Optimizing Data Measurements at Test Beds Using Multi-Step Genetic Algorithms

K. KNÖDLER, J. POLAND, A. MITTERER* and A. ZELL
WSI-RA Universität Tübingen
Sand 1, D - 72076 Tübingen
GERMANY
knoedler@informatik.uni-tuebingen.de http://www-ra.informatik.uni-tuebingen.de

Abstract: - Series of measurements should be planned carefully in order to reduce the costs and to allow an efficient execution at measuring devices. For this planning process, we present a multi-step optimization method using genetic algorithms. As a concrete application, we arrange a set of Design of Experiment measuring points appearing during the calibration of combustion engines in an optimal way.

Key-Words: - Genetic Algorithm, Traveling Salesman Problem, Multi-Step Optimization, Adjacency Coding, Variable Alphabet Coding, Monte Carlo Algorithm, Design of Experiments.

1 Introduction

The process of calibrating combustion engines is currently becoming an extremely time-consuming task for car manufacturers. Legal limits concerning exhaust emissions and growing customer requests for economy and performance can only be met by introducing additional engine functionality.

As a consequence, the dimension of an engine's parameter space constantly increases. Therefore it is no longer economical or even possible to use conventional full factorial techniques for calibrating the engine in an optimal way. Statistical Design of Experiments (DOE) reduces the set of measuring points describing the area-of-interest of an engine's parameter space. The results of the measurments are used to tune software engine models, e.g. artificial neural networks or multivariate regression models ([7], [10], [4]). Thereby a faster and thus significantly lower-priced searching for optimal parameters can be performed in an offline process.

Not only the *choice* of the measuring points (DOE) is critical, but also the *arrangement*. The change of the parameters, in particular the engine speed *nmot* and the relative air mass flow *ramf*, results in oscillations of the total engine system. The measurements can only be executed when the system has stabilized again. This needs more relaxation time, the more parameters are changed. Our goal is therefore to minimize the oscillations

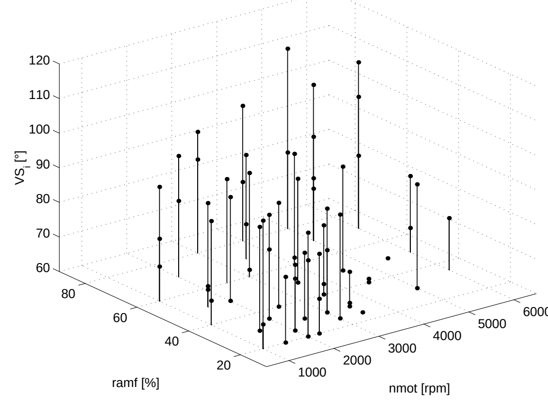

Fig. 1. The N measuring points x_k at the $M < N$ operating points y_j

in order to reduce the total measuring time, by the way improving the robustness and the reproducibility of the measurements. Another criterion is the fact that test bed engineers tend to prefer changing only one parameter at a time.

At our starting point, there already exists a set of N DOE measuring points at M unique operating points $\{y_j\}_{j=1}^{M} = \{(nmot_j, ramf_j)\}_{j=1}^{M}$. The operating range is spanned by the engine speed *nmot* and the relative air mass flow *ramf* (with respect to the maximum value). There are n_j valve spread combinations $\{x_k^{(j)}\}_{k=1}^{n_j} = \{(VS_i^{j,k}, VS_e^{j,k})\}_{k=1}^{n_j}$ at each operating point, where VS_i defines the inlet and VS_e the exhaust valve spread. Figure 1 visualizes this situation using the VS_i components of these measuring points.

*A. Mitterer, BMW Group, D-80788 München, Germany. E-mail: alexander.mitterer@bmw.de.

2 Separation of the problem

Mostly, a change of the engine speed *nmot* results in a particularly oscillating engine system. Changing the relative air mass flow *ramf* is slightly less critical. For some engines the converse may be true. Changing the valve spreads yields less instability. This fact strongly suggests a *separation* of the sorting problem. In a first step we consider the operating points only, the second step concentrates on the valve spread components of the measuring points. These two subproblems are each similar to the Traveling Salesman Problem (TSP): in the first part we search for a shortest path between the M operating points y_j resulting in a certain order of the $\{x_k^{(j)}\}_{k=1}^{n_j}$ blocks. The second step requires the sorting of the valve spread components within each block. In order to satisfy the constraints, each TSP-like subproblem is solved by its own algorithm using different measures for calculating the minimum path length.

