Evolutionary Multi-objective Integer Programming for the Design of Adaptive Cruise Control Systems

Nando Laumanns[1], Marco Laumanns[2], and Hartmut Kitterer[3]

[1] RWTH Aachen, Institut für Kraftfahrwesen (ika), D-52704 Aachen, Germany,
laumanns@ika.rwth-aachen.de
[2] ETH Zürich, Computer Engineering and Networks Laboratory (TIK),
CH–8092 Zürich, Switzerland,
laumanns@tik.ee.ethz.ch,
http://www.tik.ee.ethz.ch/aroma
[3] WABCO Vehicle Control Systems, D-30453 Hannover, Germany,
hartmut.kitterer@wabco-auto.com

Abstract. Adaptive Cruise Control (ACC) systems represent an active research area in the automobile industry. The design of such systems typically involves several, possibly conflicting criteria such as driving safety, comfort and fuel consumption. When the different design objectives cannot be met simultaneously, a number of non-dominated solutions exists, where no single solution is better than another in every aspect. The knowledge of this set is important for any design decision as it contains valuable information about the design problem at hand.
In this paper we approximate the non-dominated set of a given ACC-controller design problem for trucks using multi-objective evolutionary algorithms (MOEAs). Two different search strategies based on a continuous relaxation and on a direct representation of the integer design variables are applied and compared to a grid search method.

1 Introduction

Crowded motorways and a higher average vehicle speed create increasing difficulties for drivers. The automobile industry tries to compensate these additional demands by inventing driver assistance systems such as antilock braking system (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, 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 rather 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, a simulation tool is used to analyze the ACC-system's behavior. This offers the possibility to improve the development process further by applying numerical optimization techniques such as evolutionary algorithms to optimize the ACC-controller.

This paper now examines the design of an ACC-controller for trucks under multiple, possibly conflicting criteria. In the next section the ACC-system is explained along with its simulation environment and the corresponding optimization problem that results from the system's degrees of freedom and the different design criteria. Section 3 addresses multi-objective optimization and describes the algorithms that are used to approximate the set of non-dominated or Pareto-optimal solutions. In section 4 the results of the different optimization runs are presented and discussed.

2 Adaptive Cruise Control (ACC)

The adaptive cruise control system is intended to support the driver with controlling the vehicle's longitudinal dynamics. A radar sensor detects moving objects in front of the car. A tracking algorithm along with a yaw rate sensor determines a relevant target, including its distance and its relative velocity to the ACC-vehicle.

The ACC-controller uses this information to calculate a desired speed and a corresponding acceleration. This is transferred via CAN-bus to the appropriate parts of the car (see Fig. 1). Concerning trucks, ACC-systems usually have influence on the various elements of the braking system and on the engine control. In addition to that, the gear-box can be controlled. The interaction between the system and the driver is accomplished through an intelligent dashboard with switches, displays and a buzzer.

2.1 Design Variables

The longitudinal controller is responsible for the translation of the incoming data about the traffic situation in front of the vehicle and its own driving condition into a desired acceleration. The data produced by the sensor contains some deviation. This requires four different filters in order to create a smooth acceleration signal. The influence of these filters can be regulated by four integer parameters $x_1, \ldots, x_4$. Strong filters result in very smooth signals. However, they somewhat delay the vehicle's reaction to incoming data and that 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.

