Applied Intelligence 23, 55–70, 2005

Evolutionary Multiobjective Design in Automotive Development

MARCO LAUMANNS
ETH Zürich, Institute for Operations Research, CH-8092 Zürich, Switzerland
laumanns@ifor.math.ethz.ch

NANDO LAUMANNS
RWTH Aachen, Institut für Kraftfahrwesen, D-52074 Aachen, Germany

Abstract. This paper describes the use of evolutionary algorithms to solve multiobjective optimization problems arising at different stages in the automotive design process. The problems considered are black box optimization scenarios: definitions of the decision space and the design objectives are given, together with a procedure to evaluate any decision alternative with regard to the design objectives, e.g., a simulation model. However, no further information about the objective function is available. In order to provide a practical introduction to the use of multiobjective evolutionary algorithms, this article explores the three following case studies: design space exploration of road trains, parameter optimization of adaptive cruise controllers, and multiobjective system identification. In addition, selected research topics in evolutionary multiobjective optimization will be illustrated along with each case study, highlighting the practical relevance of the theoretical results through real-world application examples. The algorithms used in these studies were implemented based on the PISA (Platform and Programming Language Independent Interface for Search Algorithm) framework. Besides helping to structure the presentation of different algorithms in a coherent way, PISA also reduces the implementation effort considerably.

Keywords: multiobjective optimization, evolutionary algorithms, engineering design, automotive development

1. Introduction

Like many areas of engineering design, vehicle development has become an increasingly complex process in recent years. Engineers must meet conflicting demands concerning efficiency, performance, costs, etc. Vehicle design problems are typically characterized by the presence of multiple decision criteria or design objectives. Additional difficulties arise due to the increasingly complex system models used in the design process. Computer simulation is often used to model parts of the system to be designed as well as its environment since experiments with the real system are time consuming and expensive. The complexity of the model often makes it impossible for the designer to understand its input-output mapping, thus it must be regarded as a black box.

Approaches to cope with difficult design problems have traditionally been categorized into experimental and analytical methods. Experimental methods generally rely on the intuition and experience of the engineer and often follow a trial-and-error strategy. Analytical methods represent a more formal and systematic approach, but are normally limited to very simple models. With the availability of sufficient computing resources, numerical optimization methods have gained much popularity as a new option to deal with engineering design problems. The focus of this study are evolutionary algorithms (EAs), a special type of probabilistic numerical optimization techniques [1].

The use of evolutionary algorithms in engineering design is not new (see, e.g., [2–4] for overviews). EAs possess several properties that make black box design problems one of their primary application areas:

(i) they are easy to describe and implement, (ii) they do not make any assumptions about the properties of the optimization problem (such as differentiability, continuity, unimodality) or type of design variables, and (iii) they process sets of solutions in parallel and thus are able to obtain several different design alternatives simultaneously. The third property is often regarded as the key advantage of EAs over other numerical optimization techniques for problems with multiple design objectives. The recent monographs of [5] and [6] give a detailed survey of multiobjective evolutionary algorithms (MOEAs) including application examples.

The multitude of different MOEA versions introduced in the literature makes it difficult for a user, who is often not an EA expert, to choose a suitable algorithm. Many algorithms are published in conjunction with a certain application problem and hence contain many problem-specific operators, which might be irrelevant and confusing for a different user. In addition, state-of-the-art MOEAs are much more complex than standard EAs, resulting in a tedious and error-prone implementation effort.

The aim of this paper is to give a practical introduction to multiobjective black box optimization, used in engineering design processes, through real-world examples from the field of automotive engineering. To overcome the aforementioned difficulties, we employ the PISA framework [7]. The concept of PISA is to split the optimization into a problem-dependent and a problem-independent part. The use of PISA serves two purposes here. On the didactical side, the principle of separation of concerns facilitates the users' comprehension of the algorithms and hence their ability to choose and implement. On the practical side, the standardized interface specification of PISA allows to use existing algorithms with minimum programming effort. It also enables to easily combine the optimization algorithms with the application problem at hand.[1]

This article is therefore intended to be both practical and readily comprehensible such that an engineer with little knowledge of evolutionary computation and the mathematics of multiobjective optimization can benefit from it. The focus is to clarify which algorithms can be applied and how they might be implemented from a practitioner's perspective. On the other hand, interested readers will find selected research issues in evolutionary multiobjective optimization, demonstrated through practical examples, together with references for further study. First, we introduce the necessary mathematical and methodological background of multiobjective design with evolutionary algorithms and the PISA framework. The remaining sections describe multiobjective decision problems that arise at different stages in the design process of automotive systems and demonstrate the use of the proposed methodology. The first application explores the design space of road trains. Since road trains are a new concept in the European freight sector, it is necessary to explore their potential concerning various economic and environmental criteria at a preliminary design stage. The algorithm applied is extremely simple and suitable for self-implementation. We address the question of how to obtain a well-distributed and diverse set of design alternatives in problems with a fairly large number of objectives. In addition to that we introduce a simple technique of density-based selection. The second application area is a parameter optimization of adaptive cruise control systems. Here, the task is to develop a filter structure for optimal controller behavior regarding driving performance, safety and fuel consumption. On the algorithmic side, the focus is on the question of how to represent the design alternatives in the MOEA, and how to define appropriate variation operators. In order to compare different algorithms it is necessary to complete a performance assessment of MOEAs, which will exemplarily be discussed on this problem. Finally, a system identification problem is addressed. The aim is to fit a vehicle dynamics simulation model to data acquired in real driving tests. A simple, but realistic, model is needed for later integration into a vehicle dynamics controller. For this application, convergence problems are observed with standard MOEAs—a topic that has so far only been discussed in theoretical studies. To solve this problem, a recently developed selection operator is applied.

2. Multiobjective Design by Evolutionary Algorithms

Many design problems involve several criteria or objectives according to which the alternatives are evaluated. Objectives can be incommensurable, meaning they are not comparable with respect to magnitude and value. They can also be non-cooperative, meaning there is no single alternative that is better than all other alternatives in each objective. If set of alternatives is explicitely given, small, and finite, methods from the area of multiattribute decision analysis [8–10] can be used to aid the choice process under multiple criteria.

In engineering design problems, though, the set of different alternatives is usually not given explicitly, but rather implicitly via design variables and system constraints. The multiobjective decision problem then becomes a search or optimization problem and can be represented in its general form as

$$\text{minimize} \quad \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})) \qquad (1)$$

$$\text{subject to} \quad \mathbf{x} \in X, \qquad (2)$$

where X denotes the set of feasible design alternatives (design space) and $\mathbf{f}$ the vector of the $m \in \mathbb{N}$ objective functions. As there is rarely a single solution that minimizes all components of $\mathbf{f}$ simultaneously, the goal is to find elements of the so-called Pareto-optimal set.

Definition 1 (Pareto optimality). Let $\mathbf{f} : X \to F$, where X is called decision space and $F \subseteq \mathbb{R}^m$ objective space. A decision alternative $\mathbf{x}^* \in X$ is Pareto optimal if there is no other $\mathbf{x} \in X$ that dominates $\mathbf{x}^*$. $\mathbf{x}$ dominates $\mathbf{x}^*$, denoted as $\mathbf{x} \prec \mathbf{x}^*$ if $f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$ for all $i = 1, \ldots, m$ and $f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$ for at least one index i. The set of all Pareto-optimal decision alternatives X^* is called Pareto set. $F^* = \mathbf{f}(X^*)$ is the set of all Pareto-optimal objective vectors and denoted as Pareto front.

The Pareto set represents the collection of all reasonable alternatives, independent of the relative importance of the different objectives for the decision-maker. The knowledge of the Pareto set is useful for the decision-maker because it reveals much about the design problem at hand and about the trade-offs between the different objectives. It is obvious that any final solution should be Pareto optimal, but the decision as to which alternative to choose is subjective and depends on the decision-makers' preferences.

