Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach

Eckart Zitzler and Lothar Thiele

***Abstract*—Evolutionary algorithms (EA's) are often well-suited for optimization problems involving several, often conflicting objectives. Since 1985, various evolutionary approaches to multiobjective optimization have been developed that are capable of searching for multiple solutions concurrently in a single run. However, the few comparative studies of different methods presented up to now remain mostly qualitative and are often restricted to a few approaches. In this paper, four multiobjective EA's are compared quantitatively where an extended 0/1 knapsack problem is taken as a basis. Furthermore, we introduce a new evolutionary approach to multicriteria optimization, the Strength Pareto EA (SPEA), that combines several features of previous multiobjective EA's in a unique manner. It is characterized by a) storing nondominated solutions externally in a second, continuously updated population, b) evaluating an individual's fitness dependent on the number of external nondominated points that dominate it, c) preserving population diversity using the Pareto dominance relationship, and d) incorporating a clustering procedure in order to reduce the nondominated set without destroying its characteristics. The proof-of-principle results obtained on two artificial problems as well as a larger problem, the synthesis of a digital hardware–software multiprocessor system, suggest that SPEA can be very effective in sampling from along the entire Pareto-optimal front and distributing the generated solutions over the tradeoff surface. Moreover, SPEA clearly outperforms the other four multiobjective EA's on the 0/1 knapsack problem.**

***Index Terms*— Clustering, evolutionary algorithm, knapsack problem, multiobjective optimization, niching, Pareto optimality.**

I. Introduction

MANY real-world problems involve simultaneous optimization of several incommensurable and often competing objectives. Often, there is no single optimal solution, but rather a set of alternative solutions. These solutions are optimal in the wider sense that no other solutions in the search space are superior to them when *all* objectives are considered. They are known as *Pareto-optimal* solutions.

Consider, for example, the design of a complex hardware/software system. An optimal design might be an architecture that minimizes cost and power consumption while maximizing the overall performance. However, these goals are generally conflicting: one architecture may achieve high performance at high cost, an alternative low-cost architecture might considerably increase power consumption—none of these solutions can be said to be superior if we do not include preference information (e.g., a ranking of the objectives). Thus if no such information is available, it may be useful to have knowledge about those alternative architectures. A tool exploring the design space for Pareto-optimal solutions in reasonable time can essentially aid the decision maker in arriving at a final design.

Evolutionary algorithms (EA's) seem to be particularly suited for this task because they process a set of solutions in parallel, possibly exploiting similarities of solutions by recombination. Some researchers suggest that multiobjective search and optimization might be a problem area where EA's do better than other blind search strategies [1], [2]. Although this statement must be qualified with regard to the "no free lunch" theorems [3], up to now there are few if any alternatives to EA-based multiobjective optimization [4].

Since the mid-1980's, there has been a growing interest in solving multicriteria optimization problems using evolutionary approaches. In the meantime, several multiobjective EA's are available that are capable of searching for multiple Pareto-optimal solutions concurrently in a single run. They differ mainly in the fitness assignment, but the question of which of these methods is better on what type of problem is mostly unsettled. The few comparative studies that have been published up to now remain mostly qualitative and are often restricted to a few algorithms. Therefore, extensive quantitative comparisons are needed in order to assess the performance of the EA's in a greater context. Previous effort in this direction has been reported in [5].

In the present study, we provide a comparison of five multicriteria EA's, four previously known and one new, by solving a multiobjective 0/1 knapsack problem. Thereby, two complementary quantitative measures are considered in order to assess the performance of the algorithms concerning the tradeoff surfaces produced. A random-search strategy as well as a single-objective EA serve as additional points of reference. The Strength Pareto Evolutionary Algorithm (SPEA), the new multiobjective approach proposed in this paper, has been developed on the basis of a comparative study previously carried out [5]; it integrates established techniques used in existing EA's in a single unique algorithm. We show that SPEA can have advantages over the other algorithms under consideration in convergence to the Pareto-optimal front.

Manuscript received July 24, 1998; revised March 18, 1999. This work was supported by the Swiss National Science Foundation.

The authors are with the Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zürich, Switzerland (e-mail: zitzler@tik.ee.ethz.ch; thiele@tik.ee.ethz.ch).

Publisher Item Identifier S 1089-778X(99)08067-4.