3 Subproblem 1: The operating range

First we search for the shortest path between the M unique operating points y_j. There are two differences to the original TSP: first, we look for the shortest path instead of cycle. As starting point, we use the operating point defined by the minimum *nmot* and *ramf* values. Second, the amount of *simultaneous* changes in the *nmot* and the *ramf* direction has to be minimized in order to keep the relaxation times as short as possible. The solutions on the operating range are therefore either formed as slalom courses in the *nmot* or in the *ramf* direction.

There is a well known heuristic for the Traveling Salesman Problem, consisting of

- random edge exchange and
- random single point insertion.

These operations are performed a number of times (in our algorithm $1000 \cdot M$ times). This heuristic is known to produce good solutions in general and even optimal solutions in the case of a small number of points. Since we are interested in the overall optimal path, while computation time is rather negligible, we carry out the heuristic 10000 times and choose the best result. Figure 2 shows the percentage of runs finding the optimum over the number of operating points. If M grows too large ($M \geq 99$), the pure heuristic almost never finds the optimum ($< 0.01\%$) and is thus no longer acceptable.

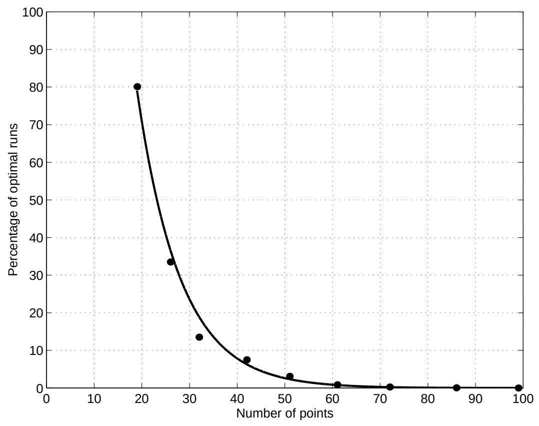

Fig. 2. Percentage of heuristic runs finding the optimum over the number of operating points

Hence, we expand the heuristic with a genetic algorithm. There are several possible representations for a path, we follow the suggestion of Grefenstette et al. ([5], see also [6], [2], [3], [8] for this topic). A tour is described as a list of numbers. There is an edge from operating point y_i to y_j, if the component of the list at position i takes the value j. A simple example: the path tour $1 \to 4 \to 2 \to 5 \to 3$ corresponds to the adjacency tour $(4, 5, -, 2, 3)$.

For the fitness function, we define a path length according to the above considerations:

$$\phi(p) \; = \; \sum_{j=1}^{M-1} d(y_{p(j)}, y_{p(j+1)}),$$

where p is a permutation of $\{1, \ldots, M\}$ representing a path. The distance between two operating points y and $\tilde{y}$ is defined by their euclidean distance plus their respective *nmot* and *ramf* distances with appropriate weights:

$$\begin{aligned} d(y, \tilde{y}) = \|y - \tilde{y}\|_2 + \\ g_{nmot} \cdot |y_{nmot} - \tilde{y}_{nmot}| + \\ g_{ramf} \cdot |y_{ramf} - \tilde{y}_{ramf}|. \end{aligned}$$

The weights g_{nmot} and g_{ramf} should be chosen in such a way that

- paths run mainly parallel to the coordinate axis and
- the parameter which is more critical with respect to oscillations is changed less frequently.

We use the genetic algorithm described in [9], which is an improved and extended version of the

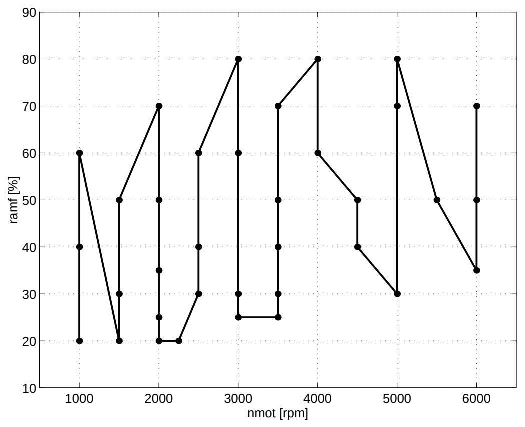

Fig. 3. Minimizing operating points path length: The shortest slalom course in the *nmot* direction