Most numerical methods to deal with multiobjective optimization problems are restricted to a certain type of problem (see, e.g., [11] for linear objectives, [12] for nonlinear objectives with continuous variables, or [13] for combinatorial problems). Engineering design problems, however, do not necessarily fit into those categories. For the purpose of this article, we do not want to restrict the type of problem and we assume the designer can provide (i) a definition of the decision space, (ii) a definition of the design objectives, and (iii) a system model that constitutes the mapping from decision alternatives to objective values. This is the scenario of black box optimization and the reason we apply evolutionary algorithms.

Evolutionary algorithms are probabilistic search techniques inspired by models of natural evolution. EAs work by representing different decision alternatives as *individuals*, which undergo cycles of *variation* and *selection*. The variation operator usually consists of *recombination* (the exchange of information between individuals) and *mutation* (the random alteration of individuals). The selection operator is used to grade the decision alternatives represented by the individuals based on their objective function values. The best individuals are kept for the production of offspring, while the worse alternatives are discarded.

A prevalent obstacle for the application of evolutionary algorithms is the practical integration of the optimization algorithm with the system model. Both the system model and the optimization algorithm are typically available as independent computer programs; however, neither are necessarily implemented in the same programming language or on the same operating system. Partial re-implementation of the optimizer or the system model is time consuming and error prone, thus it is desirable to keep both parts as separate and freely combinable components, as indicated in Fig. 1. For the interaction of these two components we make use of PISA, a platform-independent and programming language-independent interface for search algorithms, which has been developed to facilitate and standardize the integration of application problems and iterative optimization algorithms [7]. The purpose of PISA is to split the optimization component into two logically independent elements, 'variator' and 'selector', which operate with sets of decision alternatives denoted as 'populations'.[2]

The *variator* is responsible for the production of solutions. It can either create new solutions from scratch ('initial population') or modify existing solutions ('offspring population'). The variation of solutions takes place in the decision space X and is therefore application-specific. The variator is also responsible for calculating the objective values. This can be achieved either by integrating the system model into the variator or by invoking a separate simulation program.

The *selector* is responsible for deciding which solutions to discard and which to keep for further exploration during the search process. The selection is based on the objective values (which are provided by the variator as described above) and is thereby application-independent. The selector maintains an archive of all

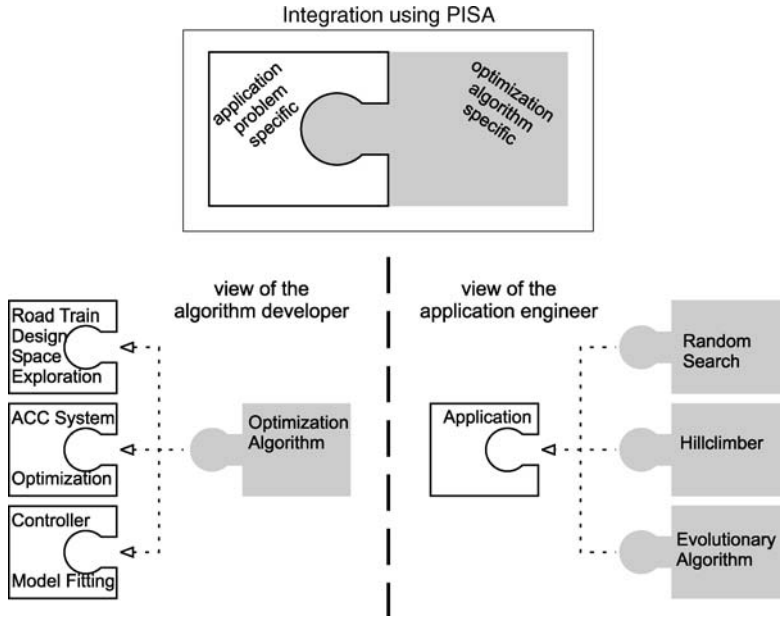

Figure 1. Illustration of the concept underlying PISA. The application problems on the left hand side and the multiobjective selection schemes can be combined arbitrarily.

solutions to be kept ('archive population'), from which it samples promising individuals for further modification by the variator ('parent population').

PISA requires that variator and selector are implemented as independent processes that exchange information via text files. The only common parameters to be specified are the size of the initial population α, the size of the parent population λ, the size of the offspring population μ and the number of objectives dim. For further details on the PISA protocol refer to [7].

3. Design Space Exploration of Road Trains

The first example of a development task solved by the described approach is the adaptation of a normal truck's power train to suit a road train [14]. Increasing the maximum payload is one possible approach to overcome traffic problems on crowded European highways. We focus on a concept for European freight traffic featuring two semi-trailers connected by a one-axle dolly [15].

The optimization of a new vehicle concept with respect to fuel consumption and driving dynamics is a very complex subject. The lack of existing data and knowledge leaves a void in experiments concerning the power train and the overall weight of the road train. Furthermore, it is impossible to acquire knowledge in driving tests as prototypes are too expensive to build. Therefore, the vehicle concept is modeled by a vehicle simulation.

3.1. Optimization Problem

Several considerations must be taken into account when developing a new vehicle concept. On one hand, an optimal combination of vehicle weight and engine power has to be found to ensure efficient driving. On the other hand, the correct choice of gear box type and gear ratio influence driving comfort and performance. The design variables and their ranges are displayed in Fig. 2. The first four variables are scale factors of different engine and gear box parameters. The last variable, x_5, represents the choice of the gear box type, specifically, whether the 15th or the 16th gear is chosen to be the direct gear. The direct gear is most efficient because the power is not transmitted through toothed wheels, but directly through the transmission shaft (gear ratio = 1). If the 15th gear is chosen as the direct gear, the 16th has a ratio below 1, which is less efficient. However, the low ratio causes higher engine rates and thus lower torque, which increases the gear box durability.

An increase in weight leads to an increase of road and climbing resistance. This changes the engine operating

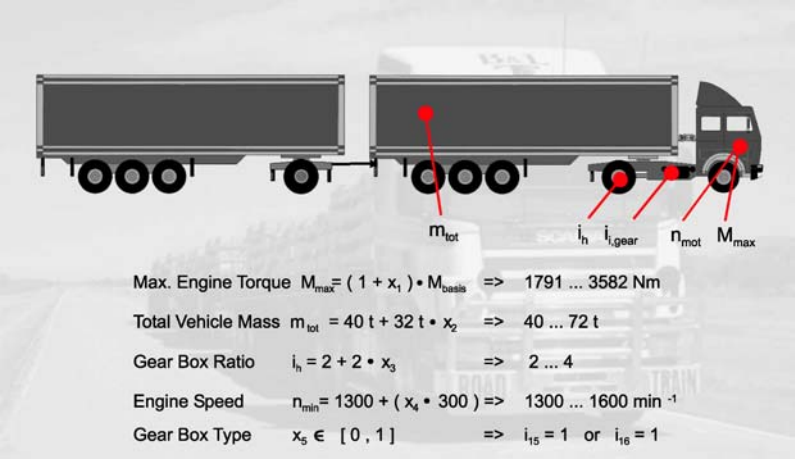

Figure 2. Schematic view of the road train and its design variables.