The paper is organized as follows. Section II introduces key concepts used in the field of evolutionary multicriteria optimization and gives an overview of the multiobjective EA's considered in this investigation. The comparison of the four multiobjective EA's on the 0/1 knapsack problem is the subject of Section III, which itself is divided into three parts: description of the test problem, methodology of the comparison, and experimental results. Section IV is devoted to SPEA and describes both underlying principles and the application to three problems (Schaffer's f_2, knapsack problem, and system-level synthesis). The last section offers concluding remarks and future perspectives.

II. Multiobjective Optimization Using Evolutionary Algorithms

A. Definitions

A general multiobjective optimization problem can be described as a vector function f that maps a tuple of m parameters (decision variables) to a tuple of n objectives. Formally:

$$\begin{aligned} &\min/\max \boldsymbol{y} = f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})) \\ &\text{subject to } \boldsymbol{x} = (x_1, x_2, \ldots, x_m) \in X \\ &\qquad\qquad \boldsymbol{y} = (y_1, y_2, \ldots, y_n) \in Y \end{aligned} \tag{1}$$

where $\boldsymbol{x}$ is called the *decision vector*, X is the *parameter space*, $\boldsymbol{y}$ is the *objective vector*, and Y is the *objective space*.[1]

The set of solutions of a multiobjective optimization problem consists of all decision vectors for which the corresponding objective vectors cannot be improved in any dimension without degradation in another—these vectors are known as *Pareto optimal*. Mathematically, the concept of Pareto optimality is as follows: Assume, without loss of generality, a maximization problem and consider two decision vectors $a, b \in X$. Then, a is said to *dominate* b (also written as $\boldsymbol{a} \succ \boldsymbol{b}$) iff

$$\begin{aligned} &\forall i \in \{1, 2, \ldots, n\} : f_i(\boldsymbol{a}) \geq f_i(\boldsymbol{b}) \quad \wedge \\ &\exists j \in \{1, 2, \ldots, n\} : f_j(\boldsymbol{a}) > f_j(\boldsymbol{b}). \end{aligned} \tag{2}$$

Additionally, in this study $\boldsymbol{a}$ is said to *cover* $\boldsymbol{b}$ ($\boldsymbol{a} \succeq \boldsymbol{b}$) iff $\boldsymbol{a} \succ \boldsymbol{b}$ or $f(\boldsymbol{a}) = f(\boldsymbol{b})$. All decision vectors which are not dominated by any other decision vector of a given set are called *nondominated* regarding this set. If it is clear from the context which set is meant, we simply leave it out. The decision vectors that are nondominated within the entire search space are denoted as *Pareto optimal* and constitute the so-called *Pareto-optimal set* or *Pareto-optimal front*.

B. Fitness Assignment Strategies

In their excellent review of evolutionary approaches to multiobjective optimization, Fonseca and Fleming [1] categorize several multicriteria EA's and compare different fitness assignment strategies. In particular, they distinguish plain aggregating approaches, population-based non-Pareto approaches, and Pareto-based approaches.

Aggregation methods combine the objectives into a higher scalar function that is used for fitness calculation. Scalarization is mandatory when applying an EA, but aggregation approaches have the advantage of producing one single solution. On the other hand, defining the goal function in this way requires profound domain knowledge that is often not available. Popular aggregation methods are the weighted-sum approach, target vector optimization, and the method of goal attainment [1], [6]. Nevertheless, *pure* aggregation methods are not considered here because they are not designed for finding a family of solutions.

Population-based non-Pareto approaches, however, are able to evolve multiple nondominated solutions concurrently in a single simulation run. By changing the selection criterion during the reproduction phase, the search is guided in several directions at the same time. Often, fractions of the mating pool are selected according to one of the n objectives [9], [10]. Other non-Pareto algorithms use multiple linear combinations of the objectives in parallel [11], [12].

Pareto-based fitness assignment was first proposed in [13]. All approaches of this type explicitly use Pareto dominance in order to determine the reproduction probability of each individual. While non-Pareto EA's are often sensitive to the nonconvexity of Pareto-optimal sets, this is not the case for Pareto-based EA's [1].

Finally, some multiobjective EA's also make use of combinations of the presented fitness assignment strategies (e.g., [14], [15]).

