Optimisation of the Gas-Exchange System of Combustion Engines by Genetic Algorithm

C. D. Rose, S. R. Marsland, and D. Law
School of Engineering and Advanced Technology
Massey University
Palmerston North, New Zealand
C.D.Rose@massey.ac.nz, S.R.Marsland@massey.ac.nz

***Abstract*—Current techniques for the optimisation of combustion engine gas-exchange systems still predominantly use trial and error. This paper proposes a new method for the optimisation of these systems through the use of modified Genetic Algorithm techniques, principally a variable length chromosome encoding. Promising initial results are presented and discussed.**

Keywords - genetic algorithm; variable-length input encoding; combustion engine; optimisation

I. Introduction

The gas-exchange system is a primary factor in the performance of a combustion engine. Designing an efficient gas-exchange system is therefore a very important task, but there are many parameters that must be considered, with complex non-linear interactions between the parameters. This makes the development of optimal systems very difficult. While, in the past, the only way that designs could be tested was by building test-rigs and physically running them, modern simulation software can reliably predict the behaviour of these systems, and this has significantly simplified the process. However, the design process still involves a large amount of trial and error from human designers. In order to reduce the complexity of the problem, the very large and complex tasks are often separated into smaller sections that can then be optimised individually, but in doing so information about the interactions between these smaller sections is lost and the best solution may never be found.

Given a set of output parameters that should be maximised (such as power output at some set number of revolutions per minute, or minimum emissions over a range of engine speeds) it is possible to use computational optimisation methods to search for solutions to this problem. The complexity of the function being optimised, and the lack of known gradients for the functions, mean that stochastic search methods are more suited to the problem.

In this paper we will investigate the use of Genetic Algorithms (GA) as an optimisation tool for combustion engine gas-exchange systems. Genetic Algorithms are powerful heuristic search methods that have been successfully used to find optimal or near-optimal solutions in many very complex design spaces. Their use in automotive optimisation has so far been limited to either very specific tasks or very simple parameter sets. We modify the basic genetic algorithm by allowing the complexity of the input encoding to change, so that the gas-exchange system itself can become more complicated.

A. Engine Design

The gas-exchange system of an internal combustion engine controls the movement of air into the combustion chamber and the movement of exhaust gasses out after combustion. The performance of this system will affect the way the engine behaves, including areas often of interest to designers such as power output, exhaust emissions, fuel economy and noise. Compared to many other engine parameters, the arrangement of the gas-exchange system is fairly easy to modify and new components can be retrofitted to existing engine designs to adjust their performance for specific uses.

The evaluation of gas-exchange components has been made relatively easy through the use of computer simulation and there are several commercial software packages available to designers. These tools can significantly reduce the time required to find good solutions, but they do not in themselves solve the problem of finding good models. The optimisation methods used are still based primarily on trial and error, guided by human designers [1].

In order to find the completely optimal solution it would be preferable to consider all variables in our model at once however this is rarely possible. A human designer is unlikely to cope well with an optimisation problem containing 10 or more variables and this would require either a very simplistic model of the system or the division of the system into smaller sections. A simple model with only 10 variables is unlikely to be able to describe the optimal solution, while dividing the system into sub-models results in the loss of information regarding interaction effects between the sub models.

Another problem which may be encountered is a lack of knowledge on how various design features will interact or affect the outcome. Problems like this are often referred to as non-routine design problems and they are usually approached in a more ‘creative’ manner [2]. Due to the extreme complexity of automotive gas-exchange systems it is virtually impossible to understand how the many parameters will interact with each other and as a result the design process becomes more of an art than a science.

An alternative approach would be to use a computer based optimisation tool to help us solve this problem, although the task is not trivial for computers either. The simplest method would be to evaluate every possible combination of our input variables and then pick the best one, however as the number of input variables increases this approach soon becomes unfeasible due to the massive number of possible combinations. Other methods of searching through these many combinations include gradient based algorithms however these rely on the presence of a continuous, differentiable function to follow, something which may or may not exist in a complex problem such as this.

B. Genetic Algorithms

One method that has been used on a great many search and optimisation problems in recent years is the Genetic Algorithm, also known as evolutionary optimisation, which uses a population of possible solutions and utilises the principles of Darwinian evolution to evolve new possible solutions that may be better fitted to the problem. Operators based on genetic crossover, 'survival of the fittest', and mutation are applied to a population of possible solutions with the goal of increasing the overall fitness [3]. While there is no guarantee that a GA will find a good solution, in practice they often return good or even near-optimal solutions to complex problems, although they require a very large number of function evaluations, and therefore a lot of computational resources, during operation.