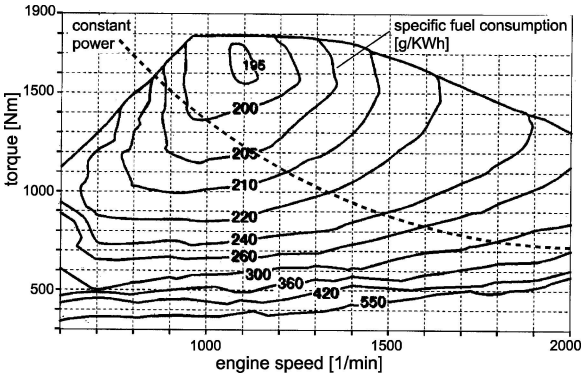

Figure 3. Engine characteristic graph.

point as well as its efficiency and fuel consumption. Every driving condition defines a point in the engine characteristic graph (see Fig. 3). The number of revolutions is determined by the velocity of the vehicle and the total gear ratio (consisting of rear-axle ratio and transmission ratio). The required torque is a result of power output (influenced by velocity, efficiency of the gear box, acceleration, and road gradient) and revolutions. A lower total gear ratio reduces the engine speed. Assuming constant running resistance (due to constant velocity and road gradient) the required power remains unchanged. The line of constant power indicates this relationship in Fig. 3. Long-distance transport vehicles usually drive statically, operating at their maximum authorized speed. This leads to the assumption that the gear ratio should be low enough to create an engine operating point in the area of lowest specific fuel consumption. This area is close to the line of maximum torque. Small increases in the running resistance, resulting from headwind or road gradient, cannot be compensated for by requesting more torque from the engine. Instead, they force the driver to shift gears or to go at a lower speed. Since the drivability of the vehicle requires a large distance between the most frequent engine operating point and the line of maximum torque (resulting in powerful engines and high gear ratios), it counteracts the attempt to reduce fuel consumption.

The goal of the optimization is to find a combination of overall weight, gear box, engine and driving strategy that minimizes fuel consumption and optimizes driving performance and driving convenience. Ten objective functions are defined to give a complete characterization of the vehicle performance with respect to fuel consumption and drivability. The resulting problem can be stated as follows:

$$\text{Minimize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_{10}(\mathbf{x})),$$

$f_1(\mathbf{x}) = t_{a1}(\mathbf{x})$ [time for acceleration 0–40 km/h]

$f_2(\mathbf{x}) = t_{a2}(\mathbf{x})$ [time for acceleration 40–90 km/h]

$f_3(\mathbf{x}) = (-1) \cdot v_{\max}(\mathbf{x})$ [maximum velocity]

$f_4(\mathbf{x}) = (-1) \cdot v_{14}(\mathbf{x})$ [maximum velocity, 1.5 gradient, 14th gear]

$f_5(\mathbf{x}) = (-1) \cdot v_{16}(\mathbf{x})$ [maximum velocity, 1.0 gradient, 16th gear]

$f_6(\mathbf{x}) = c_{100}(\mathbf{x})$ [average fuel consumption per ton load, 100 km/h]

$f_7(\mathbf{x}) = c_{80}(\mathbf{x})$ [average fuel consumption per ton load, 80 km/h]
$f_8(\mathbf{x}) = (-1) \cdot v_{\text{ave}}(\mathbf{x})$ [average speed on a highway (including road gradient)]
$f_9(\mathbf{x}) = c_{\text{h}}(\mathbf{x})$ [average fuel consumption per ton load on a highway]
$f_{10}(\mathbf{x}) = g_{\text{tot}}(\mathbf{x})$ [number of gear shifts on a highway]

subject to $\mathbf{x} = (x_1, \ldots, x_5) \in X = [0, 1]^5$.

The characteristic values t_{a1}, t_{a2}, $v_{\max}$, v_{14}, v_{16}, c_{100}, c_{80}, v_{ave}, c_h, g_{tot} are derived by simulation and therefore cannot be given in closed form. Six simulation scenarios are used, a full-load acceleration scenario, two constant-velocity scenarios (80 km/h and 100 km/h), two scenarios with constant gradient and the engine operating at full load and a highway scenario. The highway scenario consists of an 18 km drive over an empty highway, with road gradient varying from −4.5% to +3.9%.

3.2. Algorithms

This application is an example of design space exploration at an early stage. Here, simplicity is a main criterion to select a suitable optimization algorithm if the designer needs quick results and has to implement the algorithm himself. A further, technical requirement for this specific problem is that the archive population must be able to store a large number of individuals because the number of non-dominated solutions usually increases with the number of objectives. In this case, the archive size does not have to be bounded at all because the long duration of the simulation (about 30 seconds) already limits the total number of alternatives that can be evaluated in a reasonable amount of computing time. Our simple replacement strategy merely keeps all non-dominated solutions. To avoid genetic drift and an oversampling of already sufficiently explored regions, density-based selection [16, 17] is used to determine the parent population. In each iteration, the local density of each element of the archive population is estimated using a simple histogram-based method: a hyper-grid is defined in the objective space, and each individual is assigned the total number of individuals occupying its grid cell. The individual for the parent population is then drawn (with replacement) from the archive population with a probability reciprocal to this value.

The variation operator for this study only uses mutation. Each individual represents a decision alternative by a vector of design variables $\mathbf{x} = (x_1, \ldots, x_5) \in X$. For each component of the decision vector, a random number is drawn from a standard normal distribution and multiplied with a scaling factor σ. This product is then added to the old component x_i to form the new component x_i':

$$x_i' := x_i + \sigma \cdot r_i, \quad r_i \sim N(0, 1) \tag{3}$$

A constant mutation step size of $\sigma = 0.02$ is used for simplicity, i.e., two percent of the range of each design variable. Recombination is not used here since the interdependence of the design variables in every part of the objective space seems to be very high. The variation and selection operators are summarized in Table 1.

Table 1. PISA specification of the EA for the road train problem.

PISA Parameters: $\alpha = 20$ $\mu = 1$ $\lambda = 1$ dim = 10	
Variator	Selector
Initial population: Draw each decision vector uniformly from X.	**Archive population:** Keep all non-dominated individuals, discard dominated ones.
Offspring population: Mutate each parent decision variable by adding a normal random variable with zero mean and standard deviation 0.02.	**Parent population:** Draw parent individual inversely proportional to its local density in objective space.

To judge the quality of the design alternatives obtained by the evolutionary algorithm, a road train version is designed in a traditional way, based on simple rules for optimizing a truck's power train [18]. In addition, two grid searches over the whole design space are performed, each with a total number of 2160 elements. One of them is restricted to a maximum authorized speed of 80 km/h, the other to 100 km/h.

3.3. Results

A hierarchical approach is used for the design space exploration with the evolutionary algorithm. The first run of the evolutionary algorithm is performed to narrow the design variable intervals. An analysis of the trade-offs between the different objectives leads to the conclusion that a focus on reducing fuel consumption does not necessarily worsen the other objective values in an unacceptable way. Furthermore, this goal is the