C. Multimodal Optimization and Preservation of Diversity

When we consider the case of finding a *set* of nondominated solutions rather than a single-point solution, multiobjective EA's have to perform a multimodal search that samples the Pareto-optimal set uniformly. Unfortunately, a simple (elitist) EA tends to converge toward a single solution and often loses solutions due to three effects [16]: selection pressure, selection noise, and operator disruption. To overcome this problem, several methods have been developed that can be divided into niching techniques and nonniching techniques [16]. Both types aim at preserving diversity in the population (and therefore try to prevent from premature convergence), but in addition niching techniques are characterized by their capability of promoting the formulation and maintenance of stable subpopulations (*niches*).

Fitness sharing [17] is used most frequently, which is a niching technique based on the idea that individuals in a particular niche have to share the available resources. The more individuals are located in the neighborhood of a certain individual, the more its fitness value is degraded. The neighborhood is defined in terms of a distance measure $d(i, j)$ and specified by the so-called *niche radius* σ_{share}. Depending on whether the distance function $d(i, j)$ operates on the genotypes or the phenotypes, one distinguishes between *genotypic*

[1] The definitions and terms presented in this section correspond to the mathematical formulations most widespread in multiobjective EA literature (e.g., [6], [1]). For more detailed information, we refer to [7] and [8].

sharing and *phenotypic sharing*; phenotypic sharing can be performed on the decision vectors or the objective vectors. Currently, most multiobjective EA's implement fitness sharing (e.g., [11], [14], [18], [6], [15], [19], [20]).

Among the nonniching techniques, *restricted mating* is the most common in multicriteria function optimization. Basically, two individuals are allowed to mate only if they are within a certain distance (given by the parameter σ_{mate}) to each other. This mechanism may avoid the formation of lethal individuals and therefore improve the online performance. Nevertheless, as mentioned in [1], it does not appear to be widespread in the field of multiobjective EA's (e.g., [11], [14], [21]).

To our knowledge, other niching methods like *crowding* [22] and its derivatives as well as nonniching techniques as *isolation by distance* [23] have never been applied to EA's with multiple objectives (an exception is offered in [24], cf. Section IV-D).

D. Four Population-Based Approaches

In the following we present the multiobjective EA's applied to the knapsack problem in our comparison. For a thorough discussion of other evolutionary approaches, we refer to [1], [25], and [4].

1) Vector Evaluated Genetic Algorithm: Schaffer [9] presented a multimodal EA called vector evaluated genetic algorithm (VEGA) that carries out selection for each objective separately. In detail, the mating pool is divided into n parts of equal size; part i is filled with individuals that are chosen at random from the current population according to objective i. Afterwards, the mating pool is shuffled and crossover and mutation are performed as usual. Schaffer implemented this method in combination with fitness proportionate selection.

Although some serious drawbacks are known, this algorithm has been a strong point of reference up to now. Therefore, it was included in this investigation.

2) Aggregation by Variable Objective Weighting: Another non-Pareto approach was introduced in [11] (in the following referred to as HLGA—Hajela's and Lin's genetic algorithm), that used the weighted-sum method for fitness assignment. Thereby, each objective is assigned a weight $w_i \in]0, 1[$, such that $\sum w_i = 1$, and the scalar fitness value is calculated by summing up the weighted objective values $w_i \cdot f_i(\boldsymbol{x})$. To search for multiple solutions in parallel, the weights are not fixed but instead encoded in the genotype. The diversity of the weight combinations is promoted by phenotypic fitness sharing. As a consequence, the EA evolves solutions and weight combinations simultaneously. Finally, [11, p. 102] emphasized mating restrictions to be necessary in order to "both speed convergence and impart stability to the genetic search."

Several other multiobjective EA's make use of weighted-sum aggregation (e.g., [12]). We have chosen HLGA to represent this class of multiobjective EA's.

3) Niched Pareto Genetic Algorithm: The niched Pareto genetic algorithm (NPGA) proposed in [18] and [26] combines tournament selection and the concept of Pareto dominance. Two competing individuals and a comparison set of other individuals are picked at random from the population; the size of the comparison set is given by the parameter t_{dom}. If one of the competing individuals is dominated by any member of the set and the other is not, then the latter is chosen as winner of the tournament. If *both* individuals are dominated (or not dominated), the result of the tournament is decided by sharing: The individual that has the least individuals in its niche (defined by σ_{share}) is selected for reproduction. Horn and Nafpliotis [18], [26] used phenotypic sharing on the objective vectors.

This algorithm seems to be widespread and has been often taken as reference in recent publications [2], [21], [20], hence, it is also examined here.

