Evolutionary Search for Smooth Maps in Motor Control Unit Calibration

Jan Poland, Kosmas Knödler, Alexander Mitterer*, Thomas Fleischhauer*, Frank Zuber-Goos* and Andreas Zell

Universität Tübingen, WSI-RA, Sand 1, D - 72076 Tübingen, Germany
poland@informatik.uni-tuebingen.de,
http://www-ra.informatik.uni-tuebingen.de

Abstract. We study the combinatorial optimization task of choosing the smoothest map from a given family of maps, which is motivated from motor control unit calibration. The problem is of a particular interest because of its characteristics: it is NP-hard, it has a direct and important industrial application, it is easy-to-state and it shares some properties of the wellknown Ising spin glass model. Moreover, it is appropriate for the application of randomized algorithms: for local search heuristics because of its strong 2-dimensional local structure, and for Genetic Algorithms since there is a very natural and direct encoding which results in a variable alphabet. We present the problem from two points of view, an abstract view with a very simple definition of smoothness and the real-world application. We run local search, Genetic and Memetic Algorithms. We compare the direct encoding with unary and binary codings, and we try a 2-dimensional encoding. For a simple smoothness criterion, the Memetic Algorithm clearly performs best. However, if the smoothness citerion is more complex, the local search needs many function evaluations. Therefore we prefer the pure Genetic Algorithm for the application.

1 Introduction

This paper deals with a combinatorial optimization problem that is motivated from the calibration of electronic control units (ECUs) for combustion engines. We briefly sketch the situation. During engine operation, each engine parameter[1] is being controlled by a map that is stored in the ECU as a lookup table. In this way, the engine is adjusted to the actual operating situation, which is necessary to obtain optimal fuel consumption, exhaust emissions, etc. Technically, an operating situation is a combination of engine speed and relative air mass flow and thus corresponds to a point in a rectangular domain, the *operating range*. Moreover, the maps in the ECU are defined by a grid $(x_j)_{j=1}^n$ on the operating range, the *operating points*, and the respective co-domain values. If there are m engine

* BMW Group, 80788 München, Germany.

[1] engine parameters are controlled variables that determine the behaviour of the engine, e.g. ignition timing angle

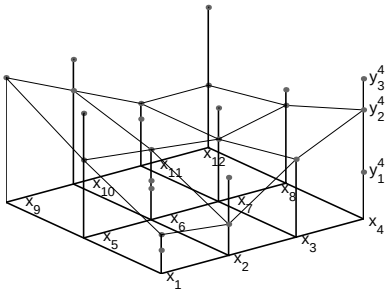

Fig. 1. The context: grid, candidates, and a sample map

parameters, then the control unit must store m maps $\left((y^{(j)(\mu)})_{j=1}^{n}\right)_{\mu=1}^{m}$. For operating points that are not grid points, the ECU will interpolate the parameter values. (See e.g. [1] for the technical background).

Our problem now arises in the final phase of the lookup table design. For each operating point x_j, we are given a set of candidates $\{(y_k^{(j)(\mu)})_{\mu=1}^{m}\}_{k=1}^{n_j}$, each of which defines a combination of values for the engine parameters. These candidates have been obtained by a previous optimization process separately for each operating point, and they have approximately the same quality (fuel-consumption etc.). In order to populate the lookup table, one candidate has to be selected at each operating point. The constraint is the fact that some of the parameters are adjusted *mechanically* and thus have a certain inertia. Hence, one is interested in *smooth* maps for these parameters, in order that the value can be adjusted swiftly when the operating point changes.

Note that this is a *multiple objective optimization* task unless $m = 1$, since we have m maps to smooth and choosing a candidate means fixing *all* parameter values at the corresponding operating point. Thus, the problem is not separable, i.e. it is not possible to optimize each map separately.