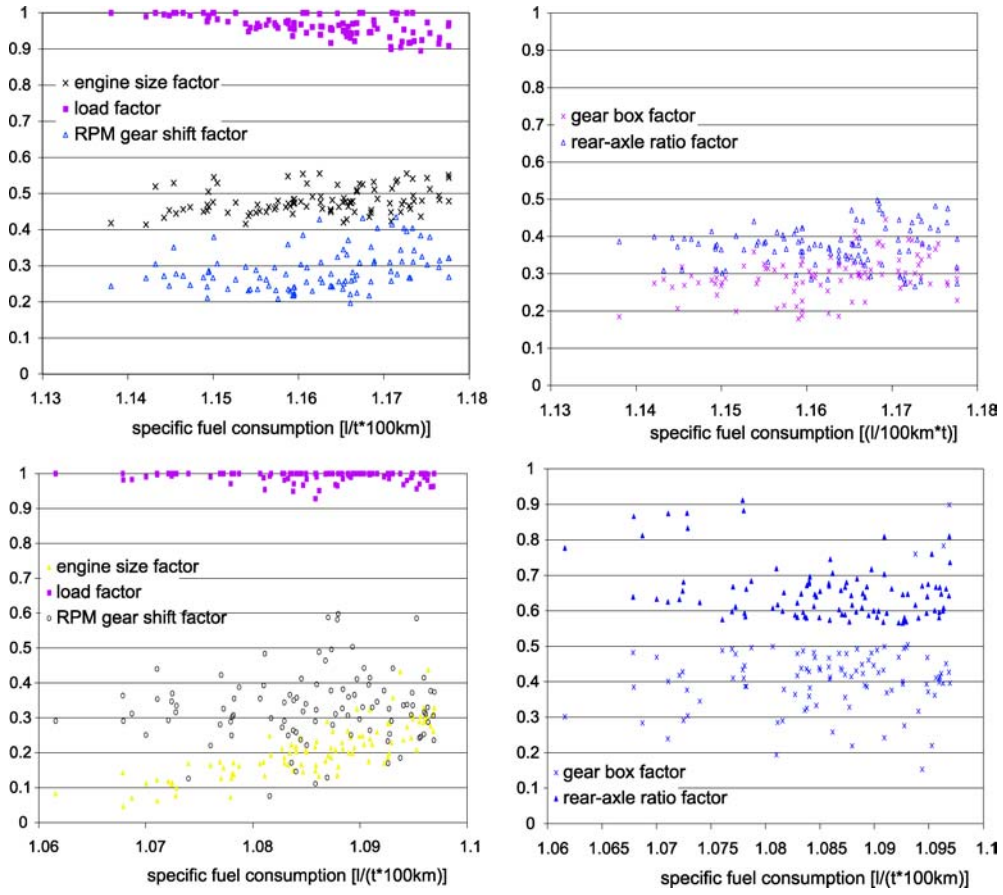

Figure 4. Design variable and specific fuel consumption for the 100 km/h road train (top) and the 80 km/h road train (bottom).

main factor for the profitability of a vehicle concept and deserves special attention. Therefore, we choose average fuel consumption on the highway, f_9, as the objective value that defines a ranking of the solutions; f_4 and f_5 can be used to represent the second important part of driving performance, the required climbing ability. Here, reduction of maximum velocity must not exceed 5 km/h. Solutions that do not meet this criterion are removed from the ranking. With the help of these preliminary solutions, shown in Fig. 4, the design variable intervals are narrowed down to $x_1 \in [0.4, 0.6], x_2 = 1, x_3 \in [0.3, 0.4], x_4 \in [0, 0.5], x_5 = 0$ for the 100 km/h version and $x_1 \in [0.0, 0, 4], x_2 = 1, x_3 \in [0.55, 0.85], x_4 \in [0, 0.5], x_5 = 0$ for the 80 km/h version. Limited to those intervals, a second run of the same evolutionary algorithm performs a more exact approximation of the Pareto set in the region of interest. Of course, there are other ways to cope with the large number of incomparable alternatives in the presence of many objectives. These typically rely on preference information, for instance aggregating (or dropping) objectives, lexicographic ordering or the transformation of objectives into constraints. In many cases, however, it is very difficult to derive an exact numerical representation of the preferences. Moreover, since we have different decision-makers with different preferences in mind, the aim is first to explore the Pareto set as broadly as possible with a minimum number of simulations before exploiting interesting regions through restricting the decision variable space as described above.

Final results show the impressive advantage of road trains over normal trucks with respect to fuel consumption: decreases of 23% (80 km/h-version) and 26% (100 km/h-version) are achieved on highways in spite of the rather tough gradients. In steady-state operation, fuel consumption advantages of up to 35% are accomplished. When acceleration is at a sensible level, the road trains have no disadvantages in climbing ability and required gear shifts. The comparison of the different road train versions (see Fig. 5) indicates that the

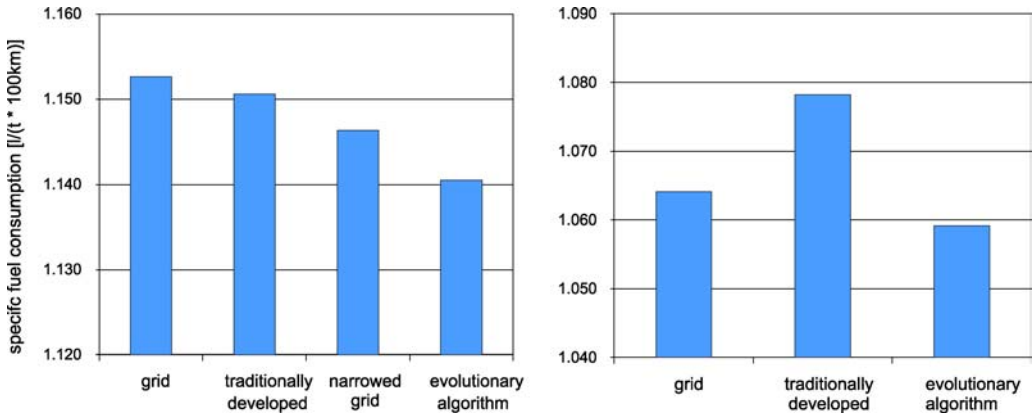

Figure 5. Specific fuel consumption on a highway for the 100 km/h road train (left) and the 80 km/h road train (right).

evolutionary algorithm is able to generate better solutions than the other approaches. Showing the same climbing ability and acceleration as both the traditionally developed versions and the ones obtained by a grid-scan over the whole parameter area, the EA-solution needs about 1% less fuel on the highway. The 100 km/h version is even better than the best version found by a grid-scan of 1000 elements distributed over the narrowed intervals.

Figure 6 shows the relation between the objective function f_4 (maximum velocity in 14th gear with 1.5% road gradient) and f_9 (specific fuel consumption on highway) for elements of the archive population at the end of the run. This relationship provides information about the trade-off between driveability and fuel economy. The creation of 1300 individuals already produces a rather large number of solutions, which must be considered better than any solution that was found without the evolutionary algorithm. This advantage in efficiency becomes even more important when more sophisticated driving scenarios—and thus more time consuming simulations—are used, which is subject to further research.

4. Parameter Optimization of Adaptive Cruise Control Systems

Crowded motorways and a higher average vehicle speed create increasing difficulties for drivers. The

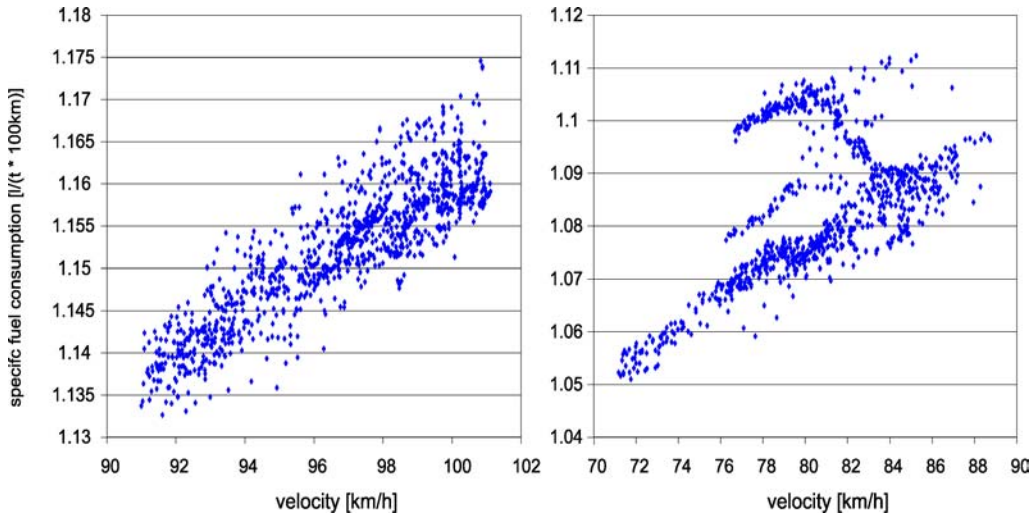

Figure 6. Trade-off between velocity (1.5% road gradient) and specific fuel consumption on a highway for the 100 km/h road train (left) and the 80 km/h road train (right).

automobile industry tries to compensate for these additional demands by inventing driver assistance systems such as antilock braking systems (ABS), cruise control (CC) and electronic stability control (ESC). In contrast to the systems mentioned above, adaptive cruise control (ACC) has not been thoroughly established yet.

