Biogeography-Based Optimization of a Variable Camshaft Timing System

George Thomas
Cleveland State University

Daniel J. Simon
Cleveland State University, d.j.simon@csuohio.edu

John Michelini
Ford Motor Company

Biogeography-based optimization of a variable camshaft timing system

George Thomas [a], Dan Simon [a,*], John Michelini [b]

[a] Department of Electrical Engineering and Computer Science, Cleveland State University, Cleveland, OH 44115, United States
[b] Ford Motor Company, Dearborn, MI United States

ARTICLE INFO

Article history:
Received 2 March 2015
Received in revised form
29 May 2015
Accepted 13 July 2015
Available online 7 August 2015

Keywords:
Computationally expensive optimization
Biogeography-based optimization
Automotive system optimization
Design and analysis of computer experiments

ABSTRACT

Automotive simulations often prohibit the use of traditional optimization techniques because these simulations are complex and computationally expensive. These two qualities motivate the use of evolutionary algorithms and meta-modeling techniques respectively. In this work, we apply biogeography-based optimization (BBO) to optimize radial basis function (RBF)-based lookup table controls of a variable camshaft timing system for fuel economy. Also, we reduce computational search effort by finding an effective parameterization of the problem, optimizing the parameters of the BBO algorithm for the problem, and estimating the cost of a portion of the candidate solutions in BBO with design and analysis of computer experiments (DACE). We find that we can improve fuel economy by 1.7% over the original control parameters, and we find a tradeoff in population size, and an optimal value for mutation rate. Finally, we find that we can use a small number of samples to construct DACE models, and we can use these models to estimate a significant portion of the candidate solutions each generation to reduce computation effort and still obtain good BBO solutions.

1. Introduction

Internal combustion engine (ICE) variables have been optimized for over half a century, although the optimization approaches and objectives have changed over the years in response to the changing environment in which these engines are used. Early work in ICE optimization includes an application of control systems theory in which the spark timing and air intake throttle are manipulated as inputs to optimize the output brake mean effective pressure (BMEP) (Draper and Li, 1951). The oil embargo to the USA in 1973 (Society of Automotive Engineers, 1976) and the subsequent corporate average fuel consumption (CAFE) standards (Congress of the United States: Office of Technology Assessment, 1991) set by the US government shifted the auto industry's focus toward optimizing ICEs for fuel economy, instead of engine power output as in (Draper and Li, 1951). The reduction of exhaust emissions formed into an objective at about the same time, starting with 1966 regulations in California (Ribbens, 1984). Further, the introduction of the microprocessor into engine control in the middle 1970s greatly simplified tuning of engine performance over previous analog, mechanical engine control, and thus facilitated further ICE optimization (Ribbens, 1984).

Since the 1970s, there has been an explosion in the application of new engineering techniques such as standard and multi-objective evolutionary algorithms (EAs) (Zhao and Min, 2013), (Kim et al., 2005) and artificial neural network-based surrogate models (Wu et al., 2006) to the optimization of different ICE actuator variables such as variable valve actuation (VVA), spark angle, and air-fuel (A/F) ratio (Sellnau and Rask, 2003; Zhao and Min, 2013) for fuel economy and emissions.

EAs are robust global optimizers and only require ways of evaluating the fitness of solutions, instead of differentiable or analytical models for a given problem. The robustness of EAs makes them attractive options for complex, nonlinear, multimodal, or black-box optimization problems like ICE optimization, and thus we have chosen an EA called biogeography-based optimization (BBO) for this problem.

BBO is an EA inspired by biogeography, which is the study of the migration of species between habitats (Simon, 2008). The novel evolutionary operator in BBO is migration. With migration, each solution in the population is given an immigration rate and an emigration rate based on its fitness; these migration rates determine the likelihood that a solution will give or receive features from other solutions in the population.

We are primarily interested in research on our automotive problem and applications of general EAs to this problem, instead of pure EA research. Because of this, we have chosen to use BBO as a typical EA. We justify our use of BBO by noting that it outperforms many EAs on a variety of benchmarks (Simon, 2008), and some promising theoretical research has been done to support it, including an analysis of BBO's migration models (Ma and Simon, 2011), a theoretical comparison between BBO and GAs (Simon et al., 2011b), and a probabilistic study of BBO (Simon, 2011a). BBO has also been successfully applied to a variety of nonlinear control problems such as fuzzy logic-based robot path tracking control optimization, open-loop knee prosthesis control, and load dispatching in power systems (Thomas et al., 2011; Wilmot et al., 2013; Bhattacharya, 2010), and so it is reasonable to expect it to perform well for ICE control optimization.

Automotive simulations are often computationally expensive, and if they are to be used as part of a cost function for an EA, they must be run many times. This fact poses a problem of particularly high computational effort. One solution is the use of surrogate models of the cost function, such as Design and Analysis of Computer Experiments (DACE) (Simon, 2013). DACE is a promising Gaussian process surrogate modeling method that applies the algorithm of kriging to the estimation of deterministic computer experiments. DACE has been applied to engineering meta-modeling (Wang and Shan, 2007) and reduction of fitness function evaluations in EAs (Jin and Branke, 2005), and the variety of EA applications of DACE has inspired us to use it for our problem.

Further, the choices of EA parameters, (i.e., population size) has an effect on the performance of the EA. For instance, a sensitivity analysis of the parameters of a particular GA on two different scheduling problems shows that crossover rate is the most important for one problem, and population size is the most important for the other (Pinel et al., 2011). This problem-dependent sensitivity motivates us to optimize the EA parameters for our particular EA and problem, because doing so will allow us to find better solutions with less search effort.

Our main contribution to the topic of engine control tuning is the development of cam timing control tables that result in a 1.7% improvement in fuel economy over the tables that we started with. This fuel economy improvement has important implications that are explored in Section 4. Also, our use of DACE to reduce EA computational effort for automotive system optimization, and our use of subfunctions of a vehicle drive trace may both be novel, because we have not seen these particular research items discussed in the literature.

The remainder of this paper is organized as follows. In Section 2, we explore our formulation of the problem, and how we choose to construct a parameterization for use with BBO and our application of the DACE algorithm to BBO; in Section 3, we show our development of effective BBO and DACE parameters for the cam timing optimization problem; in Section 4, we examine BBO solutions to the VCT problem; and finally, we draw conclusions and suggest future work in Section 5.

2. Problem formulation and related work

The Simulink® vehicle simulation that we use throughout this work is of a passenger vehicle equipped with a turbocharged gasoline powered direct injection engine, and includes variable timing of both intake and exhaust camshafts. Although there are a variety of control systems that we can optimize in this vehicle, we restrict our attention to the variable camshaft timing (VCT) system. VCT is a particular type of VVA technology in which the timing of intake valves, exhaust valves, or both are manipulated by changing the angular position of one or more camshafts with respect to the crankshaft.

Our objective is to optimize the fuel economy of a simulated vehicle by adjusting the control of the vehicle's VCT. Specifically, we make modifications to lookup tables that define the controller set points for the two actuators in this system – the independent variables used for these lookup tables are engine speed and engine load, which can be considered the state variables of the ICE system (Meyer, 2007). Further, we take three approaches to improve the performance of BBO for the problem of optimizing the fuel economy of simulated internal combustion engines via adjustments of intake and exhaust cam timing. The first of these three meta-optimization approaches is to make adjustments to the problem formulation to make the problem more conducive to BBO; the ways we can accomplish this include choosing a parameterization of the optimization problem that is best suited for BBO and adjusting simulation parameters, such as driving conditions. The next approach is to apply modifications to BBO to improve its performance in light of the computationally intensive nature of the ICE fuel economy problem, and we use surrogate modeling with DACE to accomplish this. The third approach is to find optimal values of the standard BBO parameters that result in the fastest convergence to an optimum solution.

The simulation that we use is of moderate fidelity, in that it uses simplified models of combustion and engine flow, and is intended for preliminary control system parameter optimization. We choose to use this simulation model because preliminary parameter optimization with an EA is often an interactive process that requires running the simulation many times, and to do so using a high fidelity model would be prohibitive in terms of CPU time. In order to illustrate what systems and subsystems of the vehicle we simulate, we provide a simplified block diagram of our Simulink software in Fig. 1.