GAs are used for optimisation in the same manner as more traditional optimisation algorithms, where the values of one or more input variables are adjusted until an optimum (or near optimum) output value is reached. The two requirements for GA optimisation are a method of encoding the problem to a gene-like representation, and some form of evaluation function which can be applied to the decoded solutions to establish how successful they will be. The gene-like representation is often a form of binary string, although, as will be seen later, other encoding methods can be used.

C. Engine Design using GAs

The use of genetic algorithms for the design of engine components has been investigated by several researchers. Some of these studies have focused on the comparison of several different optimisation strategies, while others have just used the GA as a tool to solve their particular problem. In most cases the GA is shown to be the most effective option for solving complex problems with several input variables.

Zucker [1] investigated the optimisation of engine gas-exchange systems and compared the performance of Simplex optimisation and evolutionary algorithms with one, two and several population members. The multi-member evolutionary algorithm was shown to be the best option for his more complex test problem, in which 10 basic intake, exhaust and cam parameters were optimised for maximum cylinder filling. His evaluations were carried out by the 1-D flow simulator PROMO.

Ahmadi [4] also investigated engine gas-exchange systems, but using the Multi Objective Genetic Algorithm (MOGA). Like Zucker, he uses a set of 10 simple intake, exhaust and cam parameters that are optimised for maximum volumetric efficiency. The solution evaluation used in this study is another 1-D flow simulator GT Power.

In both of these studies they attempted to optimise several engine components simultaneously, and good solutions were found using GA-based optimisation methods. However, the parameters that were optimised were very simple, comprising only a few pipe lengths and diameters. Ahmadi noted that when the criteria for high fitness involved high efficiency at both 3000 and 6000 rpm, the solution produced by the GA was good at these points only and performed quite poorly at other engine speeds.

Reitz has been involved in several studies (including [5-7]) investigating the optimisation of various diesel engine parameters to improve fuel efficiency and reduce emissions output. In the earlier experiments they used a fairly simple GA, and later progressed to more complex optimisation strategies. The GAs found good solutions for all of their test problems.

Branney et al. [8] compared the performance of a MOGA to the Simplex method and Simulated Annealing for the design of an airbox. A complex CFD tool was used to evaluate the air flow, and the GA was shown to be the most successful optimisation tool.

Only in one study [9] was a problem with many input variables addressed. Here, the profile of a camshaft was optimised to maximise performance and minimise mechanical stress on associated components. The shape of the camshaft was described by 40 variables and a good solution was found after 20 000 function evaluations.

Throughout these studies, GAs have been shown to be a highly effective tool for engine optimisation. However, they have either been applied to very specific areas [5-9] or only been allowed to change very basic parameters [1,4]. In this paper we will investigate the application of GAs to the simultaneous optimisation of multiple engine components, while providing the flexibility to explore a wide range of possible solutions. This is of clear importance given the highly non-linear interaction between the components, but it does mean that a standard GA may not be suitable for the task.

II. Methodology

As mentioned previously, one problem with gas-exchange system optimisation is the need to specify the complexity of the model before it can be optimised. If the model is too complex then it is very difficult to optimise, however if it is too simple it may not be possible to find the optimal solution. In this paper we address this problem by using a GA with chromosomes which can vary in length. This will allow the use of very complex models which can then be reduced to a sensible level of complexity by the omission of various genes from the chromosome.

We will compare a normal simple GA (labelled as SGA in the results) with a GA variant that utilises, in addition to other changes, a variable length chromosome similar to the 'messy' GA of Goldberg et al. [10]. Messy GAs (MGAs) get their name from the fact that they remove the requirement for the genes to be in a particular order in the chromosome. The Messy GA also allows genes to be over specified (repeated) or underspecified (missing) in the chromosome. The concept of messy chromosomes will allow our GA creatures to expand and contract if it is beneficial for them to do so.

The implementation of the SGA is standard. It consists of a single population of binary strings that are modified by bitwise mutation and single point crossover. Mating partners are chosen by via Tournament Selection, the next generation is created by selecting the best creatures from the parents and children. For more on GAs, see for example [3].