The ACC-system is an enhanced cruise control system, not only designed to keep the vehicle's speed constant, but also to analyze the traffic situation in front of the vehicle and regulate its longitudinal dynamics accordingly. Thus, it especially suits the demands of truck drivers, who frequently have to follow a leading vehicle. Used effectively, ACC-systems can increase driving safety, make driving more comfortable and reduce fuel consumption. However, it is difficult to develop a controller that meets the drivers' requirements concerning its speed-regulating behavior as well as safety criteria and fuel efficiency.

Since experimental testing of each modified controller variant would enormously raise development costs and time, the ACC-system's behavior is evaluated and analyzed by simulation. This offers the possibility to improve the development process further by applying numerical optimization techniques such as evolutionary algorithms to optimize the ACC-controller [19].

4.1. *Optimization Problem*

The structure of the ACC-system is depicted in Fig. 7. The longitudinal controller translates incoming data about the traffic situation in front of the vehicle and its own driving condition into the desired acceleration. The data produced by the sensor contain some deviation. As such, it requires four different filters in order to create a smooth acceleration signal. The influence of these filters can be regulated by four integer parameters, represented by the design variables $x_1, \ldots, x_4$. Strong filters result in very smooth signals. However, they delay the vehicle's reaction to incoming data, which weakens its driving performance. Two further design variables, x_5, x_6, are used to define the longitudinal controller's reaction to the vehicle's distance from the leading vehicle and their relative velocity.

Four objectives are defined to give a sufficient characterization of the ACC-system's longitudinal controlling behavior with respect to driving comfort, fuel efficiency and safety. These objective functions are computed within the simulation. Thus, the resulting multiobjective optimization problem can be stated as follows (where for a given design alternative, its characteristic values c_{ave}, t_{acc}, d_{vel}, and d_{acc} are calculated by the simulator):

$$
\begin{aligned}
&\text{Minimize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_4(\mathbf{x})),\\
&f_1(\mathbf{x}) = c_{\text{ave}}(\mathbf{x}) \quad \text{[average fuel consumption]}\\
&f_2(\mathbf{x}) = t_{\text{acc}}(\mathbf{x}) \quad \text{[time for acceleration]}\\
&f_3(\mathbf{x}) = d_{\text{vel}}(\mathbf{x}) \quad \text{[velocity deviation]}\\
&f_4(\mathbf{x}) = d_{\text{acc}}(\mathbf{x}) \quad \text{[acceleration deviation]}\\
&\text{subject to } \mathbf{x} = (x_1, \ldots, x_6) \in X = \{1, 2, \ldots, 99\} \times\\
&\{1, 2, \ldots, 16\} \times \{1, 2, \ldots, 8\}^3,\\
&\quad g(\mathbf{x}) \geq d_{\min} \quad \text{[minimum follow-up distance]}
\end{aligned}
$$

4.2. *Algorithms*

In order to approximate the Pareto set for the constrained multiobjective integer programming problem

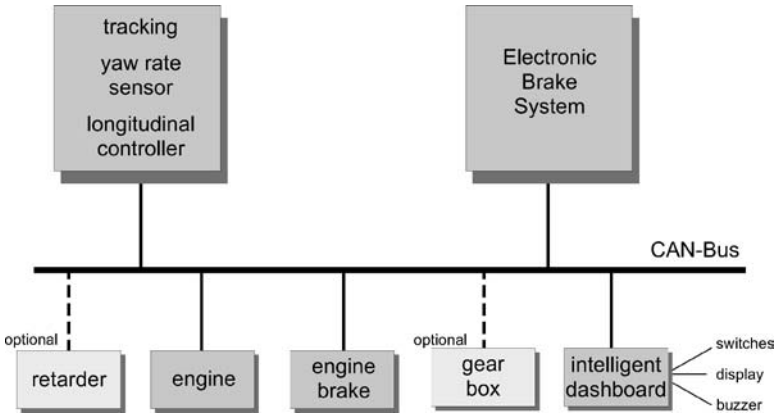

Figure 7. Structure of the ACC-system.

Table 2. PISA specification of the EA for the ACC controller optimization.

PISA Parameters: $\alpha = 20$ $\mu = 10$ $\lambda = 10$ $dim = 4$	
Variator	Selector
Initial population: Draw each decision vector uniformly from X.	**Archive population:** Environmental selection of SPEA2 [20]
Offspring population: Read-coded version: SBX operator [5] Integer version: Geometric Distribution with self-adaptation [21]	**Parent population:** Mating selection of SPEA2 [20]

above, a grid search and two evolutionary algorithms are applied and compared. The computation time of the simulator makes an exhaustive search or complete enumeration of all alternatives impractical. Thus, a grid search with 2^{15} representative solutions (regularly distributed in the decision variable space) is performed. In comparing all these alternatives to each other, the dominated ones are eliminated and the remaining represent a first approximation of the non-dominated set as a baseline for comparison.

Here, two algorithms based on SPEA2 ([20], an improved version of the Strength Pareto Evolutionary Algorithm, [22]) are applied. The PISA specification is given in Table 2. Also in SPEA2, selection is performed in two steps: environmental selection (to determine the new archive population) and mating selection (to determine the new parent population). The best α individuals out of the old archive population and the λ new offspring survive. First, all non-dominated individuals are selected. If there are more than α such solutions, a truncation procedure is invoked which iteratively removes the individual closest to the others. If less than α individuals are non-dominated, the space is filled with the dominated individuals in ascending order of their fitness values. In the mating selection step, the μ-sized parent population is created by binary tournament selection (with replacement) based on the fitness values.

One purpose of this study is to compare different representations for the individuals, a real-valued relaxation of integer design variables and a direct integer coding. Accordingly, two different variation schemes are used:

Real-valued individuals. Many standard search operators are based on a floating-point representation of (real-valued) decision variables. Therefore, a continuous relaxation of the search space to $[0, 99]^2 \times [0, 16] \times [0, 8]^3$ is used, and the variables are rounded to their integer part (plus 1) before each run of the simulation tool. For the recombination, we use the SBX-operator [5] with distribution index $\eta = 5$. The offspring individuals are then mutated by adding normally distributed random numbers according to Eq. 3, where the standard deviation σ is set to 5 per cent of the interval length.

Integer-valued individuals. As the relaxation produces an artificially magnified search space, a direct representation of the decision variables as integer numbers seems more appropriate. It also eliminates the potential problem of mapping several different individuals to the same decision alternative by the rounding procedure. Search operators working directly on integer variables are not as common in evolutionary computation. We adopt the techniques from Rudolph [21], who developed an EA for integer programming with maximum entropy mutation distributions, enabling self-adaptive mutation control similar to real-valued evolution strategies. A successful application to a mixed integer design problem for chemical plants is reported in [23]. Here, the initial mutation step size was set to $s = 2$ for all variables.

Both versions of SPEA2 were terminated after 3000 objective function evaluations. During the run, an archive of all non-dominated solutions was maintained. At the end of the run the archive represented an approximation of the Pareto set.

4.3. Results

To evaluate the performance of the evolutionary algorithm, a grid search over the whole parameter range is performed, along with a manual optimization of the ACC-controller. The grid search contains 16384 elements, requiring a computation time of almost 137 hours.[3] Both instances of the evolutionary algorithm only used 3000 function evaluations each. Since their internal operations and data processing can be neglected compared to the simulation, they have a clear advantage in terms of computation time.