The intake and exhaust camshaft timing actuators in our VCT system are controlled via lookup table mappings that cover a limited domain of engine load and engine speed. The reference values used for controlling the camshaft timing actuators are generated at runtime by linearly interpolating between adjacent table values. We have chosen to modify the tables produced via prior research. These tables are shown in (Thomas, 2014). We have chosen to modify these tables instead of generating completely new tables, because the tables we have chosen to modify produce good results in simulation tests, despite being suboptimal. We have chosen this strategy because using BBO to optimize modifications to these tables, rather than using BBO to create new tables, effectively reduces the size of the search space and takes advantage of prior expert knowledge.

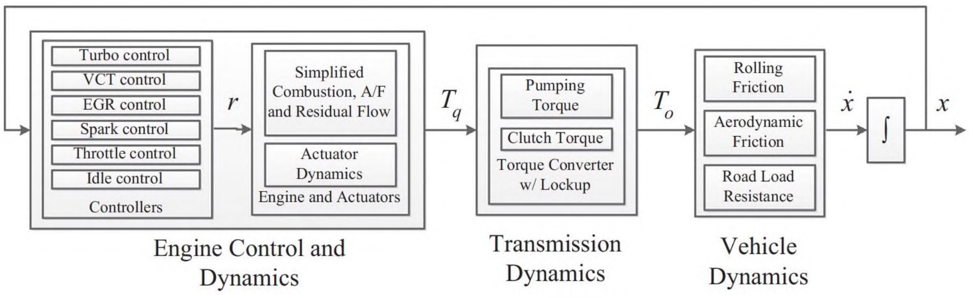

Fig. 1. Block diagram of Simulink vehicle simulation, showing subsystems and quantities that are taken into account. r is the collection of actuator target (set-point) signals (note that there are closed-loop controllers within the block labeled “Engine and Actuators” that are not depicted) T_q is the torque output of the engine, T_o is the torque supplied to the wheels, x is the state of the vehicle, including states of the included subsystems, and $\dot{x}$ is the derivate of the vehicle state.

Further, we have chosen to parameterize these modifications (which are arbitrary mappings in (load, speed)-space) with Gaussian radial basis functions (RBFs) in order to formulate the lookup table optimization problem as a parameter optimization problem for use with BBO. We choose Gaussian RBFs because each basis in such a RBF-based mapping contributes locally about its center, while still producing a continuous mapping when they are summed. As a result, RBF-based mappings are intuitive, since it is easy to visually examine the numerical optimization results. Further, depending on which parameters of the optimization problem are fixed and which are allowed to vary as independent optimization variables, the problem dimension can be made relatively small while still retaining a great deal of flexibility in the variety of solutions that can be represented.

We generate RBF-based lookup-tables by sampling all of our RBFs at all of the (load-speed) points specified by the elements of the original lookup tables, and we add the sampled RBF values to the original tables at these points. Finally, we set any resulting lookup table values outside the range that the actuators can produce, $[-27, 50]$, to the nearest minimum or maximum. This approach optimizes the parameters (lookup tables) of a specific control structure. This is a typical case in industry, where the same control structure is applied in multiple instances, each with its own lookup table values or calibration parameters.

2.1. RBF parameterization approach

The parameterization approach we have chosen is to fix the locations (μ) and the shapes (the widths, σ and the correlation, ρ) of the RBFs, while optimizing the heights or magnitudes. The choices of these fixed parameters itself is an optimization problem. Our approach to this optimization step is to chose these parameters such that the resulting RBFs span the trajectory of the simulated vehicle through (load, speed)-space given the original actuator lookup tables.

In this method, we chose a fixed number of RBFs to span (load, speed)-space, given a subjective inspection of the number of clusters of data in the simulation trajectory, and we fix the heights of these RBFs to unity. We then manually adjust the locations and shapes of the RBFs until the level curves of the unity-height RBFs visually correlate with the data from the nominal simulation.

We choose to use the five clusters in our formulation, based on results of ad hoc BBO runs. We were able to find better solutions using a five cluster formulation than we could with the three and 10 cluster formulations we tried, which suggests that five clusters provides a good tradeoff between search space manageability and parameterization flexibility.

Fig. 2 shows the locations and shapes produced via the manual pre-BBO step using five clusters of (load, speed) data; Fig. 3 shows filled contour plots of the surface made by the sum of the RBFs with their heights set to unity. These plots are useful for this design process, because they qualitatively show us what areas of the controller lookup tables we can adjust with BBO and how well these correlate to the (load, speed) subspace that the vehicle traverses. The centers of the RBFs are given in captions of Fig. 2 and Fig. 3. Note that the locations of the centers given in these figures do not directly correspond to the RBF parameters used with BBO, since these parameters were manually tuned after the RBF placement procedure to further to fit the unity height RBF surfaces as shown in Fig. 3 to the scatter plots of simulation data as shown in Fig. 2. Future work may involve applying a more systematic optimization approach to fitting the shapes and locations of the RBFs to the simulation data.

Finally, we note we arbitrarily choose to use a search range of $[-20, 20]$ for the RBF heights with BBO. This range was chosen because it is a round number that is roughly one quarter of the full

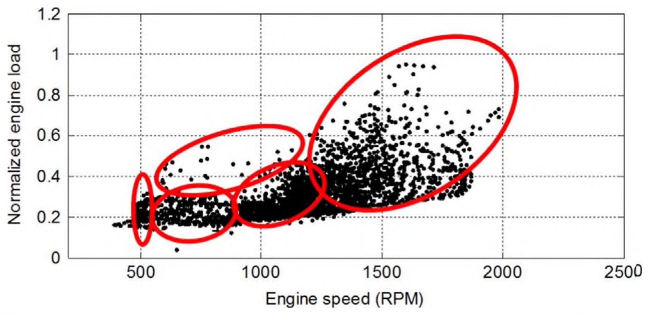

Fig. 2. Engine load and speed data scatter plot with circles showing the locations and shapes of five RBFs found with an ad-hoc placement approach. We note that the first RBF is centered at (525, 0.21), the second, third, fourth, and fifth are centered at (700, 0.15), (850, 0.45), (1200, 0.30), and (1600, 0.65).

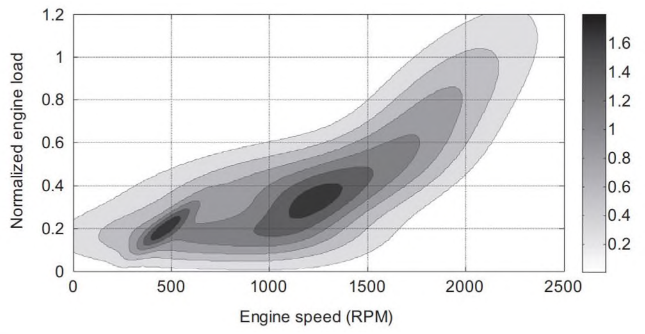

Fig. 3. Contour plot of the RBF surface created by setting the RBF heights to unity, in the case of five clusters.

range of camshaft phase angles, $[-50, 27]$, that can be produced by the actuators that we are controlling. Larger search ranges could be used, but the slow dynamics of the actuators require that difference between adjacent camshaft angle values in the lookup tables be small, and giving BBO the freedom to make drastic changes to the tables may encourage BBO to produce solutions that include large changes between adjacent table values. This constraint is important, because if the engine control unit (ECU) constantly applies drastic changes to the set points of the camshaft timing actuators (which may occur during slow, stop-and-go traffic), the actuators may not be able to reach their set points. This may reduce the improvements in efficiency that we can obtain with VCT, and may result in an engine operation that is undesirable to the driver, as it may constantly and abruptly change its performance characteristics. Future work may include adjustments of the search ranges, and formulating the problem as a constrained one.

2.2. Simulation drive traces

The choice of simulation parameters can have a drastic effect on the search effort needed to obtain good solutions, therefore, we need to closely examine the vehicle velocity profiles that we evaluate the vehicle simulation over. We have chosen to use the EPA urban dynamometer driving schedule (UDDS), also known as the LA4 cycle (US EPA, 2012). We have also developed a method to pick sub-traces from a given drive trace. Any particular drive trace can be split into a sequence of sub-traces that each begin with a period of idle (vehicle velocity equal to zero) and end with the next period of idle. For instance, the LA4 city trace used in our work can be split into 18 different sub-traces, including one sub-trace that consists only of a period of idling. A plot of the LA4 drive trace with vertical lines denoting the beginning of each sub-trace is shown in Fig. 4.