Clearly the meaning of "smoothness" has to be defined precisely. The first part of this paper will assume a very simple smoothness criterion. This turns the problem into an easy-to-state combinatorial optimization problem with several interesting properties. The second part of this paper deals with the application.

We use local search heuristics and Genetic Algorithms to find smooth maps. Even with a simple smoothness definition, it can be seen that a pure heuristic is not powerful enough to perform an efficient global optimization. On the other hand, the application implies more complex smoothness criteria. Here, the local search becomes less efficient, and other methods that require deeper insight into the problem structure are very expensive to develop or even intractable. However, we will see that a Genetic Algorithm can still produce good solutions.

2 The abstract case

In this section, we will consider an abstract version of our problem and a very simple smoothness definition.

Let $G = (x_j)_{j=1}^n$ be a set of points defined by a rectangular P by Q grid in $\mathbb{R}^2$. For each grid point x_j, a number of candidates $\{y_k^{(j)}\}_{k=1}^{n_j} \subset \mathbb{R}$ is given (see Fig. 1). Then, $M(x_j) = y_{k_j}^{(j)}$ defines a map from G into $\mathbb{R}$, this is the map obtained by choosing the candidate $y_{k_j}^{(j)}$ at the grid point x_j with $1 \leq k_j \leq n_j$ (Fig. 1 shows an example map). Clearly, there are $\prod_{j=1}^n n_j$ different maps, we denote the family of all these maps by $\mathcal{M}$.

This situation is quite similar to the wellknown spin glass model in two dimensions. Therefore, we will use a similar terminology and refer to the *energy* of a map instead of its smoothness or steepness. Then, the problem of minimizing the energy of a map corresponds to the problem of finding an exact ground state of a spin glass model (see e.g. [2]). Of course, there are also relations to other fields and problems with a two-dimensional structure, e.g. image processing.

Definition 1 *For two neighbouring grid points $x_i, x_j \in G$ and a map M, we define the energy of the connection $x_i - x_j$ in the map as*

$$E_{(i,j)}(M) = \mathbf{1}_{|y^{(i)} - y^{(j)}| > c},$$

i.e. the connection energy is 1 if the ordinate distance is greater than a constant threshold $c > 0$, and 0 otherwise. We may fix $c = \frac{1}{2}$ without loss of generality.

Two grid points are neighbouring if they are adjacent in the grid.

Definition 2 *For a map M, we define its energy (i.e. steepness) by the sum over all connection energies:*

$$E(M) = \sum_{i<j \text{ are neighbours}} E_{(i,j)}(M).$$

The above definition has a natural interpretation in terms of inert engine parameters: If the distance of two neighbouring grid points x_i and x_j is too large, then the adjusting time for the engine parameter is larger than a certain limit when the operating point changes from x_i to x_j. The limit could be derived e.g from the time resolution of the control unit.

Since the smoothest map has the lowest energy, our problem can be stated as follows.

Problem ("SmoothMap"): Find the map $M_0 \in \mathcal{M}$ with minimal energy, i.e.

$$E(M_0) = \min_{M_0 \in \mathcal{M}} E(M).$$

Although the above energy definition is very simple, and we have restricted to only one optimization objective ($m = 1$), the resulting abstract problem SmoothMap has very interesting features:

1. It is NP-complete in the grid dimension, even if the number of candidates n_j is limited to 2 for all $1 \leq j \leq n$. This is shown in [3].
2. It admits a PTAS (polynomial time approximation scheme) for certain "non-degenerate" instances. (Instances with a very small energy of the optimal solution are called degenerate, since in this case the approximation quality of the PTAS cannot be assured.) However, the PTAS does not produce good solutions in reasonable time, on the other hand, there is no fully PTAS .
3. If one grid dimension is equal to 1 (the *one-dimensional* case), there is a polynomial time algorithm. This is extendible to the case where one grid dimension is bounded. See [3] for these results.
4. Because of the simple neighbourhood definition, the problem has a strong 2-dimensional local structure.
5. A very appropriate encoding for Genetic Algorithms is the direct encoding, which leads to a variable alphabet. This will be discussed in the sequel.