As a first interesting observation from the output of the different algorithms, no trade-off is visible for the second objective f_2 (acceleration/deceleration time). All algorithms have found the optimal value of 66.6 for almost all non-dominated alternatives. This is the

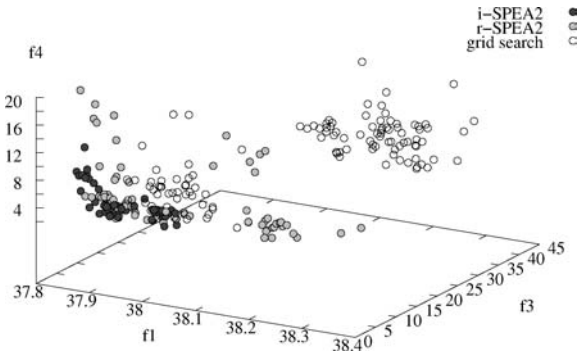

Figure 8. Scatter plot of the non-dominated solutions produced by the grid search and the evolutionary algorithm with continuous relaxation (r-SPEA2) and direct integer coding (i-SPEA) for the objective function values f_1, f_3, f_4.

optimal value attainable by immediate full acceleration, without any delays caused by the ACC system. Hence, it cannot be improved further. The remaining objective values of the different non-dominated sets are displayed in Fig. 8. The trade-off characteristic is visible from the three-dimensional scatter plot.

A further goal of this case study is to conduct a systematic performance assessment and comparison of the different techniques to exemplify the aforementioned theoretical results reported in [24]. We start with the *hypervolume* indicator [22] as an example of an absolute quality indicator with strong inferential power. The hypervolume indicator calculates the normalized volume of the dominated space to evaluate a single non-dominated set alone. As it requires a bounded objective space, a reference cuboid is defined between the ideal point $\mathbf{f}^*$ and the nadir point, given by the maximum objective function values of the maximum elements of the output of all three algorithms. The value $I_{\mathrm{H}}(A)$ gives the fraction of this reference volume that is dominated by A. It is intuitively clear that, the more space an algorithm can dominate, the better the algorithm. However, as it was proven in [24], a comparison based on such scalar indicator values does not allow us to conclude that one solution set is entirely better than the other in the sense that each element of the latter is dominated by at least one element of the former set. The results given in the last column of Table 3 show:

$$I_{\mathrm{H}}(A_{\mathrm{i-SPEA2}}) > I_{\mathrm{H}}(A_{\mathrm{r-SPEA2}}) > I_{\mathrm{H}}(A_{\mathrm{gridsearch}})$$

which allows to conclude

$$A_{\mathrm{gridsearch}} \prec A_{\mathrm{r-SPEA2}} \prec A_{\mathrm{i-SPEA2}}.$$

Table 3. Results of the binary hypervolume indicator I_{H2} applied to all pairs of algorithms and the absolute hypervolume indicator I_{H} (last column).

$I_{\mathrm{H2}}(A, B)$	i-SPEA2	r-SPEA2	grid search	$I_{\mathrm{H}}(A)$
i-SPEA2		0.0038	0.223	0.949
r-SPEA2	0.002		0.188	0.913
grid search	0.0003	0.0018		0.726

where the symbol $\prec$ denotes the extension of the dominance relation to sets of decision alternatives. These statements are quite weak, and we have to apply relative quality indicators to arrive at stronger statements. We consider two relative quality indicators proposed by [22], the *coverage* indicator I_{C} and the *binary hypervolume* indicator I_{H2}. Both indicators are among those with the strongest inferential power [24].

The coverage indicator provides information about how much of one algorithm's output has also been reached by the other algorithm. Specifically, $I_{\mathrm{C}}(A, B)$ calculates the relative number of points of set B that are dominated by at least one point in set A. Table 4 shows the results. It can be seen that none of the points found by the grid search is better than any point in the non-dominated sets of the evolutionary algorithms. The SPEA2, working with the floating point representation, does not cover many (less than 10%) of the solutions produced by the integer version, which in turn is able to dominate nearly half of the solutions of its competitor. However, as far as the dominance relations of solution sets are concerned, the results only lead to the conclusion that the output of all algorithms is mutually incomparable because all values of the coverage indicator are strictly smaller than one. The same conclusion can of course be drawn from the binary hypervolume indicators, whose results are also listed in Table 3. The binary hypervolume $I_{\mathrm{H2}}(A, B)$ evaluates to the volume dominated by set A, but not dominated by set B.

This situation of mutually incomparable approximation sets is very typical for a comparative study because

Table 4. Results of the coverage quality indicator I_{C} applied to the output of all pairs of algorithms.

$I_{\mathrm{C}}(A, B)$	i-SPEA2	r-SPEA2	Grid search
i-SPEA2		0.423567	0.991597
r-SPEA2	0.070588		0.991597
grid search	0	0	

the performance differences are seldom so strong that one set entirely dominates another. Nevertheless, the indicator values provide insight into differences between the outputs of the algoritms. In our case, the order of the indicator values is always the same, showing that both evolutionary algorithms largely dominate the output of the grid search, with the integer-valued version having a slight advantage over the real-valued one. Such conclusions can of course be drawn, although one has to be careful with the interpretation of results to avoid general statements such as "algorithm A is better than algorithm B" when this is formally incorrect.

We conclude this performance assessment by investigating further performance *aspects* in more detail. These aspects correspond to the different preferences of the designer, i.e., which regions of the objective space one is most interested in. Such an assessment is of course a *subjective* one. All of the following indicators can be seen as special cases of the distance-indicator I_{D} [24].

One possibility is to define a utility function based on a weighted distance to an ideal point $\mathbf{f}^*$, which is given by the minimum objective values in each dimension. The difference between each objective value and the optimum value in the corresponding category is multiplied with a factor that represents the importance of the category. Thus, the interpretation of the results reflects an adaptation to the decision maker's preferences. In this case, the objectives f_1 and f_3 are considered most important, while f_2 is least important. Representing the distance to the optimal solution, the sum of those values gives the overall quality of the individual

$$D(\mathbf{x}) = 150(f_1(\mathbf{x}) - f_1^*) + (f_2(\mathbf{x}) - f_2^*) + 6(f_3(\mathbf{x}) - f_3^*) + 4(f_4(\mathbf{x}) - f_4^*) \quad (4)$$

with $\mathbf{f}^* = (37.8339, 66.6, 2.06935, 3.03196)$. Accordingly, a ranking of the individuals developed by the different optimization strategies can be produced. The best 100 solutions are displayed in Fig. 9. The two evolutionary algorithms create the best solutions, while the integer-version holds a slight advantage in terms of density close to the optimum solution. Out of the top 100 solutions, 46 were created by this integer-version, 40 by the real-coded version, and only 14 by the grid-search.

5. Model-Fitting for a Vehicle Dynamics Simulation

A crucial element of the application process of simulation and optimization techniques in vehicle development is model-fitting. Here, a MATLAB simulation is to be fitted to data acquired in real driving tests.

The two most important elements of modeling are simplification and exactness. In this case, an extended bicycle model is used for vehicle driving dynamics research. The original bicycle model is a rather simple representation of real cars, because the four tire contact points are centralized in the longitudinal axle of the car. Time-delays for the lateral tire force generation

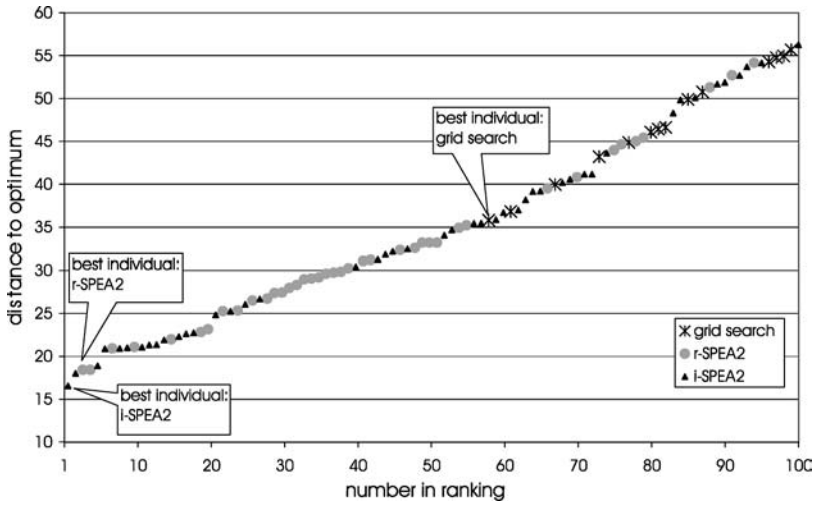

Figure 9. Ranking of solutions according to the scalar utility function (4).

enable a sufficiently exact reproduction of real vehicle measurements. However, it is necessary to fit a number of vehicle parameters in order to achieve a satisfying performance of the vehicle model.