The sub-trace approach is a way of extracting sub-functions of the cost function as mentioned in (Simon, 2013). The details of this approach (especially initial velocity, $v_0=0$) are important because they ensure that the initial conditions of each sub-trace are consistent.

We can determine which sub-traces to use for approximating the full fitness function trace by examining quantities such as root-mean-square (RMS) velocity and acceleration, the time length of the sub-trace (proportional to the number samples in the drive trace, since the traces are sampled at a constant rate), and the empirically measured CPU time needed to run the simulation over that sub-trace. Afterwards, we can determine which of these quantities are most strongly correlated with the minimum cost of a BBO run to decide which sub-traces to use.

To execute this test, we ran three Monte Carlo simulations of BBO over each of the 18 sub-traces and evaluated the sub-traces at the four previously mentioned quantities. In order to evaluate how well a sub-trace represents the full LA4 city trace and to provide a basis for comparisons between sub-traces, we evaluate all of the BBO solutions from each run on the full trace during the final generation. In other words, the cost (fuel economy) values that are discussed in this section are all evaluated on the full city trace, though the candidate solutions were evolved using a particular sub-trace. The BBO parameters for this test are given in Table 1. We note that population size was chosen to be a multiple of 12 to reduce overhead, as we run fitness function evaluations with 12 MATLAB® parallel workers on a 24-core AMD Opteron-based™ PC. Also, the number of Monte Carlo simulations was chosen arbitrarily as part of a tradeoff between computational effort and collecting a significant sample size.

Pearson correlation coefficients between minimum cost and each of the metrics, as well as the associated no-correlation probabilities from these Monte Carlo trials are shown in Table 2. Specifically, the data in the table are the averages over all sub-traces and over all Monte Carlo simulations (i.e., 18 sub-traces • 3 Monte Carlo trials=54 simulations). We have chosen to use the Pearson

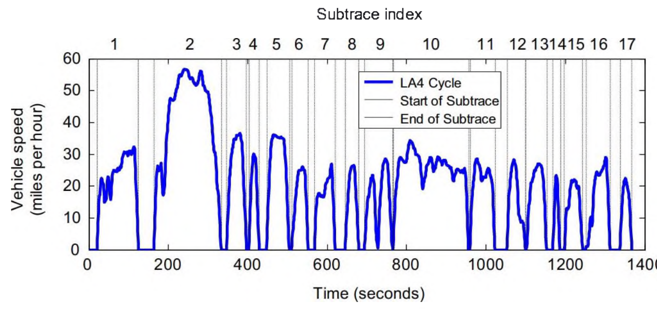

Fig. 4. Plot of EPA urban drive cycle with vertical lines showing how we have partitioned it into sub-traces. An index is denoted above each sub-trace, and these are referred to in the remainder of this paper. Not shown in this plot is the 18th sub-trace which consists of all of the idling time between each of the other sub-traces.

Table 1
BBO parameters for the Monte Carlo simulations where we examine correlations of various quantities with fuel economy for different sub-traces.

Parameter	Value
Generation count	20
Population size	48
Mutation probability	0.02
Problem dimension	10
Number of elite solutions	2

Table 2
Correlations between minimum cost after a BBO run and the following metrics; RMS velocity, v_{RMS}; RMS acceleration, a_{RMS}; simulation time, T_{sim}; and CPU time, T_{CPU}. Also shown are the associated probabilities that there is no correlation between each metric and final minimum cost.

Metric	Pearson's ρ:	p-value (H_0)
vRMS	−0.6851	0.0017
aRMS	−0.5299	0.0237
Tsim	−0.1907	0.4485
TCPU	−0.1995	0.4273

correlation coefficient because straight lines, and thus linear relationships, can be fit to these data most easily.

Given a confidence interval of 5%, the two metrics that have statistically significant correlations are RMS acceleration and especially velocity. Further, the fact that all of the significant correlation coefficients are negative indicates that running BBO over sub-traces that measure higher than average in these metrics is more likely than average to result in better solutions. Since there is little evidence supporting a correlation between simulation or CPU time and minimum cost, we should be able to run for a relatively short simulation time and still get good BBO results, and it does not really matter how much CPU time it takes (which may vary depending on how numerically stiff the problem dynamics turn out to be, given the shape of the sub-traces).

Finally, we directly examine the minimum cost after running 20 generations of BBO. The sub-trace that resulted in the lowest costs after running BBO is sub-trace 2. Further, running BBO with the full city trace results in minimum costs almost identical to those found by running BBO over the second sub-trace. Further, we also examined the cost after running various combinations of sub-traces (i.e., adding up the fuel consumed by running over both sub-traces 2 and 9). We found that these combinations generally give poorer costs than sub-trace 2.

The conclusion we can draw from these data is that it is better to run over sub-trace 2 than any other individual sub-trace, any of the combinations of sub-traces that we have tried, or the full drive trace itself. This is because, on average, BBO finds solutions with comparable performance when computing cost over sub-trace 2 and the full city trace (i.e.; the differences in best costs found with both are insignificant), yet the simulation time of the full city trace is 1371 seconds long, whereas sub-trace 2 is only 174 seconds long. This means that, by running sub-trace 2 instead of the full trace, we can reduce simulation time, almost by a factor of 8, yet still find the best solutions possible given all of the drive traces that we have evaluated. One thing that should be kept in mind, is that these cost data are computed over the full drive trace as a means of comparison, however, this results in a selection bias where the solutions that are considered best are the ones that result in less fuel consumption over the particular driving conditions of the full drive trace, and so we again state that we assume the driving conditions of the city trace represent the average driving conditions faced by the public who would be driving this kind of car.

2.3. DACE application to BBO

Because DACE has been applied to other EAs before, it makes sense that it can be applied to BBO as well. Although there are several evolution control strategies (i.e., several algorithms for deciding when to estimate cost with DACE instead of calculating it) (Jin and Branke, 2005), we have chosen to develop our own strategy for using DACE with BBO to leverage the record of evaluated candidate solutions that we keep as a pool of samples for generating DACE models.

During each generation, we determine whether to estimate or calculate the cost of a candidate solution by comparing the Euclidean distance between the solution and the closest DACE sample point to it in search parameter space. Two strategies for making this decision include picking a fixed number of candidate solutions to estimate and choosing those who are closest to their nearest sample point, and estimating only those solutions that are closer than a given distance threshold to their nearest sample points. Fig. 5 illustrates the concept of the closest distance from a point that we are estimating and the nearest DACE sample to it, in a hypothetical, one-dimensional case. The distance threshold strategy makes sense because the mean-squared error (MSE) of a DACE model is zero at the sample points used to fit the model, and generally increases as one looks further and further away from the sample points (Jones et al., 1998). The main drawback to this method is the challenge associated with choosing the distance threshold given how aggressively we want the EA to estimate the cost function. We can alternatively use the MSE expression from (Jones et al., 1998) instead of distance to decide which solutions to estimate.

The set of samples chosen to generate a DACE model also strongly affects the performance of an EA using DACE. Latin hypercube sampling is often used for fitting DACE models (Jones et al., 1998), because a Latin hypercube sample set can better represent a distribution than a uniform sample set, given certain conditions (McKay et al., 1979). In contrast, building a DACE model from normally distributed samples is a bad idea, because the samples will tend to be close together, and the correlation matrix, R, which must be inverted in the process of fitting or sampling from a DACE model, becomes nearly singular (Jones et al., 1998). To see this, we show the DACE formula used to estimate the fitness function at point x^* in the problem domain.

$$\hat{f}(x^*) = \mu + r(x^*)^T R^{-1}(f(x) - \mu 1_M) \quad (1)$$

Note that $r(x^*)$ is the vector of correlations between the point at which we want to estimate the cost function and the samples used to generate the DACE model, R is the matrix of correlations between the DACE samples, μ is the constant term of the DACE model, M is the number of samples used to generate the DACE model, and 1_M is an M element vector containing all ones. If there are samples that are very close to one another, R will become nearly singular and thus pose numerical problems. This motivates us to find a sampling heuristic that results in well-conditioned DACE models.