For the spin glass model, certain cases have been proved to be NP-hard, while other cases are solvable in polynomial time (see [4]), including the simple two-dimensional case. Although the present problem seems to be very similar, its structure is quite different, and so is the proof of its complexity.

2.1 Variable Alphabet Coding

The direct encoding of a map is just a vector with n components $c = (c_j)_{j=1}^n$, each component c_j defining the candidate chosen at the grid point x_j. In terms of Genetic Algorithms, each x_j corresponds to one gene c_j of the chromosome c, and we have

$$c \;=\; (c_j)_{j=1}^n \;\in\; \bigotimes_{j=1}^{n} \{1, \ldots, n_j\}.$$

If this variable alphabet coding is used for Genetic Algorithms, we observe that the conditions and hence the assertions of the schema theorem are not at all affected (cf. [5], [6]).

Since the genes correspond to grid points, a simple 2-dimensional arrangement of the genes resulting in a 2-dimensional chromosome would be natural. Appropriate crossover operators have been suggested e.g. in [7]. On the other hand, a conventional one-dimensional chromosome can be considered as a reference.

In order to show the benefits of the direct encoding, we will additionally try a standard binary encoding as well as a unary encoding. (Here, "unary encoding" means bit-counting, i.e. a sufficiently large part of the bit string is reserved for a grid point and the number of ones in this part define the candidate chosen.)

2.2 A Local Search Operator

Because of its 2-dimensional local structure, the problem admits a very canonical and powerful local search heuristic.

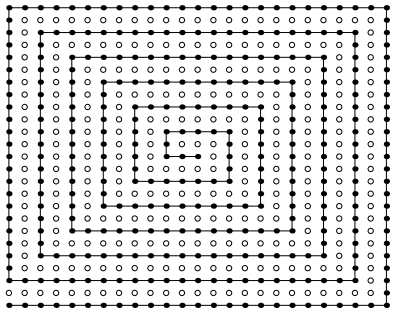

Fig. 2. Construction of a spiral line test instance

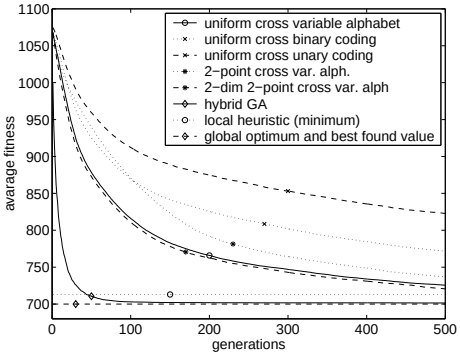

Fig. 3. Fitness curves for the spiral line test instance

Algorithm.
Repeat N times
  Choose $j \in \{1, \ldots, n\}$ randomly
  Find all $k \in \{1, \ldots, n_j\}$ such that the resulting choice has minimal energy
  Choose randomly one of theese k
End Repeat

In order to obtain reference solutions, a good choice for number of loops is $N = 10 \cdot P \cdot Q$, where P and Q are the grid dimensions. On the other hand, we will use the heuristic as mutation operator for the GA thus obtaining a Memetic Algorithm (hybrid GA). In this case, $N = P \cdot Q$ is sufficient.

2.3 Test Instances

We will use three types of test instances of the abstract problem SmoothMap.

1. Random test instances.
   For each grid point j, both the number of candidates and the candidate values are randomly generated. This is the easiest test instance type, the test instances are almost surely nondegenerate. On the other hand, the global optimum is unknown and intractable to compute, so one cannot check if an algorithm succeeds to find it.