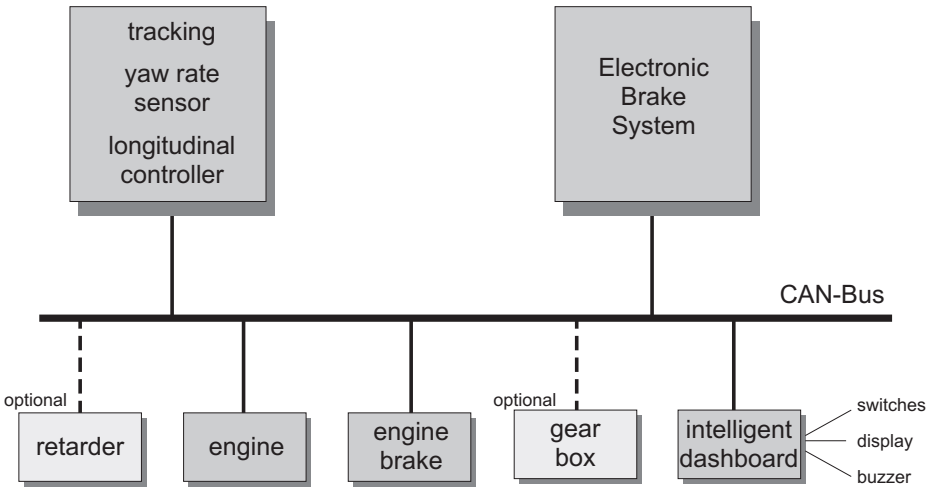

Fig. 1. Structure of the ACC-system.

2.2 Simulation

Each setting of the design variables represents a decision alternative, and the resulting controller performance is determined through simulation. For this we apply the simulation tool PELOPS, which has been developed by the ika in cooperation with BMW [3]. It analyses interactions between vehicle, driver and the environment. Its three main elements are the stretch-module, the decision-module, and the handling-module (see Fig. 2). The cause-effect-principle is used to realize the relation between vehicle performance, environmental impacts and the driver's decision. In this case the ACC-system replaces the driver in terms of regulating the speed.

2.3 Design Objectives

Four objectives are defined to give a sufficient characterization of the ACC-system's longitudinal controlling behavior considering driving comfort, fuel efficiency and safety. All these objective functions are computed within the simulation. Thus, the resulting multi-objective optimization problem can be stated as follows (where the function values of f and g calculated by the simulator).

$$\begin{aligned} \text{Minimize} \quad & f(x) = \begin{pmatrix} f_1(x) \\ f_2(x) \\ f_3(x) \\ f_4(x) \end{pmatrix} && \begin{matrix} \text{(average fuel consumption)} \\ \text{(acceleration / deceleration time)} \\ \text{(velocity deviation)} \\ \text{(acceleration deviation)} \end{matrix} \\ \text{subject to} \quad & g(x) \geq d_{min} && \text{(minimum follow-up distance)} \\ & x \in X = \{1,2,\ldots,99\}^2 \times \{1,2,\ldots,16\} \times \{1,2,\ldots,8\}^3 \subseteq \mathbb{Z}^6 \end{aligned} \tag{1}$$

For safety reasons there is a constraint on the minimum distance $g(x)$ between the ACC-vehicle and the leading vehicle.