4) Nondominated Sorting Genetic Algorithm: Srinivas and Deb [6] also developed an approach based on [13], called nondominated sorting genetic algorithm (NSGA). Analogously to [13], the fitness assignment is carried out in several steps. In each, the nondominated solutions constituting a nondominated front are assigned the same dummy fitness value. These solutions are shared with their dummy fitness values (phenotypic sharing on the decision vectors) and ignored in the further classification process. Finally, the dummy fitness is set to a value less than the smallest shared fitness value in the current nondominated front. Then the next front is extracted. This procedure is repeated until all individuals in the population are classified. In the original study [6], this fitness assignment method was combined with a stochastic remainder selection.

We have selected NSGA as the second Pareto-based EA, although there are also other Pareto-based approaches that may be under consideration for the comparison, e.g., the multiobjective EA presented in [14].

III. Performance Comparison

In the following, the case study is described that has been carried out using the above four multiobjective EA's for solving an extended 0/1 knapsack problem. The comparison focuses on the effectiveness in finding multiple Pareto-optimal solutions, disregarding their number. Nevertheless, in the case that the tradeoff surface is continuous or contains many points, the distribution of the nondominated solutions achieved is also important. Although we do not consider the distribution explicitly, it influences the performance of the EA indirectly.

A. The Multiobjective 0/1 Knapsack Problem

A test problem for a comparative investigation like this has to be chosen carefully. The problem should be understandable and easy to formulate so that the experiments are repeatable and verifiable. It should also be a rather general problem and ideally represent a certain class of real-world problems. Both conditions apply to the knapsack problem: the problem description is simple, yet the problem itself is difficult to solve (NP-hard). Moreover, due to its practical relevance it has been subject to several investigations in various fields. In particular, there are some publications in the domain of evolutionary computation related to the knapsack problem [27]–[29], even in conjunction with multiobjective optimization [30].

1) Formulation as Multiobjective Optimization Problem: Generally, a 0/1 knapsack problem consists of a set of items, weight and profit associated with each item, and an upper bound for the capacity of the knapsack. The task is to find a subset of items which maximizes the total of the profits in the subset, yet all selected items fit into the knapsack, i.e., the total weight does not exceed the given capacity [31].

This single-objective problem can be extended directly to the multiobjective case by allowing an arbitrary number of knapsacks. Formally, the multiobjective 0/1 knapsack problem considered here is defined in the following way: Given a set of m items and a set of n knapsacks, with

$$\begin{aligned} p_{i,j} &= \text{profit of item } j \text{ according to knapsack } i \\ w_{i,j} &= \text{weight of item } j \text{ according to knapsack } i \\ c_i &= \text{capacity of knapsack } i \end{aligned}$$

find a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_m) \in \{0,1\}^m$, such that

$$\forall i \in \{1, 2, \ldots, n\} : \sum_{j=1}^{m} w_{i,j} \cdot x_j \leq c_i \tag{3}$$

and for which $f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}))$ is maximum, where

$$f_i(\boldsymbol{x}) = \sum_{j=1}^{m} p_{i,j} \cdot x_j \tag{4}$$

and $x_j = 1$ iff item j is selected.

2) Test Data: In order to obtain reliable and sound results, we used nine different test problems where both the number of knapsacks and the number of items were varied.[2] Two, three, and four objectives were taken under consideration, in combination with 250, 500, and 750 items.

Following suggestions in [31], *uncorrelated* profits and weights were chosen, where $p_{i,j}$ and $w_{i,j}$ are random integers in the interval $[10, 100]$. The knapsack capacities were set to half the total weight regarding the corresponding knapsack

$$c_i = 0.5 \sum_{j=1}^{m} w_{i,j}. \tag{5}$$

As reported in [31], about half of the items are expected to be in the optimal solution (of the single-objective problem) when this type of knapsack capacity is used. We also examined more restrictive capacities ($c_i = 200$) where the solutions contain only a few items. As this had no significant influence on the relative performance of the EA's, we only present the results concerning the former type of knapsack capacity in the following.

3) Implementation: Concerning the chromosome coding as well as the constraint handling, we drew upon results published in [28], which examined EA's with different representation mappings and constraint handling techniques on the (single-objective) 0/1 knapsack problem. Concluding from the experiments in [28], penalty functions achieve best results on data sets with capacities of half the total weight; however, they fail on problems with more restrictive capacities. Since the experiments should be performed on both kinds of knapsack capacities, we decided to implement a greedy repair method that produced the best outcomes among all algorithms under consideration when both capacity types are regarded. This method is based on a vector representation and repairs infeasible solutions according to a predefined scheme. We adopted this approach with a slightly modified repair mechanism.