5.1. Optimization Problem

The two transfer functions for yaw rate and lateral acceleration with the input steering angle are the most important criteria when analyzing the quality of lateral vehicle dynamics simulation. Both phase lag and gain must represent the original vehicle behavior with maximum precision.

The main task is to develop a model structure that enables sufficient exactness while remaining as simple as possible. To achieve that, the original bicycle model is extended. Tire phase lags create a delayed reaction to steering angle changes. In addition, the vehicle's rolling behavior is modeled in a simple way in order to represent the body movement in relation to the tires. The following eight real-valued design variables enable a sufficient adjustment to different vehicles: x_1 (yaw inertia), x_2 (tire stiffness, front), x_3 (tire stiffness, rear), x_4 (phase lag front tire), x_5 (phase lag rear tire), x_6 (roll stiffness), x_7 (roll damping), x_8 (roll inertia). These parameters mainly represent tire characteristics and vehicle mass distribution. The data for other parameters like vehicle mass and length can simply be measured. Therefore it is not necessary to include those values in the design parameter set.

Typical manoeuvres for vehicle parameter identification are steering angle sweeps with a constant lateral acceleration of about 4 $\mathrm{m/s^2}$. Both transfer functions mentioned above can be derived from those manoeuvres. A comparison of the simulation to driving tests results in the following four-objective optimization problem:

$$
\begin{aligned}
&\text{Minimize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_4(\mathbf{x})),\\
&\qquad f_1(\mathbf{x}) = a_{\mathrm{a}}(\mathbf{x}) \quad \text{[gain deviation, lateral acceleration]}\\
&\qquad f_2(\mathbf{x}) = t_{\mathrm{a}}(\mathbf{x}) \quad \text{[phase lag, lateral acceleration]}\\
&\qquad f_3(\mathbf{x}) = a_{\mathrm{y}}(\mathbf{x}) \quad \text{[gain deviation, yaw rate]}\\
&\qquad f_4(\mathbf{x}) = t_{\mathrm{y}}(\mathbf{x}) \quad \text{[phase lag, yaw rate]}\\
&\text{subject to } \mathbf{x} = (x_1, \ldots, x_8) \in X = [0, 1]^8.
\end{aligned}
$$

5.2. Algorithms

The problem to be solved is a multiobjective optimization problem with eight real-valued, normalized decision variables and an objective function with four components. As we are not primarily concerned with a comparison of different algorithms, we start directly with the SPEA2 described in the previous section.

The variation operators used for this problem again apply recombination and mutation. A simple discrete recombination is chosen, which creates one offspring solution $\mathbf{x}'$ from two parents $\mathbf{x}^{(a)}$ and $\mathbf{x}^{(b)}$. For each decision variable x_i, one parent is determined randomly and its decision variable copied to the child. The resulting child is then mutated using normal-distributed random variables, again according to Equation 3. The mutation step sizes σ are chosen in each iteration adaptively and determined by half the absolute difference of the parent variables:

$$\sigma_i := \frac{1}{2}\left|x_i^{(a)} - x_i^{(b)}\right|.$$

The first run of this SPEA2 version, however, soon reaches a situation where the population stagnates and no further progress is visible. Instead, the population oscillates around a certain area in objective space. The problem arises as a result of deterioration, which is discussed in [25]. To overcome this convergence problem, the selection operator of SPEA2 must be replaced by the selection operator maintaining the ϵ-Pareto set proposed in [25]. Such convergence problems, which have been verified for many multiobjective EAs, indicate that the algorithm is operating close to the Pareto set. To achieve further progress, special care must be taken regarding the selection and deletion of solutions from the archive population. Using this particular selection algorithm guarantees that the set of archived solutions never deteriorates and thus monotonously converges to the Pareto set. The PISA specification for this algorithm is given in Table 5.

5.3. Results

The evolutionary algorithm is able to find solutions of a sufficient quality rather quickly. Regarding the final approach to the Pareto set, the ϵ-archive selection algorithm performs considerably better than SPEA2. In the first part of the optimization process, the user can derive interesting information about the model

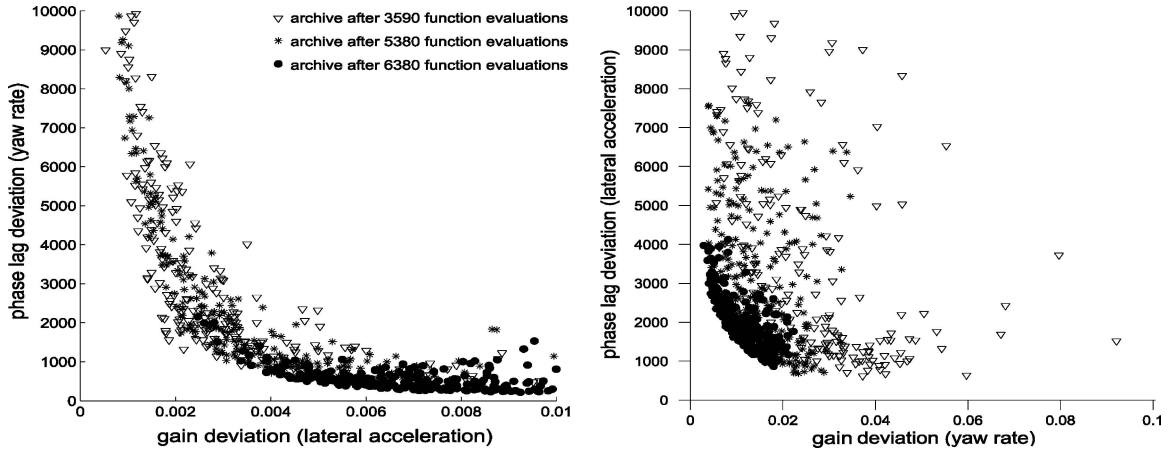

Figure 10. Selected trade-offs between the four objective functions.

behavior by analyzing the trade-offs between the objective functions, since the archive is still rather widely spread as visualized in Fig. 10.

In the course of the optimization, the focus of the designer shifts from diversity to examining the area of favoured solutions in detail. To achieve this goal, further constraints on the objective values are defined while the ϵ value is scaled down. This adaptation, however, is not automated, but user driven. The development of the solutions, especially the focus on favoured regions, is also visible in Fig. 10 for different stages defined by different ϵ values. The left diagram shows that in the third stage, more emphasis was put on improving the phase lag deviation of the yaw rate, so all solutions above a certain threshold were prohibited by an additional constraint. This measure subsequently led to a considerably better approximation of the lower part of the trade-off surface, keeping the gain deviation of the lateral acceleration in the interval between [0.002, 0.01].

The discovered solutions create a simulation environment, in which lateral vehicle dynamics can be

Table 5. PISA specification of the EA for the model-fitting problem.

PISA Parameters: $\alpha = 300$ $\mu = 10$ $\lambda = 10$ $dim = 4$	
Variator	Selector
Initial population: Draw each decision vector uniformly from X.	**Archive population:** Select archive population using the ϵ-archive selection algorithm [25].
Offspring population: For each pair of parents: swap each decision variable with probability 0.5, then mutate by adding a normal randomvariable with zero mean and standard deviation given by the absolute difference of the parent variables, divided by 2.	**Parent population:** Select parent population by sampling μ times uniformly from the archive population (with replacement).