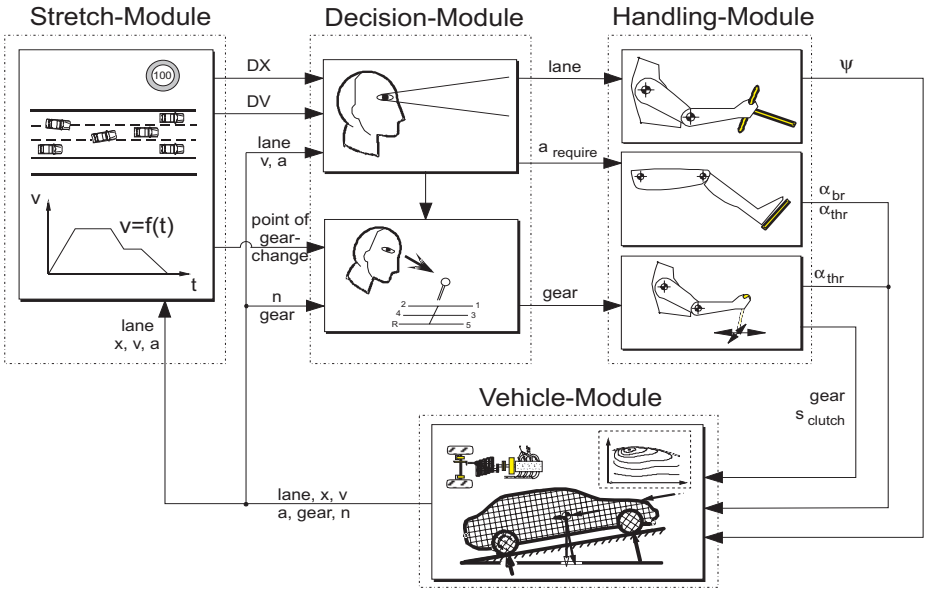

Fig. 2. Elements of simulation environment PELOPS.

3 Multi-objective Optimization

The presence of multiple objectives in a design problem induces a partial order on the set of alternatives. This partial order is represented by the dominance relation $\prec$, where a decision alternative is said to dominate another ($a \prec b$) if and only if it is at least as good in all objectives and strictly better in at least one objective. The solution of the constrained multi-objective integer programming problem (1) is defined as the set of feasible non-dominated decision alternatives (Pareto set) $X^* = \{a \in X : (\nexists b \in X \text{ with } b \prec a) \wedge g(a) \geq d_{min}\}$.

In order to approximate the non-dominated set for the constrained multi-objective integer programming problem (1), three methods are applied and compared: a grid search and two evolutionary algorithms. The computation time of the simulator makes 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 to the non-dominated set as a baseline.

3.1 Multi-objective Evolutionary Algorithms

Evolutionary algorithms (EA) have shown to be a useful tool for approximating the non-dominated set of multi-objective optimization problems [1, 6, 5]. In many engineering design problems the evaluation of decision alternatives is based on

(computationally expensive) simulations which limits the use of traditional techniques like gradient-based methods. When *a priori* incorporation of the decision maker's preferences is difficult or not desired (so that aggregation of the different objectives to a scalar surrogate function is not possible), there are hardly any alternative techniques available.

Algorithm 1 $(\mu + \lambda)$-SPEA2

1: Generate an initial population P of size $\mu + \lambda$.
2: Calculate objective values of individuals in P.
3: **while** Termination criteria are not met **do**
4: Calculate fitness values of individuals in P.
5: Copy all non-dominated individuals in P to P'.
6: **if** $|P'| > \mu$ **then**
7: Reduce P' by means of the truncation procedure.
8: **else if** $|P'| < \mu$ **then**
9: Fill P' with dominated individuals from P in increasing order of their fitness.
10: **end if**
11: Create P'' of size λ by iteratively selecting parents from P', applying recombination and mutation.
12: Calculate objective values of individuals in P''.
13: $P \leftarrow P' + P''$
14: **end while**

Here, two algorithms based on SPEA2 ([7], an improved version of the Strength Pareto Evolutionary Algorithm, [8]) are applied. SPEA2 can be seen as a $(\mu + \lambda)$-EA with special fitness assignment and selection techniques, a pseudocode description is given in Alg. 1. Each individual in the combined parent and offspring population P is assigned a strength value S which equals the number of solutions it dominates. On the basis of the S values, the raw fitness $R(\boldsymbol{i})$ of an individual $\boldsymbol{i}$ is calculated $R(\boldsymbol{i}) = \sum_{\boldsymbol{j} \in \boldsymbol{P}, \boldsymbol{j} \prec \boldsymbol{i}} S(\boldsymbol{j})$. In addition, the local density is estimated for each individual and added to R to discriminate between individuals with equal raw fitness (line 4).

Selection is performed in two steps: environmental selection and mating selection. The best μ solutions out of P are selected to constitute the set of parents for the next generation. First, all non-dominated individuals, i.e., those which have a fitness lower than one, are selected (line 5). If there are more than μ such solutions, a truncation procedure is invoked which iteratively removes that individual which is closest to the others (line 7). If less than μ individuals are non-dominated, the space is filled with the dominated individuals in ascending order of their fitness values (line 9). In the second step the recombination partners for the next generation are selected by binary tournament selection (with replacement) based on the fitness values (line 11).