Our GA variant with variable length chromosomes (hereafter referred to as a VLCGA) is kept as simple as possible to allow us to focus on its features of interest, namely the chromosome structure. The algorithm is based on Goldberg's 'messy' codes and chromosomes are implemented in a linked list as they were in the original MGA. The 'cut' and 'splice' operators are retained, however the 'competitive templates' are not needed, as flexible engine models have been designed to allow fitness calculations for underspecified creatures. Mutation is carried out in the same manner as the MGA, with two separate operators used. Allelic Mutation allows the flipping of a bit in the string ie 1 to 0 or 0 to 1, while Genic Mutation swaps two genes in the string. Tournament selection is used and the implementation is identical to that utilised in the SGA.

An alternate chromosome encoding occasionally used in GAs utilises real values rather than binary strings. In some ways this simplifies the implementation as a single gene can have many possible allelic values, however mutation and crossover do not function in the same manner as in the binary encoding. The purpose of mutation in the GA is to provide a degree of 'local' searching: a small change to the chromosome allows the evaluation of a nearby location on the search space. This can be implemented in a real valued GA by adding or subtracting a small quantity from a real valued gene. Crossover is rather more difficult and can either be implemented by translating the real valued chromosome into a binary string and applying conventional crossover, or simply removing the ability for crossover to split a chromosome 'mid-gene' and forcing the operator to only split between genes.

Initially the VLCGA was coded to use a real valued chromosome, as this makes the cut and splice operations, and the interface with the evaluation function, much simpler. This however, changes the way the algorithm behaves, so a real valued variation on the SGA was also created to allow these changes to be explored. The real valued crossover operator in our implementation is only able to split the chromosome between genes. Eventually a solution was found to the problems encountered with the binary coded VLCGA. This now provided four possible GA variations which could be applied to the engine optimisation problem.

In addition to the optimisation algorithm, we also need to evaluate creature fitness. For this study, the WAVE software package from Ricardo Engineers is used. WAVE is a 1-D gas dynamics and engine performance simulator which should produce comparable results to the 1-D simulators used in previous studies [1,4].

The basic structure of a WAVE model is a network of connected tubes or 'ducts' with complex elements such as y-junctions and engine cylinders where required. Values for virtually any variable in the engine model can be defined in a 'constants table' that can also be imported from an external file. A single WAVE model is usually run for several 'cases', such as different engine speeds, with other variables like temperature and pressure values being specified as appropriate. By running several cases across a range of engine speeds the designer can gain a good picture of how the engine will perform.

The use of the binary SGA with WAVE is fairly straight forward. The chromosomes of the creatures are decoded to real values which are then entered into the WAVE constants table. The real valued chromosome does not require any decoding at all. The use of the VLCGA on the other hand is rather more difficult, as the variable length chromosomes require the evaluation of solutions which do not have every variable specified. Ideally we would like to remove elements from the model entirely if they are not specified in the genome, however in practice this would be quite difficult to achieve. The alternative solution was to leave the elements in the model, but automatically set their values such that their impact on the fitness of the creature is negligible.

As an example of this technique, consider two adjacent ducts A and B, each has several variables: a left diameter, a right diameter, and a length. Duct A is specified in our genome, it has left and right diameters of 60mm and a length of 250mm. If duct B is not specified in our genome, we can effectively remove it by setting its diameters to be the same as those of the preceding duct, and its length to a very small value, say 5mm. The overall result is a single duct with left and right diameters of 60mm, a length of 255mm, and specified by 3 variables not 6.

It is acknowledged that this is not an ideal solution as the resulting single duct is not quite the same as if we had just removed duct B entirely, however this shouldn't affect the optimisation process. If the optimal length for this section is 250mm, the GA should be able to set the length of duct A to 245mm resulting in a total length of 250mm with the desired level of complexity.

III. Experimental Design

Initially, a simple experiment was conducted using an optimisation problem similar to those addressed by other researchers. The experiment is intended as a starting point to see whether the unmodified simple GA is able to successfully find good solutions to simple design problems. The model used for the initial experiment is a 4-cylinder, 16-valve Direct Injection gasoline engine based on a demonstration model included with the WAVE software.

Input variables are selected from the intake manifold runner ducts and the exhaust manifold runner ducts. The intake plenum was left as originally specified in the model, as was the exhaust manifold collector. The port sizes on the engine cylinder head also remain as specified in the original model. To keep the number of variables relatively low, all four intake runners are assumed to be identical as are the four exhaust runners.

The optimisation goal is for maximum engine power produced at a fixed engine speed. The WAVE model is only run at a single speed, as adding multiple cases dramatically increases the computation time. Since the purpose of this experiment is to observe the SGA rather than to actually optimise an engine, the running of a single case model is not considered to be a problem.