2. Line test instances.
   This test instance type permits to compute the global optimum. One starts by placing a line onto the grid, such that the line does not touch itself, i.e. neighbouring grid points that belong to the line are neighbouring in the line. In Fig. 2, the line is spiral-formed, other forms are possible as well, e.g. meander. For the line points, the number of candidates and the candidate values are randomly generated. Then the optimal choice for these points is

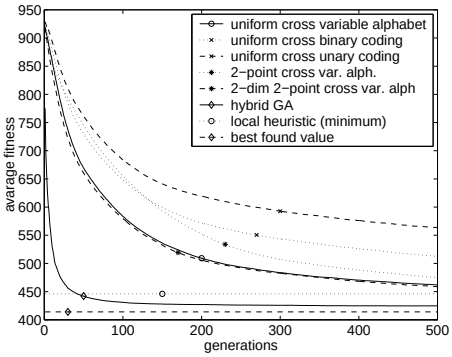

Fig. 4. Fitness curves for the random test instance

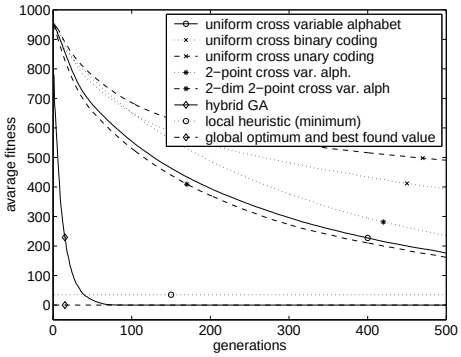

Fig. 5. Fitness curves for the noisy plane test instance

calculated with the efficient one-dimensional case algorithm (see [3]). Finally the remaining grid points (the "separating" points) are randomly endowed with candidates, where clearly one has to assure that the optimal choice for the line points does not change. The resulting test instance has a global optimum that is easy to compute only if the line is exploited. Moreover, it is almost surely nondegenerate.

3. Noisy plane test instances.
   A very simple way to produce test instances with known global optimum 0 is the following: Start with one candidate for each grid point and arrange these candidates such that the respective neighbouring differences are small, i.e. they form a noisy plane with energy 0. Then add more randomly placed candidates for each grid point, clearly the optimum remains 0. Placing these additional candidates far away from the noisy plane probably raises the optimization difficulty for the local heuristic. The resulting test instances are of course degenerate, and the optimum is easy to find if the existence of the noisy plane is known.

2.4 Experimental Results

The experiments have been performed with a spiral line instance, a random instance and a noisy plane instance. All three of them are defined on a 25×25 grid and contain from 3 to 7 candidates per grid point. These relatively large numbers were chosen in order to obtain difficult test instances and thus a good observation of the convergence speed. The following representations and crossover operators are compared:

1. direct encoding with uniform crossover,
2. binary coding with uniform crossover,
3. unary coding ("bit counting") with uniform crossover,

4. direct encoding with 2-point crossover,
5. 2-dimensional direct encoding with 2-point crossover.

In order to obtain an undistorted measure of the convergence properties of the different representations, we use the pure GA. Additionally, a Memetic Algorithm and the pure local heuristic have been executed.

All code has been implemented in MATLAB. The following GA parameters were used: five parallel populations, each of size $\mu = 200$ (one population of size $\mu = 100$ for the hybrid GA), number of generations $t_{max} = 500$, tournament selection with $q = 5$, crossover and mutation probability $p_{cross} = 0.6$ and $p_{mut} = 1/625$ (for the hybrid GA the local search is performed with probability $p_{mut} = 0.01$). For each setting, 30 runs have been executed. The fitness curves in the plots (Figs. 3 - 5) display the average fitness values over the generations, except for the pure heuristic runs, here the minimum value is shown for reference purpose.

The Memetic Algorithm converges much faster than any of the pure GAs for all test data sets, even with less function evaluations per generation. This was not unexpected because of the strong local structure of the problem. However the pure GA reaches the solution quality of the Memetic Algorithm after a sufficient number of generations, since the combination of conventional mutation and selection can simulate the local heuristic.