The representation of the individuals and the variation operators (recombination and mutation) are different and explained in the following.

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 [1] with distribution index $\eta = 5$. The offspring individuals are then mutated by adding normal distributed random numbers, where the standard deviation σ is set to 5 per cent of the interval length.

Integer-valued individuals As the relaxation produces an artificial blow-up of the search space a direct representation of the decision variables as integer numbers might be 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 so common in evolutionary computation. Here, we adopt the techniques from Rudolph [4] who developed an EA for integer programming with maximum entropy mutation distributions that enables 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 [2]. Here, the initial mutation step size was set to $s = 2$ for all variables.

For both version of SPEA2 the population size was set to $\mu = \lambda = 10$, and the runs were terminated after 300 generations (3000 objective function evaluations). During the run, an archive of all non-dominated solutions was maintained and output as the approximation to the non-dominated set at the end of the run.

4 Results

In order to evaluate the performance of the evolutionary algorithm a grid search over the whole parameter area 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. As both instances of the evolutionary algorithm only used 3000 function evaluations each, and 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 values close to the minimal value of 66.6 for almost all non-dominated alternatives. The remaining objective values of the different non-dominated sets are displayed in Fig. 3. Here, the trade-off characteristic is visible from the three-dimensional scatter plot.

Measuring the performance of different multi-objective optimizers or the quality of different non-dominated sets is not straightforward. Many performance metrics have been suggested [1] and can be applied, but there is no clear order unless one set completely dominates another.

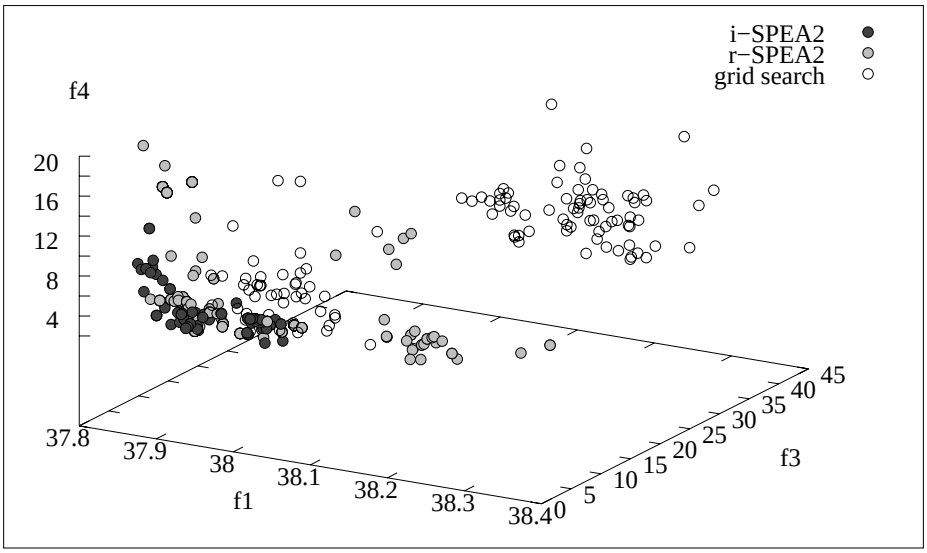

Fig. 3. Scatter plot of the non-dominated solutions produced by the grid search and the evolutionary algorithm with continuous relaxation (r-SPEA2) and direct integer coding (i-SPEA) for the objective function values f_1, f_3, f_4.