In order to obtain sample sets for use with DACE, we have developed a simple, sub-optimal sampling heuristic that selects samples from a record of evaluated candidate solutions, one-by-one, such that the resulting sample set is well spread out. This record is populated by with all of the candidate solutions whose cost has been evaluated during a particular BBO run. In our heuristic, the first sample is chosen as the solution closest to the mean of the record solutions, and the subsequent samples are chosen as those furthest away from the mean of all of the

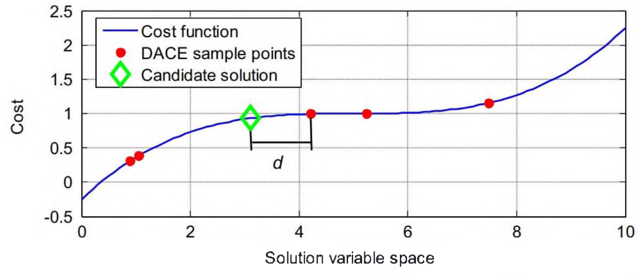

Fig. 5. Drawing depicting the distance, d, between a solution to be estimated and its closest DACE sample.

previously chosen samples. Algorithm 1 provides a pseudocode description of this "spread out" sampling heuristic.

Algorithm 1. Pseudocode representation of our DACE sampling heuristic, where N is the size of the solution record, M is the desired number of samples, x is the vector of solution vectors in the record, $\bar{x}$ is the mean of the recorded solutions, y is the resulting set of DACE samples, g is the set of indices corresponding to previously chosen record solutions and is used to implement sampling without replacement, and $i_{\min}$ and $k_{\max}$ are the indices that optimize their preceding equations.

```
$y_1 = \underset{i=[1,N]}{\operatorname{argmin}} \|\bar{x} - x_i\|$
$g_1 = i_{min}$
for $j = 2$ to $M$
    $y_j = \underset{k=[1,N] \notin g}{\operatorname{argmax}} \|\bar{y} - x_k\|$
    $g_j = k_{max}$
end
```

We also note that, in addition to our sampling heuristic, we use the pseudoinverse and pseudodeterminant instead of the classical algorithms to reduce numerical difficulties when fitting DACE models. We also formulate the model fitting process as log-likelihood maximization instead of normal likelihood maximization, as this effectively increases the dynamic range of likelihoods that we can compute during the fitting process.

3. Optimization parameter studies

It is important to examine the characteristics of the optimization method chosen for any particular problem, since one algorithm or one set of optimization algorithm parameters may be better suited for a given problem than others. This is a consequence of the No Free Lunch theorem, which states that all algorithms perform equally well on average when tested on the most general class of problems (De Jong, 2007). This motivates us to study the DACE and BBO parameters that we use for our variable camshaft timing problem, since an algorithm that takes advantage of problem specific knowledge may perform better on that particular problem than algorithms that do not.

3.1. BBO parameter studies

In order to find effective BBO parameters for our particular problem, we first define a reference set of BBO variables which are shown in Table 3. We then vary parameters such as population size and mutation rate, one at a time. We note that we ran these tests on a single arbitrarily chosen sub-trace to facilitate the Monte Carlo approach, since running a BBO simulation over even a single sub-trace can take more than 3 hours depending on the BBO parameters.

First, we examined population size by running eight Monte Carlo simulations with population size equal to the values: 25, 50, 75, 100, 150, 200, and 300. The cost of the best solution from each Monte Carlo simulation is represented by a point on a scatter plot in Fig. 6. See Thomas (2014) for standard deviation data.

Because this relationship appears to be a monotonic one (as one would expect), there is no optimum and so we must find a suitable tradeoff instead. It appears that we reach a point of diminishing returns after population size increases past 200, so this may be a useful value for population size, especially considering that we can estimate the cost of many candidate solutions with a DACE model. This implies that up to 200 parallel processors

Table 3
Nominal BBO parameters for BBO parameter studies.

Parameter	Value
Generation count	50
Population size	100
Mutation probability	0.02
Problem dimension	10
Number of elite solutions	2
Simulation drive trace	sub-trace 10

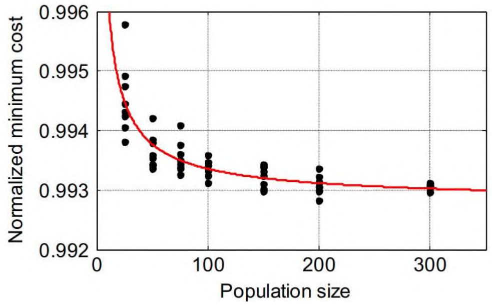

Fig. 6. Scatter plot of normalized minimum cost after 50 BBO generations versus population size. A power function curve is also shown.

would be an attractive setup for a BBO run, but any more than that would be beyond the point of diminishing returns.

In order to quantitatively gauge the relationship between population size and the fitness of the best solution after a BBO run, we compute the Spearman rank-based correlation coefficient, ρ. Spearman's correlation coefficient is useful for this case, because we are interested in gauging the strength of a general monotonic relationship between the two dimensions, not necessarily a linear one. For this data, $\rho=-0.8709$ and the probability of the no-correlation hypothesis is $2.7160 \cdot 10^{-18}$.

Next, we examine mutation probability. We started with the reference BBO parameters from Table 3 and ran eight Monte Carlo simulations for each of the following mutation probability values: 0.01, 0.05, 0.10, and 0.20. The reference BBO runs are also included in these data, for which mutation probability was set to 0.02. We show a scatter plot of these data, including a spline curve fit to the data in Fig. 7. See Thomas (2014) for standard deviation data. Most curves fit to this data will suggest that the minimum is in the neighborhood of mutation probability of 0.05. Given that the rest of the BBO parameters are set to the reference values from Table 3, this data suggests that mutation probability around 0.05 is optimal. This data appears to suggest a non-monotone relationship between minimum cost and mutation probability (one with a global minimum), and so we do not attempt to find a correlation coefficient.

We also examine the convergence of minimum cost in BBO as a function of the number of generations. We ran 10 Monte Carlo simulations for 200 generations each with the parameters from Table 3 held constant. Fig. 8 shows the average of the best costs from each Monte Carlo simulation with standard deviation bars. See Thomas (2014) for standard deviation data.

One way of quantifying the generation where the BBO population converges to a minimum solution on average, is to compare the sample sets of the minimum cost data from the final generation with the data from the previous generations. The first generation (that is, the one with the lowest index) whose cost data come from a distribution statistically similar to the final generation's data can be considered the generation at which BBO converges. Using the t-test with a confidence interval of 0.05, the first generation to be statistically

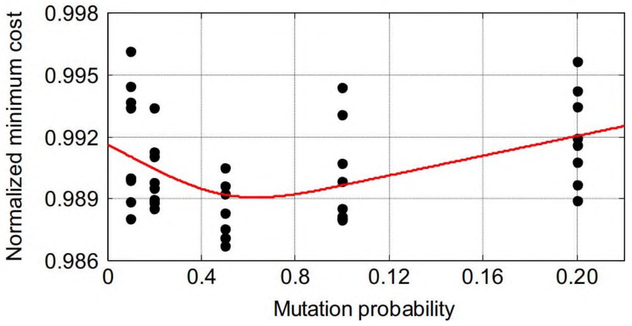

Fig. 7. Scatter plot of normalized minimum cost after 50 BBO generations versus mutation probability. A spline curve fit to this data is also shown.

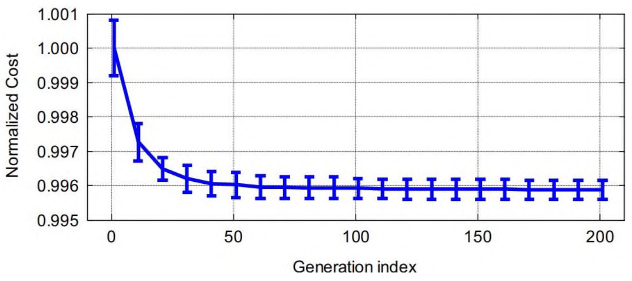

Fig. 8. Normalized minimum cost averaged over the 10 Monte Carlo simulations with vertical bars drawn to indicate minimum cost standard deviation taken over the 10 Monte Carlo simulations.

similar to the final one is generation 40. Fig. 8 corroborates this conclusion, however, the use of different confidence intervals will suggest different generations of convergence. This data tells us that we can run for 40 generations and get minimum cost solutions that are not significantly different from those that we would get from running for 200 generations. This also suggests that our choice of 50 generations for the nominal BBO parameters given in Table 3 is well justified.