The direct encoding performs considerably better than both the binary and the unary coding. This is not only valid for the avarage values: The *best* binary or unary results are even worse than the *worst* variable alphabet results after generation 35 for all three test data sets. The 2-point crossover reduces the convergence speed for the 1-dimensional encoding, while in combination with the 2-dimensional natural encoding the performance improves a little.

The average running time (with a Pentium III, 500 Mhz) is about 24 min for the variable alphabet GA, 89 min for the binary and 57 min for the unary coded GA. This is mainly due to the fact that the fitness function is relatively cheap, while the decoding of the binary or unary bit strings takes some time.

The plots for the first two data sets look very similar. This indicates that the line test instance is good in the sense that its hardness is similar to the random instance, if the line information is not exploited. However, the global optimum 700 was found 7 times by the Memetic Algorithm, while for the random instance the best ever found value 414 was obtained only once. This suggests that the random instance is still harder to optimize. The last test instance is apparently easier for the local heuristic. However, even for this data set, the pure heuristic fails to find the global optimum in any run.

3 The application

After having investigated the abstract problem, we turn to the application. That is, we are given a set of *operating points* and *candidates* and try to find a candidate choice that results in m *simultaneously* smooth maps.

As already mentioned, the operating range is spanned by the engine speed and the relative air mass flow. We will consider $m = 2$ maps here, one for the

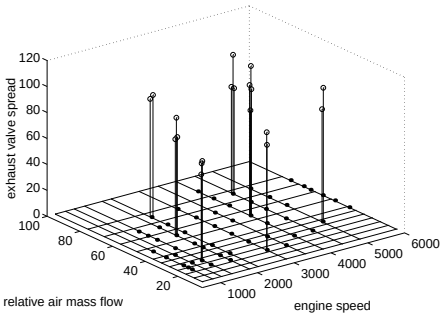

Fig. 6. The application: Operating range, operating points, grid (i.e. ECU operating points) and some exhaust valve spread candidates

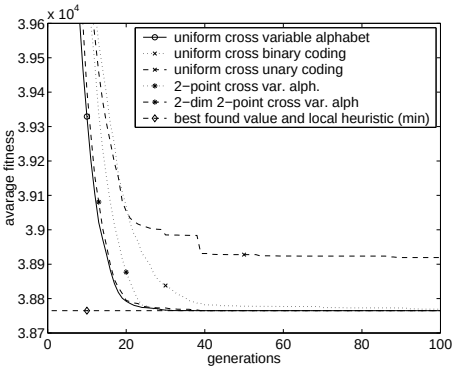

Fig. 7. Fitness curves for the application

inlet valve spread and one for the exhaust valve spread. Both valve spreads are mechanically adjusted actuators. Unfortunately, the operating points needed for the lookup tables are no more the equal to the operating points at which the candidates are available (see Fig. 6). Moreover, for the different candidates at each operating point, the location in the operating range varies slightly. I.e. two different candidates for one operating point have not only different exhaust valve spread values, but also slightly different engine speed and air mass values, as can be observed in Fig. 6.

As a consequence, when a selection of candidates is given, one has to apply appropriate interpolation and extrapolation algorithms in order to obtain values for the grid points. After that, one can apply the fitness function for evaluating the map. Here the next question arises: Which is the right smoothness criterion? Clearly there are many possibilities. We will consider a very simple criterion by integrating the square gradients of the two maps:

$$E = \int_{\substack{operating\\range}} \nabla y_{inlet}^2 + \int_{\substack{operating\\range}} \nabla y_{exhaust}^2.$$

Even with this simple formula, the local structure of the energy function becomes unclear with respect to the operating points because of the interpolation and extrapolation process. Thus a local search operator is much less efficient than before, since it cannot save computation time by directly exploiting the local structure. Instead it has to use complete evaluations of the energy function, which is quite expensive.