To ensure comparability of the results one has to look at different aspects. One possibility is to define an evaluation formula based on a weighted distance to an ideal point f^* which is given by the minimal objective values in each dimension. The difference between each decision parameter and the optimum value in the referring category is multiplied with a factor, which represents the importance of the category. Thus the interpretation of the results reflects an adaptation to the user's goals. In this case the objectives f_1 and f_3 are considered most important, f_2 is least important. Representing the distance to the optimal solution the sum of those values gives the overall quality of the individual

$$D(x) = 150(f_1(x) - f_1^*) + (f_2(x) - f_2^*) + 6(f_3(x) - f_3^*) + 4(f_4(x) - f_4^*) \quad (2)$$

with $f^* = (37.8339, 66.6, 2.06935, 3.03196)$. Accordingly, a ranking of the individuals developed by the different optimization strategies can be produced. The best 100 solutions are displayed in Fig. 4. The two evolutionary algorithms create the best solutions, with a slight advantage of the integer-version in terms of density close to the optimum solution. Out of the top 100 solutions, 46 are were created by this integer-version, 40 by the double-version and only 14 by the grid-search.

Another aspect which has to be taken into consideration in order to compare the different optimization approaches is the total objective space that their solutions extend to. The minimum overall objective function value divided by the minimum value of a single approach determines the quality of the optimization in direction of the corresponding objective. Fig. 5 visualizes the performance of

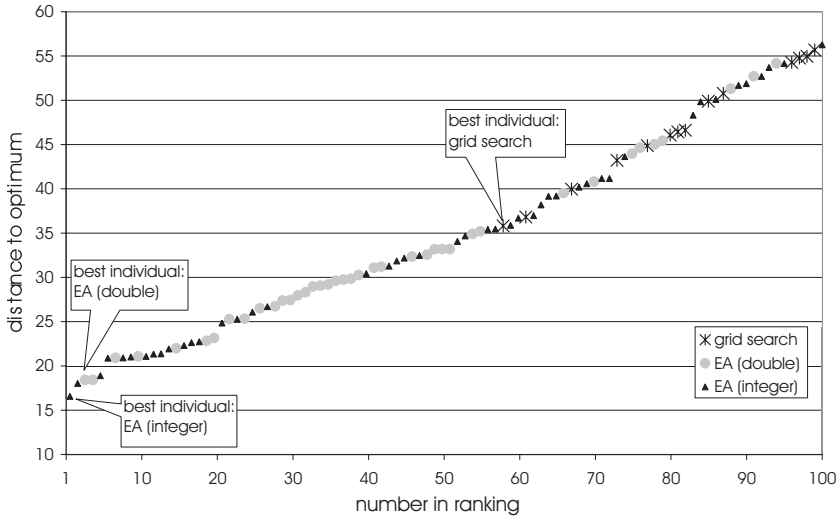

Fig. 4. Ranking of solutions according to aggregated quality measure.

the different optimization approaches in terms of objective space exploration. While all three approaches reach the optimum in the objectives f_1 and f_2, the grid search shows a performance almost 10% below the optimum in f_3 and 5% in f_4. The integer-version of the EA proves to be the best optimization method with a good performance in all four objectives and an average value of 99.62%.

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

Table 1. Results of the coverage measure $\mathcal{C}(\cdot,\cdot)$ applied to all pairs of algorithms.

For a pairwise comparison of different output sets, the *coverage* measure [8] gives an indication of how much of a one algorithm's output has also been reached by the other one. Specifically, $\mathcal{C}(\mathcal{A},\mathcal{B})$ calculates the relative number of points of set B that are dominated by at least one point in set A. Table 1 shows that none of the points found by the grid search is better than any point in the non-dominated sets of the evolutionary algorithms. Also, the SPEA2 working with the floating point representation does not cover much (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.