We conclude that the most effective BBO parameters for our problem include a population size that is large enough to contain significant information, but small enough so that computational complexity is not unnecessarily increased; specifically, a good tradeoff appears to be around 200. Also, a mutation rate around 0.05 appears to produce the best solutions to our problem. We note that population size has a significantly larger correlation with minimum cost obtained in a given BBO run than mutation rate, so we can consider population size to be the most important BBO variable.

We also note that the conclusions drawn in this section are specific to this particular problem. For instance, problems with smooth objective functions that are smooth and characterized by low frequency spectral content may benefit more from an exploitative search strategy, whereas optimization of problems with highly irregular or multimodal objective functions may benefit from explorative search strategies. The optimal search strategy indicated by these results will not apply to all problems in general. Finally, additional analysis of these data can be found in (Thomas, 2014).

3.2. DACE parameter studies

In order to study the study the effect of DACE on a BBO run, we can run Monte Carlo simulations where we vary the parameters of DACE that define how it is used in the framework of BBO. We choose the method proposed in Section 2.3, in which we use DACE to estimate the cost of a fixed number of candidate solutions each generation. We vary the number of sample points, M, used to fit DACE models each generation in the range [10, 50, 100, and 200], and the percentage of the population that is estimated using DACE in the range [5%, 10%, 20%, and 50%]. We choose simulation parameters for this study based on the BBO parameter study results given in Section 3.1 and the sub-trace results given in Section 2.3 (that is, to use sub-trace 2); these BBO parameters are shown in Table 4.

First, we show surface fits that are quadratic functions of number of DACE sample points and estimated individual percentage. We fit these quadratic surfaces to both the minimum cost and CPU time data of these simulations, and we plot them in Fig. 9 and Fig. 10. Quadratic surface fits were chosen because they appear to fit the data better than any other kind of elementary surface that was evaluated. We also note that several of these simulations were run using the R2011a version of the MATLAB software with up to 8 cost functions evaluated in parallel, and the rest of the simulations were run using MATLAB R2013a with up to 12 cost function evaluations in parallel. All of the simulations were run on the same computer using the same simulation software. We acknowledge that this is a significant source of systematic error, though only in the CPU time data that we have obtained. We have multiplied the CPU time data obtained in the simulations where up to 8 parallel simulations were run by a factor of 8/12 in order to compensate for the discrepancy in parallel computation speed.

The relationships we can infer from these figures are that, as number of DACE samples is increased, the minimum cost slightly increases and simulation time also increases. Also, as the percentage of estimated individuals increases, the minimum cost increases, while the simulation time decreases. Because of this, we can use a small number of DACE samples to reduce simulation time and obtain good BBO solutions, and we can also choose a tradeoff between good solutions and simulation time by picking the percentage of the population that we estimate. In our case, however, it seems that the penalty in best solution fitness incurred by estimating many individuals is minimal, so DACE can be used aggressively to reduce simulation time while still obtaining good BBO solutions. Finally, we note that the positive correlation between minimum cost and number of DACE samples is a counterintuitive one. More DACE samples implies more estimation effort, which one would expect to improve (i.e.; reduce) the best cost obtained with BBO, however, we see the opposite. This result leads us to further examine the error in our DACE models.

We can explain the relationship between number of samples and minimum cost obtained with BBO by viewing the estimation of solutions with DACE as a noisy way of computing cost, and we note that injecting noise into the fitness function may actually improve EA performance, especially on harder problems (Branke and Schmidt, 2003; Qian et al., 2013). Fitness function noise in an EA can have an effect on the evolution of a population similar to the mutation operator, and thus can be beneficial or detrimental in particular cases (Branke and Schmidt, 2003).

We can also analyze the cost and CPU time data using statistics. We can compute correlation coefficients to determine the relationships between both independent and dependent variables in DACE. Table 5 shows the Spearman correlation coefficients between variables, and Table 6 shows the null hypothesis probabilities associated with each pair of variables. We have chosen to use the Spearman coefficient because the relationships between the independent and dependent variables are nonlinear (in fact, the relationships are approximately quadratic) as seen in Figs. 9 and 10.

These tables show a weak positive correlation between number of samples and cost, and a strong positive correlation between the number of estimated individuals and cost. Also, the correlation between number of samples and CPU time is positive, and the correlation between number of estimated individuals and CPU time is negative. This positive correlation between number of samples and cost corroborates our observation from Fig. 9; that is, if we increase the number of samples, cost will go slightly up.

Table 4
Nominal BBO parameters for DACE parameter studies.

Parameter	Value
Generation count	50
Population size	200
Mutation probability	0.05
Problem dimension	10
Number of elite solutions	2
Simulation drive trace	sub-trace 2

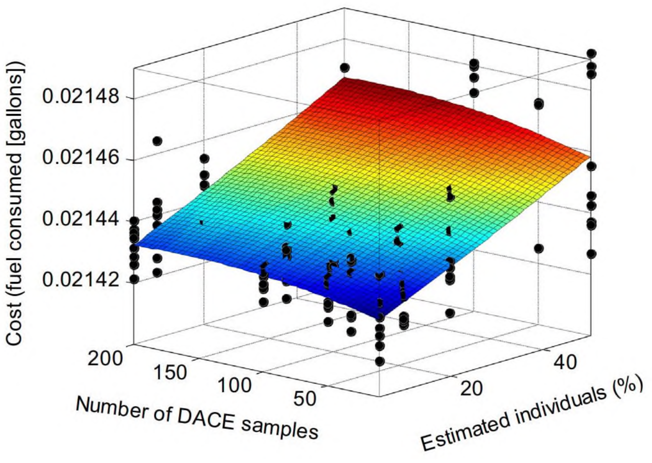

Fig. 9. Scatter plot of minimum cost data as number of DACE samples and percentage of estimated individuals are varied, with a quadratic surface fit showing general trends.

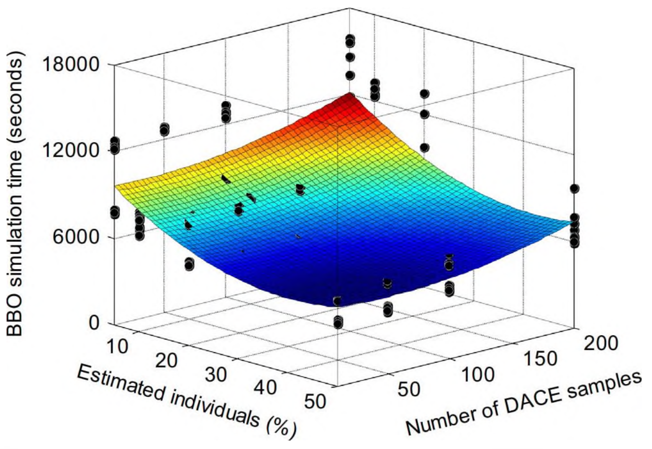

Fig. 10. Scatter plot of simulation time data as number of DACE samples and percentage of estimated individuals are varied, with a quadratic surface fit showing general trends.

Table 5
Spearman correlation coefficients between DACE variables.

Variable	Number of samples	Individuals estimated	Cost	CPU time
Number of samples	1	0	0.2353	0.3035
Individuals estimated	0	1	0.6651	−0.6763
Cost	0.2353	0.6651	1	−0.3938
CPU time	0.3035	−0.6763	−0.3938	1

Table 6
p-values associated with Spearman correlation coefficients between DACE variables.

Variable	Number of samples	Individuals estimated	Cost	CPU time
Number of samples	1	1	0.0045	2.1763E-4
Individuals estimated	1	1	9.7858E-20	1.3834E-20
Cost	0.0045	9.7858E-20	1	1.0470E-6
CPU time	2.1763E-4	1.3834E-20	1.0470E-6	1

Again, this may be explained by considering estimating individuals with DACE as a noisy way of calculating cost, which may improve the minimum cost obtained by an EA (Branke and Schmidt, 2003).

Next, we observe the effect DACE has on the convergence properties of BBO. The BBO populations generally seem to converge within 50 generations, no matter what DACE parameters are used.