In terms of multi-objective optimizition, we thus use a simple aggregation method. Clearly, other more sophisticated multi-objective techniques can be expected to produce further interesting results, in particular for growing number of maps m. This should be subject of subsequent research.

We present the experimental results for the data set sketched in Fig. 6. There are 55 operating points endowed with candidates, the maximum number of candidates for one operating point is 4, the grid has dimension 12×12. This is a relatively small data set from the real-world application. Again, we compare variable alphabet coding with uniform crossover, 2-point crossover and 2-dimensional 2-point crossover as well as unary and binary coding with uniform crossover. Since the operating points do not form a grid this time, we used the grid construction algorithm from [8] in order to obtain the 2-dimensional arrangement.

The following GA parameters were used: population size $\mu = 100$, number of generations $t_{max} = 100$, tournament selection with $q = 5$, crossover and mutation probability $p_{cross} = 0.6$ and $p_{mut} = 1/55$. Again, 30 runs have been performed for each setting. We didn't use a Memetic Algorithm, since the local search is quite expensive in this case: One call needs more function evaluations than a whole GA generation, and the local search cannot efficiently exploit the local structure of the problem, as discussed above. Nevertheless we include the results of the pure local search.

Figure 7 shows the average GA fitness curves. This smaller instance is obviously easier to optimize than the 25×25 test instances. Again, direct encoding performs considerably better than both unary and binary coding. This time the best binary or unary results are worse than the worst variable alphabet results after generation 37. The optimal value is found by the variable alphabet GA with any crossover operator in all of the 30 runs. Hence we assume this value to be the global optimum. The binary coded GA obtains it in 27, the unary coded GA in 20 out of 30 runs. The pure local search finds the optimum in 20 out of 30 runs. This seems quite good, but the heuristic does not scale up nicely, it fails to optimize larger instances. And it needs quite many function evaluations, as already mentioned.

This time, the plot indicates that uniform crossover performs slightly better than the 2-dimensional 2-point crossover. This may be due to the fact that the 2-dimensional structure of the operating points is weaker than before. The average running time of the GA is about 7 min, independently of the representation. This is a consequence of the quite complex fitness function, where the decoding time becomes negligible.

4 Discussion

Our study covers two different aspects. On the one hand, the GA is able to optimize the application we started with in a satisfactory way. The quality of the resulting maps is similar or even better compared to the maps that were previously obtained manually by an engineer, a process which took several hours. On the other hand, we presented an easy-to-state problem with interesting features. We studied the performance of different codings and showed that a direct encoding is more suitable for a GA. We also tried a 2-dimensional encoding and saw that it can slightly accelerate the GA performance. This confirmes prior results of different researchers, who find that a d-dimensional encoding can result in a

moderate, but no significant performance gain. Finally, hybridization of a GA with a local search yields superior results, if the local search can be performed efficiently.

Acknowledgments

This research has been supported by the BMBF (grant no. 01 IB 805 A/1).

References

1. 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.
2. C. De Simone, M. Diehl, M. Jünger, P. Mutzel, G. Reinelt, and G. Rinaldi. Exact ground states of Ising spin glasses: New experimental results with a branch and cut algorithm. *Journal of Statistical Physics*, 80:487–496, 1995.
3. J. Poland. Finding smooth maps is NP-complete. Preprint, 2001.
4. F. Barahona. On the computational complexity of ising spin glass model. *J.Phys:A: Math.Gen.*, 15:3241–3253, 1982.
5. D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.
6. J. Holland. *Adaptions in Natural and Artificial Systems.* Ann Arbor: The University of Michigan Press, 1975.
7. T. N. Bui and B. R. Moon. On multidimensional encoding/crossover. In *6th International Conference on Genetic Algorithms*, pages 49–56, 1995.
8. J. Poland, K. Knödler, and A. Zell. On the efficient arrangement of given points in a rectangular grid. In E. J. W. Boers et al., editor, *Applications of Evolutionary Computation (LNCS 2037)*, pages 110–119, 2001.