In contrast to a relative comparison by the coverage measure, the normalized volume of the dominated space is often used to evaluate a single non-dominated

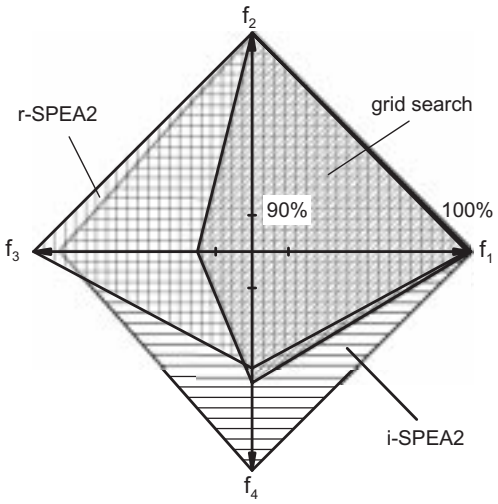

Fig. 5. This graph shows how well each algorithm could approach the minimal value in each objective dimension separately. The values on each axis is calculated by dividing the overall minimal value by the minimal value produced by each algorithm.

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

Table 2. Results of the hypervolume difference measure $\mathcal{S}(\cdot,\cdot)$ applied to all pairs of algorithms and the absolute hypervolume measure $\mathcal{S}(\cdot)$ (last column).

set alone [8]. Here, a reference cuboid between the ideal point f^* and the nadir point defined by the maximal objective function values of the union of all three non-dominated sets is chosen. $\mathcal{S}(\mathcal{A})$ gives the fraction of this reference volume that is dominated by A. It is clear that the algorithms ideally should maximize the dominated space. The results are given in the last column of Table 2. In addition, the volume differences are depicted, where $\mathcal{S}(\mathcal{A},\mathcal{B})$ evaluates to the volume dominated by the set A, but not dominated by the set B.

5 Conclusion

The combination of PELOPS as simulation tool and an evolutionary algorithm is an efficient method to optimize the longitudinal controller of an ACC-system. Even the non-adapted version of the EA shows a better performance than a grid search over the design variable area, in spite of its lower computation time.

Nevertheless it is sensible to adapt the EA to the simulation tool. Matching parameter intervals and types increase the algorithm's performance, because

they ensure that every change in the design variables, created by the EA, also changes the ACC-system's longitudinal controlling behavior within the simulation tool. This correlation is necessary for an efficient optimization process, and less redundancy is produced compared to the artificial relaxation to real variables. The superiority of results produced by the direct integer encoding shows that efficient search operators exist for this domain, and that they are applicable as well in a multi-objective environment.

Analyzing the trade-off-graph (Fig.3) the development engineer can learn about performance boundaries of the ACC-system. It helps to understand the system's behavior and shows different solutions, which can suit the company's or customer's preferences. Thus the EA can enhance the traditional developing process by quickly providing broad and detailed information about a complex optimization problem.

Acknowledgment

The second author acknowledges the support by the Swiss National Science Foundation (SNF) under the ArOMA project 2100-057156.99/1.

References

1. K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, Chichester, UK, 2001.
2. M. Emmerich, M. Grötzner, B. Groß, and M. Schütz. Mixed-integer evolution strategy for chemical plant optimization with simulators. In I. C. Parmee, editor, *Evolutionary Design and Manufacutre — Selected papers from ACDM'00*, pages 55–67, Plymouth, UK, April 26–28, 2000. Springer, London.
3. J. Ludmann. *Beeinflussung des Verkehrsablaufs auf Strassen: Analyse mit dem fahrzeugorientierten Verkehrssimulationsprogramm PELOPS*. Schriftenreihe Automobiltechnik. Forschungsgesellschaft Kraftfahrwesen mbH Aachen, 1998. (in German).
4. G. Rudolph. An evolutionary algorithm for integer programming. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature – PPSN III, Int'l Conf. Evolutionary Computation*, pages 139–148, Jerusalem, October 9–14, 1994. Springer, Berlin.
5. P. Sen and J.-B. Yang. *Multipe Criteria Decision Support in Engineering Design*. Springer, 1998.
6. E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello, and D. Corne, editors. *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, volume 1993 of *Lecture Notes in Computer Science*, Berlin, Germany, March 2001. Springer.
7. E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In K. Giannakoglou et al., editors, *Evolutionary Methods for Design, Optimisation, and Control*, 2002. To appear.
8. E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.