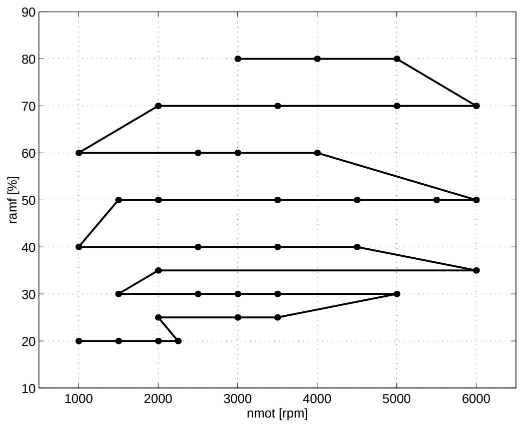

Fig. 4. Minimizing operating points path length: The shortest slalom course in the *ramf* direction

basic genetic algorithm (see e.g. [1]). The evolutionary operations are:

Crossover: Performing n-point or uniform crossover on individuals coded in the adjacency representation often leads to infeasible offsprings. The repairing algorithm takes into account the orders of the points given by both parents. For a more detailed description see [2] and [6].

Mutation: As mutation operator, the heuristic described above is used.

3.1 Results for minimizing the operating points path length

The empirical results presented here rely on a real, relatively small data set consisting of $N = 101$ measuring points at $M = 35$ unique operating points (see Figure 1). In this case, the pure heuristic works well. In the general case with a genetic algorithm, we mention that the heuristical mutation operation needs to be performed with the same probabilty as the crossover operation in order to meet the optimum.

Figure 3 and 4 respectively show the results of the genetic algorithm using different settings for the weight factors g_{nmot} and g_{ramf} in the d-distance. This leads to either the shortest slalom course in the *nmot* or in the *ramf* direction.

4 Subproblem 2: Sorting the valve spread blocks

Having ordered the M operating points, we now have to sort the n_j valve spread components within each of the M blocks. In our case, this is done by searching the shortest path between all N measuring points projected to the (VS_i, VS_e) plane under conservation of the order determined in the first step. Taking the euclidean distance between the valve spread components leads to good results. This is due to the fact that changing the camshaft parameters simultaneously results in quite short lasting oscillations compared to the ones corresponding to changes on the operating range.

We use two simple deterministic methods to sort the blocks of valve spread components at the M operating points. These methods are demonstrated in the following. Consider a situation where we have M blocks with $\{n_j\}_{j=1}^{M} = \{3, 4, 2, ..., n_M\}$ measuring points:

1	2	3	4	5	6	7	8	9	10	...

First we calculate for each block j the order of its valve spread components $\{x_k^{(j)}\}_{k=1}^{n_j}$, which leads to the shortest path inside the block. This is done by considering every possible permutation.

4.1 Method 1: Sequential orientation switching

This method uses the latter result and determines sequentially the best orientation of the shortest path inside each block j. The order is reverted, if this leads to a shorter euclidean distance to the last point of the previous block $j-1$. Table 1 demonstrates that the last point of block j is used to decide if switching the orientation is useful or not. A drawback of this method is that it processes the list of the blocks successively, starting with the second block. The choice of the orientations at previous blocks already limits the selection within the following blocks. For the first block, both orientations are taken into account.

1	2	3	4	5	6	7	8	9	10	...
		+	+	+	+	+				...
						-	-	-		...
								+	+	...
1	3	2	7	4	6	5	9	8	10	...

Tab. 1. Sequential orientation switching

4.2 Method 2: Correlated sorting

This method uses the valve spread components $\{x_k^{(j)}\}_{k=1}^{n_j}$ plus the last one of the already sorted last block $j-1$ for each block j. Now, all possibilities of ordering with $x_{n_{j-1}}^{(j-1)}$ as first point are considered in order to find the path with minimum distance. The fields in table 2 signed by $\oplus$ and $\ominus$ demonstrate that n_j+1 points are used to find the best order within block j. Again the first block has to be treated separately by considering both orientations of its shortest path.

1	2	3	4	5	6	7	8	9	10	...
		$\oplus$	+	+	+	+				...
						$\ominus$	-	-		...
								$\oplus$	+	...
1	3	2	7	4	6	5	9	8	10	...

Tab. 2. Correlated sorting

It is obvious that method 1 can be easily extended in order to find the best orientation for each optimized path at all operating points. The choice of orientation at any operating point depends on the orientations at every other operating point. We therefore use first a simple Monte Carlo Algorithm and second a genetic algorithm to master this job.