In particular, a binary string $\boldsymbol{s}$ of length m is used to encode the solution $\boldsymbol{x} \in \{0,1\}^m$. Since many codings lead to infeasible solutions, a simple repair method r is applied to the genotype $\boldsymbol{s}$: $\boldsymbol{x} = r(\boldsymbol{s})$. The repair algorithm removes items from the solution coded by $\boldsymbol{s}$ step by step until all capacity constraints are fulfilled. The order in which the items are deleted is determined by the maximum profit/weight ratio per item; for item j the maximum profit/weight ratio q_j is given by the equation[3]

$$q_j = \max_{i=1}^{n} \left\{ \frac{p_{i,j}}{w_{i,j}} \right\}. \tag{6}$$

The items are considered in increasing order of the q_j, i.e., those achieving the lowest profit per weight unit are removed first. This mechanism intends to fulfill the capacity constraints while diminishing the overall profit as little as possible.

B. Methodology

In the context of this comparison, several questions arise: What quantitative measures should be used to express the quality of the results so that the EA's can be compared in a meaningful way? What is the outcome of a multiobjective EA regarding a set of runs? How can side effects caused by different selection schemes or mating restrictions be precluded, such that the comparison is not falsified? How can the parameters of the EA, particularly the niche radius, be set appropriately? In the following, we treat these problems.

1) Performance Measures: Two complementary measures were used to evaluate the tradeoff fronts produced by the various EA's.

Size of the space covered: Let $X' = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k) \subseteq X$ be a set of k decision vectors. The function $\mathcal{S}(\mathcal{X}')$ gives the volume enclosed by the union of the polytopes $p_1, p_2, \ldots, p_k$, where each p_i is formed by the intersections of the following hyperplanes arising out of $\boldsymbol{x}_i$, along with the axes: for each axis in the objective space, there exists a hyperplane perpendicular to the axis and passing through the point $(f_1(\boldsymbol{x}_i), f_2(\boldsymbol{x}_i), \ldots, f_n(\boldsymbol{x}_i))$. In the two-dimensional (2-D) case, each p_i represents a rectangle defined by the points $(0,0)$ and $(f_1(\boldsymbol{x}_i), f_2(\boldsymbol{x}_i))$.

Coverage of two sets: Let $X', X'' \subseteq X$ be two sets of decision vectors. The function $\mathcal{C}$ maps the ordered pair (X', X'') to the interval $[0, 1]$

$$\mathcal{C}(X', X'') := \frac{|\{\boldsymbol{a}'' \in X''; \exists\, \boldsymbol{a}' \in X' : \boldsymbol{a}' \succeq \boldsymbol{a}''\}|}{|X''|}. \tag{7}$$

The value $\mathcal{C}(X', X'') = 1$ means that all points in X'' are dominated by or equal to points in X'. The opposite,

[2] The test data sets are available from the authors.

[3] This is a straightforward extension to the single-objective approach presented in [28] where $q_j = p_{1,j}/w_{1,j}$.

$\mathcal{C}(X', X'') = 0$, represents the situation when none of the points in X'' are covered by the set X'. Note that both $\mathcal{C}(X', X'')$ and $\mathcal{C}(X'', X')$ have to be considered, since $\mathcal{C}(X', X'')$ is not necessarily equal to $\mathcal{C}(X'', X')$ (e.g., if X' dominates X'' then $\mathcal{C}(X', X'') = 1$ and $\mathcal{C}(X'', X') = 0$).

The first measure $\mathcal{S}$ has the advantage that each EA can be evaluated independently of the other EA's; however, convex regions may be preferred to concave regions, possibly overrating certain solutions. The second measure $\mathcal{C}$ overcomes this drawback and can be used to show that the outcomes of one algorithm dominate the outcomes of another algorithm, although it does not tell how much better it is.

Since in this comparison the focus is on finding the Pareto-optimal set rather than obtaining a uniform distribution along the tradeoff surface, we did not consider the online performance of the EA's but rather the offline performance. Thus the nondominated set regarding *all* individuals generated over all generations was taken as the output of an optimization run. Altogether 30 independent runs were performed per EA and test problem in order to restrict the influence of random effects. Another randomly created initial population was taken each time, and for each test problem all EA's operated on the same 30 initial populations.