The SGA is set up to run for 80 generations with a population of 150 creatures. The mutation rate is 0.05 and the Tournament Selection probability is 0.75.

In total, four optimisation runs were completed. The first three are at 2500, 4500 and 6500rpm respectively with Engine Brake Power (kW) as the fitness measure. For the final run, a multi-case model is run at all three speeds with the fitness measured as the average brake power produced.

For each of the single speed optimisations, the models were also optimised by hand using the basic tools provided with the WAVE package. First the user provides the WAVE software with a min, middle and max value for each of the variables. WAVE will then run the model with each possible combination of these values and make estimates as to how the engine will respond as they are changed. The user can then move a series of sliders to adjust the input variables and gain an idea of how the engine output will change. Unfortunately with 11 input variables this process is not very efficient and finding a good solution is difficult.

The second set of experiments is designed to test the SGA and the VLCGA on problems with very large numbers of variables. This added complexity means optimisation by a human designer is impossible, however there is the potential for the GAs to discover unconventional solutions to the problem. For these experiments both the binary chromosomes and real valued chromosomes were tested.

The same 4-cylinder WAVE model from the first experiment is used, however the number of variables is increased dramatically. Three configurations are tested, the first is about as complex as is possible with the standard WAVE software, and has 94 variables. The second is a medium complexity arrangement with 54 variables, and the final simple configuration has just 20. These variables specify various lengths, diameters and volumes throughout the intake section the engine. It is expected that the Simple GA will have difficulty optimising the more complex configurations within a reasonable number of function evaluations, however the VLCGA should be able to omit a number of variables, effectively reducing the complexity in places where this added complexity does not give any benefit. The very simple configuration should be fairly straight forward to optimise, however it is expected that the best solutions for this model will have a lower fitness than those from the more complex models.

The optimisation goal is again Engine Brake Power (kW), this time at an engine speed of 6000rpm. Initially both the SGA and VLCGA are set to run with 100 creatures for 100 generations.

IV. Results

Since the GA works solely on the fitness values and has no preconception of what a good solution may be, it is quite likely that the algorithm will produce unusual or unexpected solutions to the given problem. Some simple design guidelines that a human designer may follow for intake and exhaust systems are:

- Pipes for different cylinders being of equal length
- Constant tube diameters, or only gradual tapering
- Shorter runners for high rpm power output
- Longer runners for low rpm power output

It is expected that the primary change between the runs at the different speeds will be the length of the intake and exhaust runners. As mentioned previously, conventional engine design practice would suggest that higher power would be achieved with shorter intake and exhaust runners at higher engine speeds and longer intake and exhaust runners at lower speeds. This reduction in length at high speeds allows reflected high and low pressure pulses within the manifolds to coincide with valve opening times.

Table 1. summarises the results for the single speed runs. The 'Initial Config' values were measured with a non-optimised intake and exhaust. This arrangement was the same for all speeds and consisted of a 40mm intake diameter by 750mm total length and 36mm exhaust diameter by 600mm total length.

TABLE I. Optimisation Results

Engine Speed (rpm)	**Brake Power Produced (kW)**		
	Initial Config	***Manually Optimised***	***SGA Optimised***
2500	24.04	32.4	38.23
4500	45.51	60.5	70.9
6500	57.65	74	84.61

The fitness of the SGA optimised solutions are consistently higher than those found with the manual optimisation process. Even when it was known that power outputs as high as the SGA solutions were possible, attempts to produce these by manually adjusting the parameters were unsuccessful.

It was noted that the solutions found by the SGA did not have particularly smooth torque curves across the full speed range of 1000 – 7000rpm, instead large spikes are seen at the speeds used for the fitness calculation. While this is not a particularly desirable characteristic in an engine design, it is to be expected since the GA was given the task of optimising the engine for that speed only.

When the solutions found by the SGA are sketched out, a consistent decrease in length is observed as the engine speed increases, with the multi-case run apparently preferring the longer design. It is also interesting to note that the volume of the exhaust runners seems to be steadily decreasing as the rpm increases.

Another observation from the first experiment was the development in some of the runs of two quite different solutions with similar fitness values. This suggests that the SGA is finding local optima rather than the overall global optimum, or possibly that there is no global optimum. It is certainly conceivable that there may be two or more equally good solutions to the problem at hand. Unfortunately the only way to know for sure would be to perform an exhaustive search and evaluate every possible combination of input values. Unfortunately for problems of this size it is simply not feasible due to the length of time required to evaluate so many solutions.