4.3 Method 3: Monte Carlo Algorithm

The Monte Carlo Algorithm inverts the orientation of the optimized path of a randomly chosen operating point j. This operation is repeated several times. The new orientation survives if the resulting total path between all valve spread components is reduced.

4.4 Method 4: Genetic algorithm

We solve the problem of finding the orientations of the shortest path at each block j by using a genetic algorithm. As will be explained, this method is a special case of the method discribed in the next section. We use the euclidean distance as fitness function for the individuals. The individuals are bit strings of length M with 0 or 1 at each bit position for the two orientation possiblities.

4.5 Method 5: Pure genetic sorting

We now generalize the methods 1 and 4 of the privious sections. At each operating point y_j there are n_j valve spread components. That means that there are $n_j!$ possible permutations, which can be used to describe the order of the measuring process. Remember that method 4 only used two permutations, both representing the shortest path between the corresponding valve spread measuring points, but with different orientations. This section shows, how variable alphabet coding suggested in [9] can be used to generate individuals, which will again be treated by a genetic algorithm.

Now, an individual is just one choice of the $n_j!$ permutations of the valve spread components at each block j. In order to obtain a uniform distribution, we use variable alphabet coding described in [9]: each block j corresponds to one position of the bit string and is allowed to take as many different values as there are permutations of the existing valve spread measuring points. That means position j can take $P_j = n_j!$ values. Hence, we use a different alphabet for each position of the individual. Each individual v has the form

$$v \;=\; (v_j)_{j=1}^{M} \;\in\; \bigotimes_{j=1}^{M}\{1\ldots P_j\}.$$

For our sorting problem, other representations like unary coding, binary coding with bias and binary coding with penalty, examined for comparison purposes in [9], perform worse. The reason is on one side the non-uniform distribution of the individuals for unary coding and on the other side the extremely high number of invalid individuals for binary coding.

Again the euclidean distance between the valve spread components is used as fitness function for the individuals. The initial population is generated randomly using one possible permutation at each block j. This time the genetic algorithm described in [9] together with the standard crossover and mutation operations, i.e. either uniform or n-point crossover and one-bit or random mutation with p_{mut} = (length of the individual)$^{-1}$ are applied.

4.6 Results for minimizing the valve spreads path length

In the following we present and discuss the results of the described different algorithms. We start with the results of the simple sorting meth-

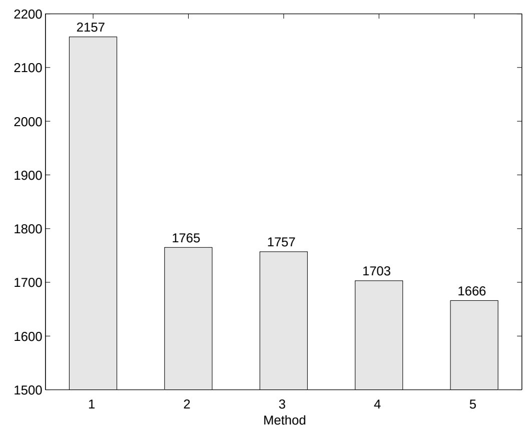

Fig. 5. Sorting valve spread blocks: Valve spreads path lengths for *nmot* priority

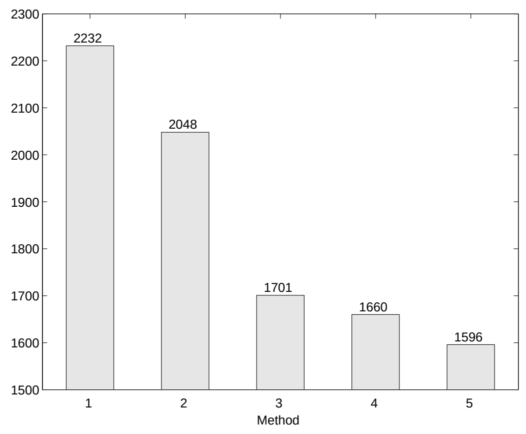

Fig. 6. Sorting valve spread blocks: Valve spreads path lengths for *ramf* priority

ods. Figures 5 and 6 show the shortest euclidean distances between all N valve spread components projected to the plane spanned by the valve spreads. For the different methods both the $nmot$ case and the $ramf$ case are displayed.

The methods 3 und 4 lead to better results compared to the straightforward methods 1 and 2. There are two reasons for that: first, the methods 3 and 4 pick randomly chosen blocks for determining the best orientations of the shortest paths; second, these methods are not limited to sequential optimization, starting with block 2.