We can examine the difference between using DACE aggressively versus using it conservatively by first noting that we can choose 10 DACE samples without a minimum cost penalty (in fact, the results show that using fewer samples actually improves the best solutions we find with BBO.) Next, we can compare the cost and simulation time obtained by running BBO and estimating 5% of the population with DACE, to estimating 50% of the population with DACE. When estimating 5% of the population with 10 DACE samples, the average simulation time is 9641, and the average best cost is 0.02142, whereas, when estimating 50% of the population with 10 samples, the average simulation time and best cost are 5476 seconds, and 0.02146 gallons respectively. When going from 5% to 50% estimated individuals, the percent increase in cost is 0.1867%, and percent decrease in simulation time is 43.20%. These percent changes should be similar when comparing not using DACE at all versus using it aggressively (e.g.; estimating half of the BBO solutions in a run). One's optimization objectives will determine whether this tradeoff is acceptable. One of the future work items we propose is the application of this framework of BBO and DACE to higher fidelity simulation models of the vehicle to further validate the simulation results we get. This simulation may run at 10% of the speed of our current automotive simulation, so instead of saving 3965 seconds by estimating half of the BBO solutions with DACE, we may instead save 39650 seconds. This corresponds to reducing the simulation time of a 27 hour BBO run by more than 11 h.

Finally, we note that these conclusions are only valid for this problem. For example, one would expect to see different results and to draw a different conclusion when applying DACE and BBO to a problem with an objective function that is more difficult to fit with DACE than our problem. In this case, a greater number of samples may be necessary to achieve enough estimation accuracy for good BBO performance.

4. Results of cam timing control optimization with BBO

We explore our BBO simulation results in the cam timing optimization problem domain, so that we can determine what particular improvements we have made and what implications these improvements will have in the engine system. Instead of running additional simulations, we choose to use the optimal BBO solutions from the various Monte Carlo simulations of the previous sections, and we note that the differences between the best solutions from run to run are minimal. Table 7 lists the BBO parameters that were used in the optimization run where we obtained our best solution.

We note that the parameters in Table 7 are not necessarily the best, since BBO is a stochastic optimization algorithm and the best solution from each BBO run will be a function of random variables however, our parameter studies given in the previous sections suggest that we have a better chance of finding good solutions with these parameters than with the others we have tried.

The solution variables (RBF heights) for the best solution that we have found are given in Table 8 – the RBF indices correspond to the RBFs as numbered in Fig. 2.

Notice that many of these RBF heights are close to the limits of the search range (i.e.; [−20, 20].) Generally, if an EA consistently finds solutions close to the search range limits, then the search ranges should be changed, as the optimum that the EA is converging to may be found outside of the range. We have chosen not to adjust our search ranges in this case, because doing so may encourage BBO to generate solutions that have deleteriously large differences between adjacent lookup table values. A significant future work direction is to increase the search ranges, establish limits on the changes between adjacent lookup table values, and implement these as constraints in BBO, which may allow us to find better solutions. We also notice that there are combinations of both large negative and positive values in this BBO solution's variables. If a large positive value is applied to the height of a RBF, and a large negative value is applied to a RBF adjacent to the first, this results in the maximum change between adjacent lookup table values that we can apply. Since we are seeing this behavior in our best solution, this indicates that we are already in danger of producing lookup tables that have abrupt changes.

It is also important to examine the differences between our optimal solution and the reference cam timing lookup tables. Fig. 11 shows visualizations of the lookup tables before and after BBO.

We can observe the difference between adjacent lookup table values by comparing the colors surrounding adjacent black circles in the above plots. The differences between adjacent table elements before and after BBO are fairly similar, so it is unlikely that the differences between adjacent values in the new lookup tables obtained via BBO are excessive and thus likely would not result in ringing or other control problems.

Table 7
BBO parameters used to obtain our best solution.

Parameter	Value
Generation count	300
Population size	100
Mutation probability	0.02
Problem dimension	10
Number of elite solutions	2
Simulation drive trace	LA4
DACE configuration	not used

Table 8
Solution parameters resulting in the lowest cost that we have obtained with BBO.

RBF Index	Intake adjustment	Exhaust adjustment
1	−14.836	−17.340
2	19.691	17.324
3	−19.729	−19.241
4	19.668	−18.753
5	19.856	−19.619

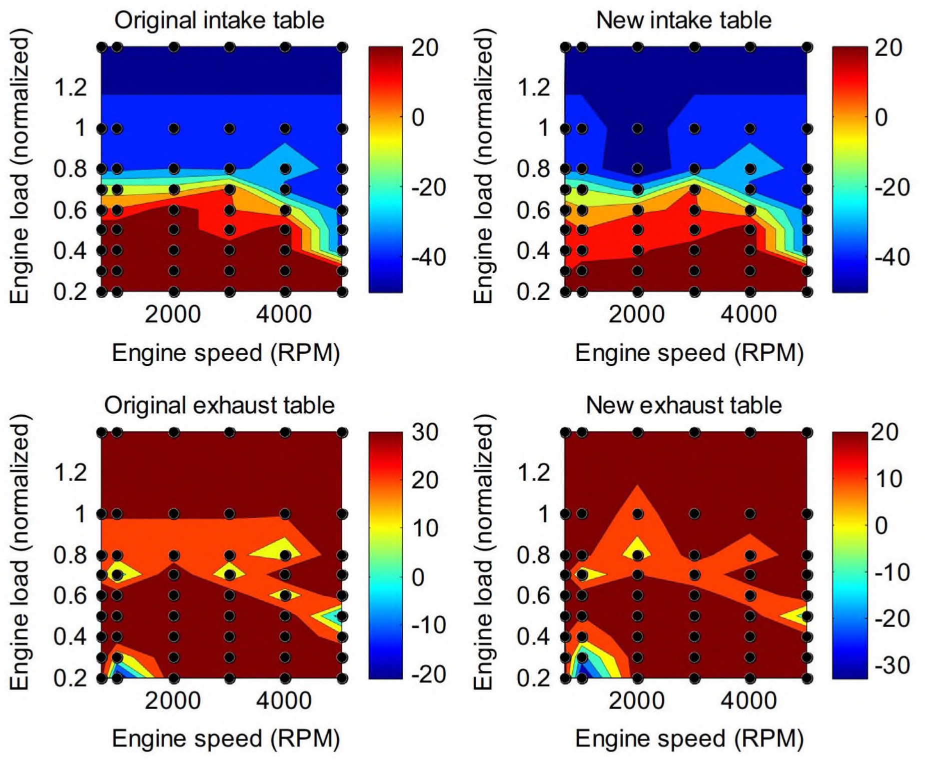

Fig. 11. Filled contour plots of the unchanged reference intake and exhaust cam timing lookup tables (on the right, labeled as Original), and the intake and exhaust tables from the best solution obtained with BBO (on the left, labeled as New.) Locations of lookup table values in (engine load, engine speed) are indicated with black circles.

The cost of this best solution is 0.44152 gallons, and the cost obtained by running the simulation with the nominal, unchanged cam timing lookup tables is 0.44919 gallons. This means a decrease of 1.7%. This is a significant improvement that can be made without any changes to the control algorithm or the hardware.

Explanations for this improvement in fuel economy with the new table may be attributable to EGR effects. Judging from the distribution of engine load and engine speed data in Fig. 2, we may identify three operating regions of the engine: idle, low speed cruising, and high speed cruising. We define idle as engine load of 0.2, and engine speed of 700 RPM. Also, we define low speed cruising as engine load of 0.4 and speed of 1200 RPM, and we define high speed cruising as a load of 0.6 and a speed of 1500 RPM. In Fig. 12, we show the states of the intake and exhaust valves of cylinder 1 as a function of crankshaft angular position for idle, low speed, and high speed cruising, both before and after modifying the tables with BBO.

We can immediately see that the tables produced by BBO have less valve overlap at low vehicle speed, and BBO retards both intake and exhaust valve events significantly, especially at higher vehicle speed. The fuel economy improvement may be attributable to earlier closing of the intake valves, which reduces pumping losses (Sellnau and Rask, 2003). We can also see exhaust port EGR behavior both before and after optimization with BBO, as there is valve overlap after top-dead-center (Meyer, 2007) (which occurs at 360° in Fig. 12). This kind of operation is defined by the intake valve being open for part of the exhaust stroke, during which the high cylinder pressure forces residual gasses into the intake manifold which can also reduce pumping losses. We also note that the reduced valve overlap at idle may be good for reasons besides fuel economy, however, since the EGR behavior that results from valve overlap reduces the amount of combustible material in the cylinder, it can reduce combustion stability. It is important to reduce EGR at idle, since combustion stability is particularly fragile at idle.