The second set of experiments is still underway however some initial results are available at the time of writing. All three of the model configurations are representations of the same engine, so the maximum possible power output is the same for all of them. As has been mentioned previously, the global optimum is unknown, however the best solution found so far produced 82.76kW. This was found using the SGA on the medium complexity model. Other runs using the SGA on this model produced solutions very near this value, indicating that this may be near to the best solution.

To asses the performance of the different chromosome encodings, all four GA variants were applied to the medium complexity model. The SGA with binary chromosome performed consistently better than the SGA with a real valued chromosome. The real valued encoding generally reaches a moderately good solution very quickly, however the best fitness obtained with this implementation was only 79.9kW. On the other hand, the binary encoding takes slightly longer to reach good solutions, but a peak fitness of 82.76kW was achieved using this genome encoding. Since the binary coded SGA was observed to be consistently better than the real valued version, there was no further investigation of the real valued SGA.

When comparing the real valued VLCGA and binary encoded VLCGA however, the result was quite different, and there appears to be very little difference in performance between these two algorithms. This suggests that the function of the cut and splice operators is not significantly affected by the use of real valued encoding, however the operation of the conventional single point crossover, as used in the SGA, is.

As expected, the binary coded SGA did not perform very well when applied to the very complex model. The best solution produced only 79.6 kW, which suggests that the large number of variables in the complex model does reduce the effectiveness of the SGA. An increase in the population size should allow the SGA to find better solutions to the more complex problem; however the associated increase in computation time has prevented further investigation in this study.

The first tests with the real and binary coded VLCGA were made using initial chromosome sizes of 5 genes. The use of the splice operator allows these small genomes to grow rapidly over the first few generations, however the solutions which the algorithm found were poor. The best solutions using this method produced between 75 and 76kW. It appears that due to the 'messy' nature of the VLCGA and the way the cut and splice operators function, the sizing of the initial population is critical to the functioning of the algorithm. In order for acceptable results to be produced, the initial population must contain most, if not all, of the possible gene values. Goldberg was clearly aware of this fact and his MGA described in [10] initialises by generating every possible combination of genes. With the initial genome size increased to 100 genes, the results from the VLCGA were much more encouraging.

While only a few runs have been completed using the VLCGA, it appears that performance on the medium complexity model is slightly worse than that of the binary coded SGA. The best solution discovered so far produces 81.2kW. On the very complex model, performance appears to be slightly better than the binary SGA with the best solution so far being 80.4kW. It is difficult to draw many conclusions from these results as at the time of writing there have been very few repetitions of the experiment. It is hoped however, that fine tuning of the VLCGA parameters will yield further improvements.

V. CONCLUSIONS

In the first set of experiments, the SGA did successfully find good solutions for each of the problems given, and these solutions produced a significantly higher power output than was achieved through a basic manual optimisation procedure. There were also some patterns and common design elements which could be observed in the solutions as the engine speed was increased.

Most of the designs produced by the SGA could be considered 'unconventional' and it would be worth while to establish what underlying mechanisms they are exploiting to gain such high power outputs. The general shape of the exhaust solutions appears similar to the 'expansion chambers' often utilised on two-stroke engines and it is possible that there are similar principles underlying the SGA designed solutions. Further investigation will need to be carried out in this area to establish if this theory is correct.

While the SGA has shown it is capable of finding good solutions, there is evidence to suggest that there are local optima in the fitness landscape. The shape of the intake ducts in particular indicate that there were at least 2 local optima, as the SGA did not converge to the same solution each time. It is possible that by simply running the optimisation algorithm for more generations, with larger population sizes, it will eventually converge to a global optimum, however it is difficult to know for sure when there is so little information available regarding the search space.

The solutions which were found by the SGA did meet the criteria specified in the fitness function; however the design of the fitness function will need to be reconsidered if more practical solutions are to be found. While single rpm optimisations may have some application in areas such as stationary power generation, for the majority of potential applications, a solution which only gives good performance at a single engine speed is not very useful. This indicates that multi-case WAVE models will be needed, in conjunction with carefully chosen fitness functions, to discourage this behaviour.

The side effect of needing to run multi-case models is that evaluation of the models takes significantly longer. With the number of creatures and generations currently required by the SGA the length of time required to complete an optimisation run will become prohibitive, and this highlights the need for a more efficient searching algorithm.