We performed 100000 independent experiments with 1000 orientation switches per run to obtain the results of method 3. The orientations leading to the shortest total distance are found using method 4, the genetic extension of method 1. We are able to find the orientations of the optimized paths which lead to the shortest total path under the restriction of using the shortest path inside each block.

With the pure genetic sorting defined by method 5 using variable alphabet coding, we are able to achieve a further improvement of method 4. The reason is that the pure genetic method is not restricted to the shortest paths at each block. We are able to find the overall shortest path length between all valve spread components, i.e. 1666 in the *nmot* case and 1596 in the *ramf* case. Figures 5 and 6 show these values, too. In the case of *nmot* priority, at about 6/7 of all blocks the arrangement of the shortest paths determined by method 4 were used. For the case of *ramf* priority this fraction is at least about 3/5. In both cases, the parameter setting of the genetic algorithm is: $\mu = 100$ parent individuals, $\lambda = 100$ offsprings, tournament selection with $q = 4$ individuals, 3-point crossover with $p_{cross} = 0.6$, bitwise (conventional) mutation with $p_{mut} = 1/35$ and a niche factor of $\alpha = 0.5$.

5 Conclusions

We presented a multi-step optimization process for sorting a set of measuring points using genetic algorithms, Monte Carlo Algorithm, and simple deterministic sorting methods. With these algorithms, an improved measuring arrangement and hence reduced relaxation times and overall measuring time is achieved. This arrangement was previously handwork. In the second step the pure genetic approach has shown to be superior to the simple sorting methods and the Monte Carlo Algorithm.

Several modifications and extensions in the methods are imaginable. If the block size for the second step increases, the coding of the permutations is no more efficient for the genetic algorithm. In this case, one could apply a blockwise adjacency coding similar to the coding we used for step one. In another situation, a one-step algorithm may yield better results, namely if the weights for the parameters are almost equal. This leads to more complex paths and to an increased running time of the algorithm.

Acknowledgments

We thank Thomas Fleischhauer and Frank Zuber-Goos at BMW Group Munich for helpful discussions. This research has been supported by the BMBF (grant no. 01 IB 805 A/1).

References:

[1] T. Bäck. *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, 1996.

[2] T. N. Bui and B. R. Moon. A new genetic approach for the traveling salesman problem. *International Conference on Evolutionary Computation*, pages 7–12, 1994.

[3] T. N. Bui and B. R. Moon. On multi-dimensional encoding/crossover. *6th International Conference on Genetic Algorithms*, pages 48–56, 1995.

[4] T. Fleischhauer, A. Mitterer, K. Knödler, J. Poland, and A. Zell. Motoroptimierung mit Hilfe neuronaler Netze und evolutionärer Algorithmen. Technical report, Universität Tübingen, WSI RA, Feb 2000. Schlussbericht des Vorprojekts.

[5] J. Grefenstette, R. Gopal, B. Rosmaita, and D. Van Gucht. Genetic algorithms for the travelling salesman problem. *Proceedings of the first International Conference on Genetic Algorithms and Application*, pages 160–168, 1985.

[6] A. Homaifar, S. Guan, and G. E. Liepins. A new approach on the traveling salesman problem by genetic algorithms. *5th International Conference on Genetic Algorithms*, pages 460–466, 1993.

[7] A. Mitterer. *Optimierung vielparametriger Systeme in der Antriebsentwicklung, Statistische Versuchsplanung und Künstliche Neuronale Netze in der Steuergeräteauslegung zur Motorabstimmung.* PhD thesis, Lehrstuhl für Meßsystem- und Sensortechnik, TU München, 2000.

[8] B. R. Moon and C. K. Kim. A two-dimensional embedding of graphs for genetic algorithms. *7th International Conference on Genetic Algorithms*, pages 204–211, 1997.

[9] J. Poland, K. Knödler, Holger Böttle, A. Mitterer, T. Fleischhauer, F. Zuber-Goos, and A. Zell. Finding smooth maps in motor conrol unit calibration using genetic algorithm with variable alphabet coding. *Preprint 2000.*

[10] K. Weicker, A. Mitterer, T. Fleischhauer, F. Zuber-Goos, and A. Zell. Einsatz von Softcomputing-Techniken zur Kennfeldoptimierung elektronischer Motorsteuergeräte. *at-Automatisierungstechnik*, 48, 2000.