We note that the best solutions obtained by BBO are generally similar, and so we can examine the change that BBO makes to the intake and exhaust tables on average, and what parts of these tables vary the greatest by taking the mean and standard deviation of the tables produced by various BBO runs. In Fig. 13, we show the means and standard deviations of the intake and exhaust lookup tables, taken over the best solutions obtained in all of the Monte Carlo simulations given in Section 3.1. We also include contour plots of some exemplary lookup tables from these Monte Carlo simulations in Fig. 14.

First, we note that the differences between tables shown in Fig. 14 occur in regions where the standard deviation is high, as shown in Fig. 13. The fact that the tables can be different in these ways, and still have similarly low costs, suggests that these changes do not affect cost. This makes sense, because changes in these regions often occur in parts of the lookup table domain that the vehicle does not operate in as can be seen by the vehicle (load, speed) trajectory in Fig. 2.

We also note that the changes made to both tables are exclusively negative or zero, this means that BBO is consistently retarding both the intake and exhaust cam timing. Further, for the intake cam, we are most strongly retarding the timing at low to moderate engine load and at low engine speed; this is the state of the engine at idle and when the vehicle starts to move after idle. We can see that the average intake table change is at its highest absolute value and the standard deviation is at its lowest value at idle and low vehicle speed conditions. This indicates that good BBO solutions consistently incorporate this change to the table, and thus, when considering only the range of BBO solutions that we have investigated, this particular change is essential for improving the intake table.

We also see that there is significant retarding of the exhaust cam timing throughout the whole table. We also see that the standard deviation is low throughout most of the table, except at very low engine load and moderate RPM. This vehicle is likely

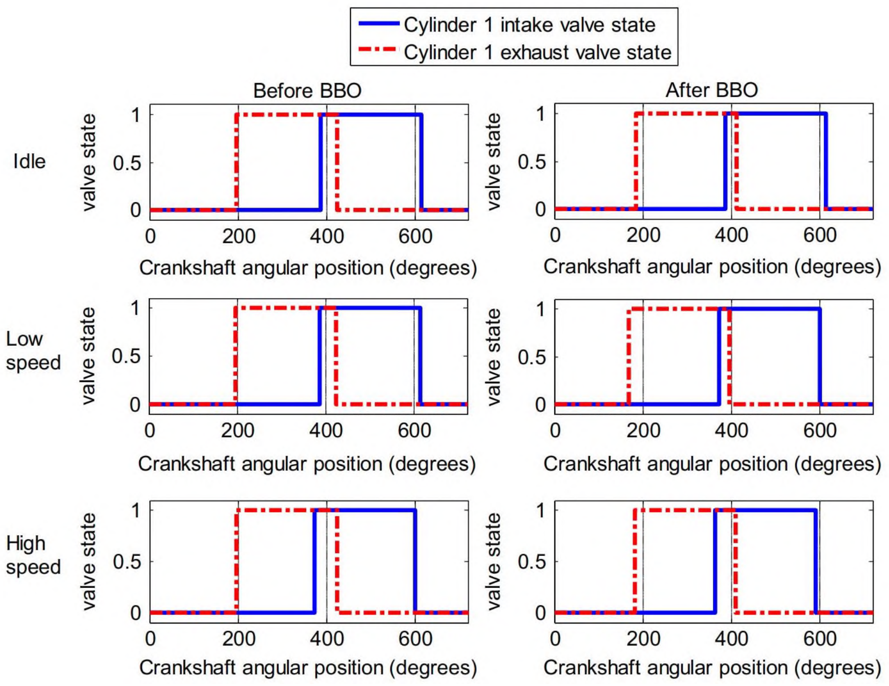

Fig. 12. Cylinder 1 valve events for idling, low speed, and high speed operation as a function of crankshaft angle, illustrating valve overlap. The results using the original lookup tables are shown on the left, and the results produced by BBO are on the right. Note that the range of angles shown is [0, 720], because there are two revolutions of the camshaft for each crankshaft revolution.

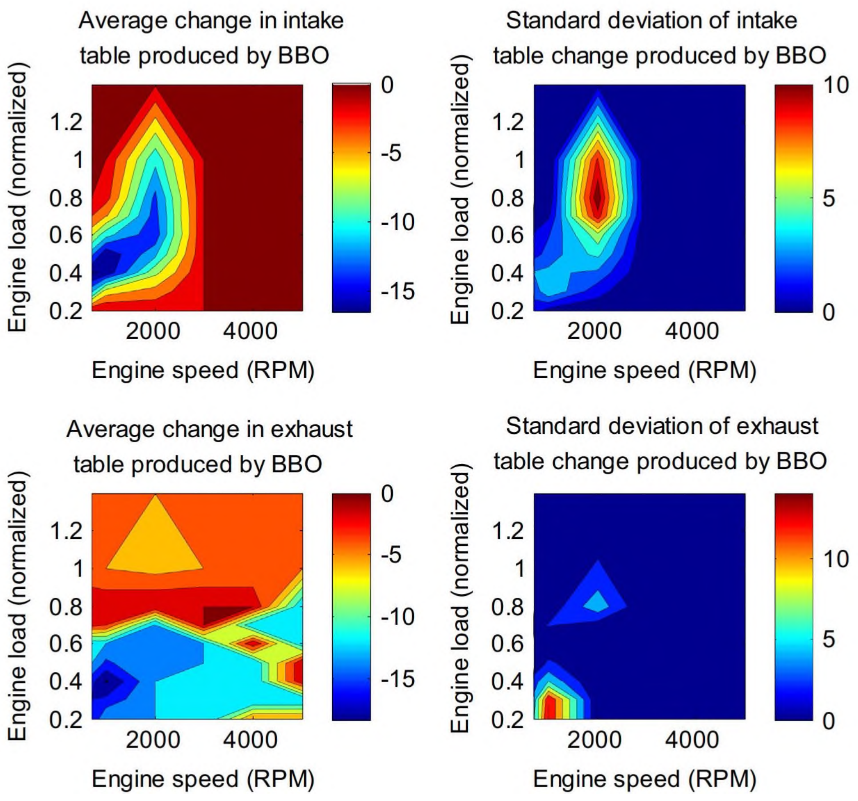

Fig. 13. Average changes in intake and exhaust lookup tables, and standard deviation of changes to those tables produced by BBO.

decelerating when the engine is in this state, and so, since modern vehicles incorporate fuel cutoff during engine braking, it would not matter what state the valves are in during deceleration. Because we observe a significant absolute value of exhaust timing change and low standard deviation throughout most of the table simultaneously, we can conclude that retarding the exhaust timing is also essential. This is likely the case, because in order to achieve the same degree of exhaust port EGR due to valve overlap, while retarding the intake cam to achieve earlier closing of the intake valve, we need to retard the exhaust cam timing as well.

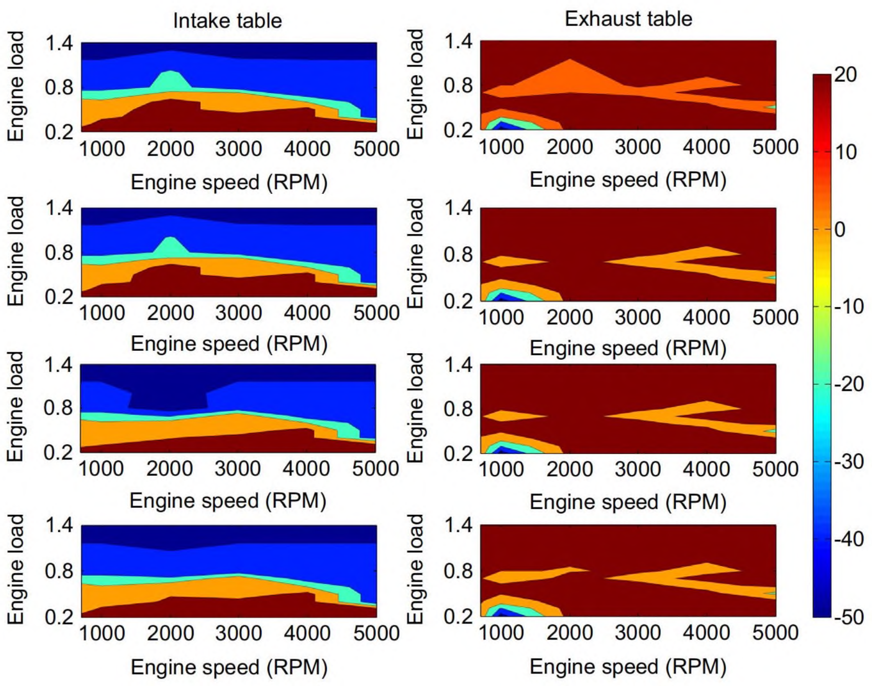

Fig. 14. Examples of lookup tables from the best solutions obtained by running BBO several times. Several exemplary intake tables are shown on the left, and several exhaust tables are shown on the right.