The second set of experiments compared the SGA and VLCGA with both binary and real valued genomes. Here it was observed that the cut and splice operators used in the VLCGA operate quite differently from the single point crossover used in the SGA. The SGA crossover operator does not perform well when used with a real valued genome, while the performance of the cut and splice operators does not seem to be affected by the genome encoding used. There are several possible explanations for this and they are briefly discussed below.

While the actions performed by cut and splice and single point crossover appear to be quite similar (for long chromosomes, the cut and splice swaps the end section of one chromosome with the end section of another), due to the 'messy' encoding of the chromosome they are actually doing very different things. The single point crossover is always exchanging equivalent genes, they will usually have different allele values, but they are coding for the same 'feature'. The cut and splice operator, however, does not exchange similar genes, it can be removing one set of features and replacing them with an entirely different set.

Single point crossover allows genes to be split in the middle and rejoined with a partial gene from the second creature. This enables new allele values to be 'discovered' that were not previously expressed in any creature in the population. This appears to be a critical part of the operation of single point crossover as when this ability is removed (as in the real valued encoding) the performance of the SGA clearly suffers. Cut and splice on the other hand showed no improvement when the binary genome encoding was used, which would indicate that being able to discover new allele values is either not of any benefit, or does not occur often enough in the process to be of any benefit.

This theory is backed up by the observation that if a VLCGA is not initialised with long enough chromosomes, the performance suffers. This would suggest that there must be a sufficiently large pool of genetic material to begin with, as the cut and splice operators do not produce a useful number of new allelic values during the course of the optimisation. When a sufficiently large initial pool is available, such as with the 100 gene starting chromosomes, the algorithm performs quite well even when a real valued encoding is used.

With regard to combustion engine gas-exchange systems, it is clear that GAs show great potential as an optimisation tool. If further work with the VLCGA on the very complex model can produce a similar level of performance to the SGA on the medium complexity model, then the technique may prove to be a very valuable tool for engine designers. At this early stage however, it is difficult to draw any meaningful conclusions as the experiments with the VLCGA are only just beginning.

Future work on the project will involve the continuation of the experiments using the binary SGA and VLCGA on WAVE models of varying complexity. More detailed analysis of both the solutions and the operation of the GA variants will be carried out to hopefully learn more about the operation of the algorithms and the nature of the design space they are searching in. One aspect of the GAs which has not been investigated in this study so far is the role of the mutation operators. It is likely that fine tuning of these and other algorithm parameters could show further improvements from both the SGA and the VLCGA.

ACKNOWLEDGMENT

We would like to thank Ricardo for the use of their WAVE software.

REFERENCES

[1] W. Zucker, R. R. Maly, and S. Wagner, "Evolution-strategy based, fully automatic, numerical optimisation of gas-exchange systems for IC engines," SAE 2001-01-0577, 2001.

[2] M. A. Rosenman, "An exploration into evolutionary models for non-routine design," Artificial Intelligence in Engineering, vol . 11, pp. 287-293, 1997.

[3] M. Mitchell, An introduction to genetic algorithms, MIT Press, Massachusetts, 1996.

[4] M. Ahmadi, "Intake, exhaust and valve timing design using single and multi-objective genetic algorithm," SAE 2007-24-0090, 2007.

[5] M. Kim, M. P. Liechty and R. D. Reitz, "Application of micro-genetic algorithms for the optimisation of injection strategies in a heavy-duty diesel engine," SAE 2005-01-0219, 2005.

[6] A. Munnannur, S. Kong, and R. D. Reitz, "Performance optimisation of diesel engines with variable intake valve timing via genetic algorithms," SAE 2005-01-0374, 2005.

[7] Y. Shi and R. D. Reitz, "Assessment of optimisation methodolgies to study the effects of bowl geometry, spray targeting and swirl ratio for a heavy-duty diesel engine operated at high-load," SAE 2008-01-0949, 2008.

[8] C. Branney, G. Cunningham, S. Spence, and G. McCullough, "Development of optimisation techniques for the design of an internal combustion engine airbox," SAE 2006-32-0114, 2006.

[9] J. Lampinen, "Cam shape optimisation by genetic algorithm," Computer-Aided Design, vol. 35, pp. 727-737, 2003.

[10] K. Deb and D. E. Goldberg, "mGA in C: A messy genetic algorithm in C," IlliGAL Report No. 91008, 1991.