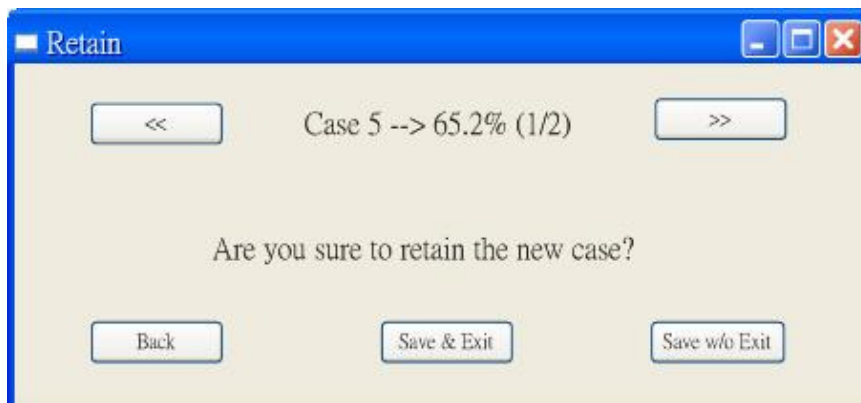


Fig. 15. Checking before retaining an adapted solution

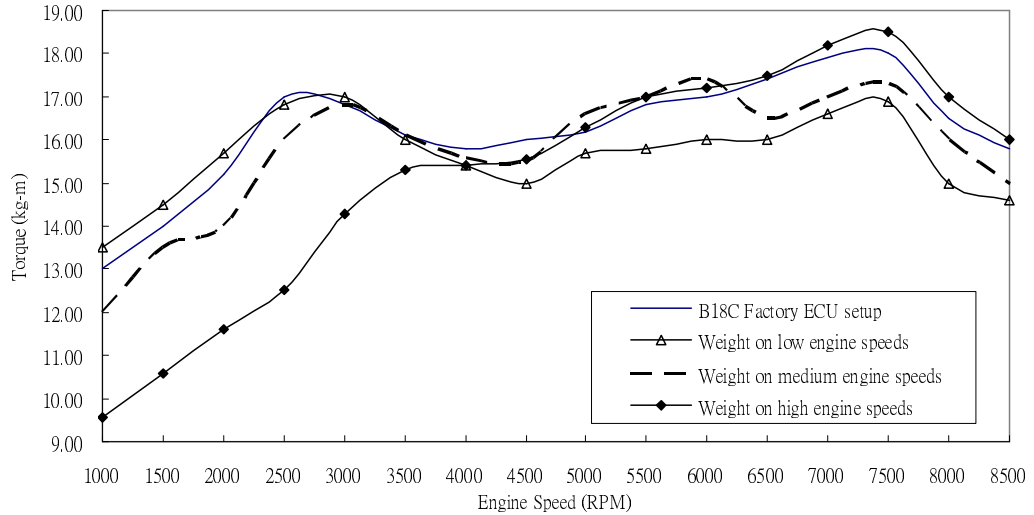


Fig. 16. Original B18C torque curve and its comparison with the torques curves of the three adapted ECU setups

Table 1. Structure of adaptation knowledge (Rules)

<i>Adaptation Rule No.</i>	<i>Sequence of operators</i>
1	{op21, op33, op45, ...}
2	{op21, op55, op53, ...}
..	...
<i>N</i>	{op15, op35, op23, ...}

An adaptation rule

Table 2. Format of an adaptation operator

<i>Operator #</i>	<i>Feature name</i>	<i>Operation</i>	<i>Ratio/Value</i>
21	t_{2000}	<i>add</i>	10

“Operator #” is the index of the operator, e.g., *op23*, *op56*, etc.
 “Feature name” means the name of the feature in the case for modification.
 “Operation” performs either: *substitute*, *add*, *minus*, *multiply*, or *divide*.
 “Ratio/Value” is the operand used for the operation, which is a real number.

Table 3. Structure of adaptation cases

Adaptation cases for B16A setup to B18C setup for torque weight difference			
<i>Adaptation capability</i>		<i>Previously successfully adapted case</i>	<i>Adaptation rule</i>
<i>Engine speed</i>	<i>tq_diff</i>	<i>Past Case ID</i>	<i>Sequence of operators</i>
1000	1	120	{op21, op33, op45, ...}
1000	2	10	{op21, op55, op53, ...}
...
8500	-9	50	{op17, op45, op24, ...}
8500	-8	70	{op17, op55, op13, ...}
...
8500	9	250	{op15, op35, op23, ...}