5. Conclusion

We have developed a framework for optimizing the controller set-point lookup tables for a dual independent VCT system using BBO and DACE. This framework includes a medium fidelity simulation of the vehicle for evaluating the fitness of control solutions, which are parameterized using five RBFs. We reduce simulation time by keeping track of the cost of solutions so that we do not need to evaluate their costs more than once in a BBO run, by approximating the full simulation fitness by running over a subset of the simulation drive trace, by adjusting our problem parameterization so that changes to the parameters change the controller lookup tables in the region of the vehicle state space that the vehicle most often traverses, by finding BBO parameters that result in the best optimization results, and by estimating solutions with DACE.

The best search configuration for BBO that we have found involves running BBO with a population size around 200 and mutation probability around 0.05, and using fuel consumption computed over sub-traces with high RMS velocity and acceleration as the fitness function for BBO. We have found other BBO variables and other quantities of the simulation drive traces to have negligible effect on the performance of BBO. We have also developed an application of DACE to BBO that takes the numerical conditioning of DACE models into account. We can improve the numerical properties of our DACE models by using sample sets that are well spread out, which can be found using the sampling heuristic we have developed.

We have found a BBO solution that results in a 1.7% improvement in fuel economy. The vehicle we are simulating is an SUV with an average annual product of 120,000 units, and an average fuel economy of 22 miles per gallon. If we assume that these improved lookup tables are applied to 120,000 vehicles, each of these cars are driven 10,000 miles annually, the average fuel economy of the vehicle without the improvement is 22 miles per gallon, and that our 1.7% improvement in fuel economy can be extrapolated to apply to all of these vehicles, the reduction in fuel consumed by this group of vehicles will be over 900 thousand gallons. Also, since about 9 kg of CO_2 are produced for every gallon of gasoline consumed (U.S. Energy Information Administration, 2014), this results in a reduction in CO_2 emissions of 8,100 metric tons. This means a significant reduction in cost from a variety of sources. Not only will consumers save on fuel, but the cost associated with environmental impact will be reduced as well.

Future work may include running BBO on higher fidelity vehicle simulations to find optimal cam timing lookup tables. It is likely that the locations of good control solutions within the search space will change when changing to a more accurate simulation, and a more accurate simulation will help validate our simulation results. Other future work may include systematic approaches to search space parameterization or BBO parameter optimization. Finally, the BBO implementation we have produced for cam timing optimization may be applied to a variety of other computationally expensive optimization problems. Another direction of future work is to use multi-objective BBO to simultaneously optimize engine control for a fuel economy, emissions, and power, and the control of other automotive subsystems can be optimized with BBO.

References

Bhattacharya, A., 2010. Biogeography-based optimization for different economic load dispatch problems. IEEE Trans. Power Syst. 25 (2), 1064–1077.

Branke, J., Schmidt, C. (2003). Selection in the presence of noise, In: Genetic and Evolutionary Computation – GECCO 2003, Chicago, IL, USA, 2003. pp 766–777.

Congress of the United States: Office of Technology Assessment, 1991. Improving Automobile Fuel Economy. US Government Printing Office, Washington, DC.

De Jong, K., 2007. Parameter setting in EAs: a 30 year perspective. Springer, Berlin Heidelberg, pp. 1–18.

Draper, C.S., Li, Y.T., 1951. Principles of Optimalizing Control Systems and an Application to the Internal Combustion Engine. American Society of Mechanical Engineers, New York, NY.

Jin, Y., Branke, J., 2005. Evolutionary optimization in uncertain environments – a survey. IEEE Trans. Evol. Comput. 9 (3), 303–317.

Jones, D., Schonlau, M., Welch, W., 1998. Efficient global optimization of expensive black-box functions. J. Glob. Optim. 13 (4), 455–492.

Kim, M., Hiroyasu, T., Miki, M. (2005). Multi-Objective optimization of diesel engine emissions and fuel economy using SPEA2+. In: Proceedings of the GECCO '05, 2005 Conference on Genetic and Evolutionary Computation, Washington, DC.

Ma, H., Simon, D., 2011. Analysis of migration models of biogeography-based optimization using markov theory. Eng. Appl. Artif. Intell. 24 (6), 1052–1060.
McKay, M.D., Beckman, R.J., Conover, W.J., 1979. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21 (2), 239–245.
Meyer, J., 2007. Engine Modeling of an Internal Combustion Engine with Twin Independent Cam Phasing, Honors Theses, The Ohio State University, Department of Mechanical Engineering, Columbus, OH.
Pinel, F., Danoy, G., Bouvry, P. (2011). Evolutionary algorithm parameter tuning with sensitivity analysis. In: Proceedings of the 2011 International Conference on Security and Intelligent Information Systems, Warsaw, Poland.
Qian, C., Yu, Y., Zhou, Z. (2013). Analyzing Evolutionary Optimization in Noisy Environments. arXiv preprint: 1311.4987.
Ribbens, W.B., 1984. Electronic engine control in fuel economy. Plenum Press, New York, NY, pp. 419–447.
Sellnau, M., Rask, E. (2003). Two-step variable valve actuation for fuel economy, emissions, and performance, in SAE World Congress, Detroit, MI.
Simon, D., 2011a. A probabilistic analysis of a simplified biogeography-based optimization algorithm. Evol. Comput. 19 (2), 167–188.
Simon, D., 2008. Biogeography-based optimization. IEEE Trans. Evol. Comput. 12 (6), 702–713.
Simon, D., 2013. Evolutionary Optimization Algorithms: Biologically-Inspired and Population-Based Approaches to Computer Intelligence. John Wiley & Sons, Hoboken, NJ.
Simon, D., Rarick, R., Ergezer, M., Du, D., 2011b. Analytical and Numerical Comparisons of Biogeography-Based Optimization and Genetic Algorithms. Inf. Sci. 181 (7), 1224–1248.
Society of Automotive Engineers, 1976. Automotive Fuel Economy. Society of Automotive Engineers, Warrendale, PA.
Thomas, G. (2014, December). Cleveland State University. Department of Electrical and Computer Engineering Masters Thesis, Biogeography-Based Optimization of a Variable Camshaft Timing System. ⟨http://rave.ohiolink.edu/etdc/view?acc_num=csu1419775790⟩.
Thomas, G., Lozovyy, P., Simon, D. (2011). Fuzzy robot controller tuning with biogeography-based optimization. In: Proceedings of the 24th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, Syracuse, NY.
U.S. Energy Information Administration (2014, April). How much carbon dioxide is produced by burning gasoline and diesel fuel? ⟨http://www.eia.gov/tools/faqs/faq.cfm?id=307&t=11⟩.
US EPA (2012, August). Dynamometer Drive Schedules. ⟨http://www.epa.gov/nvfel/testing/dynamometer.htm⟩.
Wang, G., Shan, S., 2007. Review of metamodeling techniques in support of engineering design optimization. ASME Trans. J. Mech. Des. 129 (4), 370–380.
Wilmot, T., et al. (2013). Biogeography-based optimization for hydraulic prosthetic knee control. In: Medical Cyber-Physical Systems Workshop. Philadelphia, PA.
Wu, B., Prucka, R.G., Filipi, Z.S., Kramer, D.M., Ohl, G.L. (2006). Cam-phasing optimization using artificial neural networks as surrogate models–fuel consumption and NOx emissions. In: SAE World Congress. Detroit, MI.
Zhao, J., Min, X., 2013. Fuel economy optimization of an atkinson cycle engine using genetic algorithm. Appl. Energy 105, 335–348.