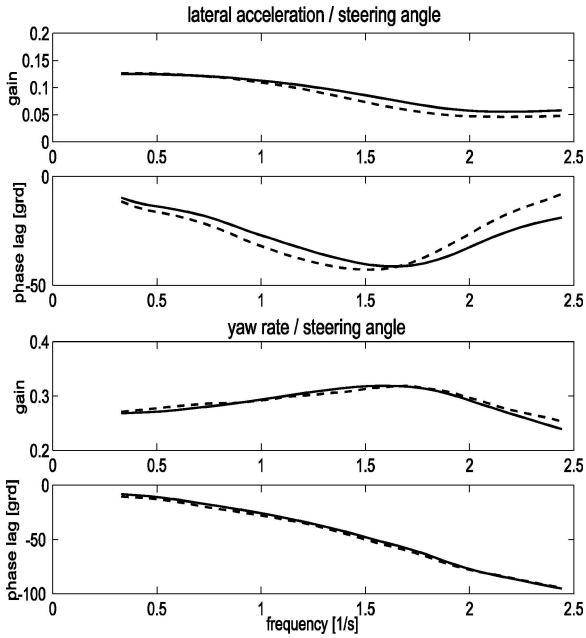

Figure 11. Comparison of simulated and real vehicle behavior. The solid line is an interpolation of the measured data from the real vehicle, the dashed line represents the simulation data obtained from the vehicle model.

simulated with a rather simple model and sufficient exactness. The exactness is depicted in Fig. 11, where the simulated time series is plotted against data of a real vehicle obtained in driving tests. The simplicity of the model results in low computation time, thereby enabling the model to be used when controlling vehicle dynamics online. This is true not only for a general purpose processor, but also in a vehicle-specific architecture, where it is impossible to run a full scale vehicle simulation.

6. Conclusion

We discussed three application case studies from the field of automotive engineering, where multiobjective evolutionary algorithms proved to be powerful optimization tools. The advantage of EAs, when applied to black box optimization problems of complex systems (along with the approximation of the Pareto-optimal set of design alternatives) are that they provide the development engineer with detailed information about the trade-offs between different objective functions. They also help clarify the problem at hand.

The road train example showed that even a simple approach is suitable for a design space exploration task at an early design stage. In addition to a detailed overview of the trade-offs between the ten objectives, the evolutionary algorithm was able to present a solution that dominates the one found by the engineer on a trial-and-error basis. The problem related to the design of an adaptive cruise control system was a parameter optimization of filters used in a controller. The goal was a comparison of the effectiveness of different variation operators. The comparative study was carried out based on previous results concerning quality indicators and performance assessments. The third case study was a model-fitting problem. Preliminary trials with a standard multiobjective EA revealed convergence problems. Therefore, a recently developed selection operator had to be applied to maintain an ϵ-Pareto set. Through a manual adjustment of the ϵ values, the approximation quality steadily increased in the area of interest to the designer.

The specification of the algorithms via the PISA framework helped us to present the basic components and essential features of a multiobjective evolutionary algorithm in a coherent and structured way. In addition, the PISA protocol made it possible to easily re-use and exchange the different components of the algorithm and therefore save a considerable amount of programming effort. This potential of PISA is still to be explored in further application areas.

Acknowledgments

The first author acknowledges funding from the Swiss National Science Foundation (SNF) under the ArOMA project 2100-057156.99/1.

Notes

1. Different MOEAs as well as different test and application problems are available both as source code and ready-to-use executables for different operating systems from the PISA website: www.tik.ee.ethz.ch/pisa.
2. Though the terminology of PISA is mainly borrowed from evolutionary algorithms, it can also be used with other iterative optimization methods.
3. This estimate is based on the average running time of the simulation on a PC with an AMD ATHLON 1800 processor.

References

1. T. Bäck, D.B. Fogel, and Z. Michalewicz (eds), *Handbook of Evolutionary Computation*, IOP Publishing and Oxford Univertity Press: Bristol, UK, 1997.
2. K. Deb, *Optimization For Engineering Design: Algorithms and Examples*, Prentice Hall of India, 1995.
3. P. Bentley (ed.), *Evolutionary Design by Computers*, Morgan-Kaufmann: San Francisco, 1999.
4. D. Dasgupta and Z. Michalewicz, Evolutionary algorithms in engineering applications, 1997.
5. K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley: Chichester, UK, 2001.
6. C.A. Coello Coello, D.A. Van Veldhuizen, and G.B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer: New York, 2002.
7. S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler, PISA—A platform and programming language independent interface for search algorithms," in *Evolutionary Multi-Criterion Optimization (EMO 2003)*, Lecture Notes in Computer Science, Springer: Berlin, 2003.
8. R.L. Keeney and H. Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, Wiley: New York, 1976.
9. D.E. Bell, R.L. Keeney, and H. Raiffa, *Conflicting objectives in decision. International Series on Applied Systems Analysis 1*, Wiley: Chichester, 1977.
10. G. Fandel and J. Spronk, *Multiple Criteria Decision Methods and Applications*, Springer: Berlin, 1985.
11. R.E. Steuer, *Multiple Criteria Optimization: Theory, Computation, and Application*, Wiley: New York, 1986.
12. K. Miettinen, *Nonlinear Multiobjective Optimization*, Kluwer: Boston, 1999.
13. M. Ehrgott, *Multicriteria Optimization*. Springer: Berlin, 2000.
14. N. Laumanns, M. Laumanns, and D. Neunzig, "Multi-objective design space exploration of road trains with evolutionary

algorithms," in *Evolutionary Multi-Criterion Optimization (EMO 2001)*, edited by E. Zitzler et al., Lecture Notes in Computer Science Vol. 1993, Springer, 2001, pp. 612–623.
15. J. Ludmann, D. Neunzig, M. Weilkes, and H. Wallentowitz, "The effectivity of new traffic-technologies and transportation-systems in suburban areas and on motorways," *International Transactions in Operational Research*, vol. 6, no. 4, pp. 423–439, 1999.
16. M. Laumanns, E. Zitzler, and L. Thiele, "On the effects of archiving, elitism, and density based selection in evolutionary multi-objective optimization," in *Evolutionary Multi-Criterion Optimization (EMO 2001)*, edited by E. Zitzler et al., Lecture Notes in Computer Science Vol. 1993, Springer, 2001, pp. 181–196.
17. R.C. Purshouse and P.J. Fleming, "Why use elitism and sharing in A multi-objective genetic algorithm?" in *Genetic and Evolutionary Computation Conference (GECCO 2002)*, edited by W.B. Langdon et al., Morgan Kaufmann Publishers, New York, July 2002, pp. 520–527.
18. H. Wallentowitz, *Longitudinal Dynamics of Motor Vehicles*, Forschungsgesellschaft Kraftfahrwesen mbH, Aachen, 2000.
19. N. Laumanns, M. Laumanns, and H. Kitterer, "Evolutionary multi-objective integer programming for the design of adaptive cruise control systems," in *Developments in Applied Artificial Intelligence (IEA/AIE 2002)*, Lecture Notes in Artificial Intelligence Vol. 2358. Springer, 2002.
20. E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, edited by K. Giannakoglou et al., International Center for Numerical Methods in Engineering (CIMNE), 2002, pp. 95–100.
21. G. Rudolph, "An evolutionary algorithm for integer programming," in *Parallel Problem Solving from Nature (PPSN III)*, edited by Y. Davidor, H.-P. Schwefel, and R. Männer, Springer, 1994, pp. 139–148.
22. E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.
23. M. Emmerich, M. Grötzner, B. Gross, and M. Schütz, "Mixed-integer evolution strategy for chemical plant optimization with simulators," in *Evolutionary Design and Manufacutre—Selected papers from ACDM'00*, edited by I.C. Parmee, Springer, 2000, pp. 55–67.
24. E. Zitzler, L. Thiele, M. Laumanns, C.M. Foneseca, and V.G. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.
25. M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, "Combining convergence and diversity in evolutionary multiobjective optimization," *Evolutionary Computation*, vol. 10, no. 3, pp. 263–282, 2002.