2) Selection and Mating Restrictions: Actually, each multiobjective EA should be combined with the selection scheme originally applied. But the influence of the selection scheme on the outcome of an EA cannot be neglected, e.g., fitness proportionate selection, which is used in VEGA, is well known to have serious disadvantages [32]. In order to guarantee a fair comparison, all EA's considered were implemented with the same selection scheme: binary tournament selection with replacement. This selection method turned out to be superior to both stochastic remainder selection (used in [6]) and linear ranking selection on our test problems—that has been confirmed experimentally.

Unfortunately, a conventional combination of fitness sharing and tournament selection may lead to chaotic behavior of the EA [33]. Therefore, both NSGA and HLGA were implemented using a slightly modified version of sharing, called *continuously updated sharing*, which was proposed by the same researchers. Thereby, the partly filled next generation is used to calculate the niche count rather than the current generation. Horn and Nafpliotis [18], [26] introduced this concept in NPGA as well.

Another problem is the influence of mating restrictions. While Hajela and Lin [11] found it necessary to restrict mating, the other EA's under consideration do not explicitly incorporate this concept. We decided not to use mating restrictions in this study, since the effectiveness of the different fitness assignment and niching methods should be compared. In addition, it was experimentally verified that no significant improvement could be observed when running HLGA with mating restrictions.

3) Parameter Settings: On all test problems, 500 generations were simulated per optimization run, the probabilities of crossover (one-point) and mutation were fixed (0.8 and 0.01, respectively). The population size N was chosen to be dependent on the complexity of the test problem, as can be seen in Table I: the more knapsacks and items involved, the greater the value for N. Following the guidelines in [34], the niche radius was calculated based on normalized distance, assuming the formation of ten (15 and 20, respectively) independent niches in the case of two (three and four, respectively) knapsacks. In Table I, σ^*_{share} relates to sharing on the parameter space, which is implemented in NSGA, while σ_{share} stands for the niche radii used by HLGA and NPGA. Finally, the domination pressure t_{dom}, a parameter of NPGA, was determined experimentally. All NPGA simulations were carried out five times, each time using another value for t_{dom} (5, 10, 15, 20, and 25% of the population size). At the end, the parameter value which achieved the best results for the $\mathcal{S}$ measure was chosen per test problem (cf. Table I).

TABLE I
PARAMETERS THAT WERE ADJUSTED TO THE PROBLEM COMPLEXITY: POPULATION SIZE (N), NICHE RADIUS (OBJECTIVE SPACE: σ_{share}, PARAMETER SPACE: σ^*_{share}), AND DOMINATION PRESSURE (t_{dom})

number of knapsacks	parameters	number of items		
		250	500	750
2	N	150	200	250
	σ_{share}	0.4924	0.4943	0.4954
	σ^*_{share}	115	236	357
	t_{dom}	7	10	12
3	N	200	250	300
	σ_{share}	0.4933	0.4946	0.4962
	σ^*_{share}	113	233	354
	t_{dom}	30	25	15
4	N	250	300	350
	σ_{share}	0.4940	0.4950	0.4967
	σ^*_{share}	112	232	352
	t_{dom}	50	75	35

C. Experimental Results

As additional points of reference, two further methods were considered in this comparison: random sampling and multiple independent sampling. The first algorithm (RAND) randomly generates a certain number of individuals per generation, according to the rate of crossover and mutation (though neither crossover, mutation, nor selection are performed). Hence the number of fitness evaluations was the same as for the EA's. The second algorithm is an elitist single-objective EA using weighted-sum aggregation. In contrast to the other algorithms under consideration, 100 independent runs were performed per test problem, each run optimizing toward another randomly chosen linear combination of the objectives. The nondominated solutions among all solutions generated in the 100 runs form the tradeoff front achieved on a particular test problem. Furthermore, two versions of the single-objective EA were investigated: one with 100 generations per linear combination (SO-1) and another one that terminated after 500 generations in every single optimization run (SO-5).

The results concerning the $\mathcal{S}$ measure (size of the space covered) are shown in Fig. 3, the direct comparison of the different algorithms based on the $\mathcal{C}$ measure (coverage) is depicted in Fig. 2. For each algorithm and ordered pair of algorithms, respectively, there is a sample of 30 $\mathcal{S}$, respectively

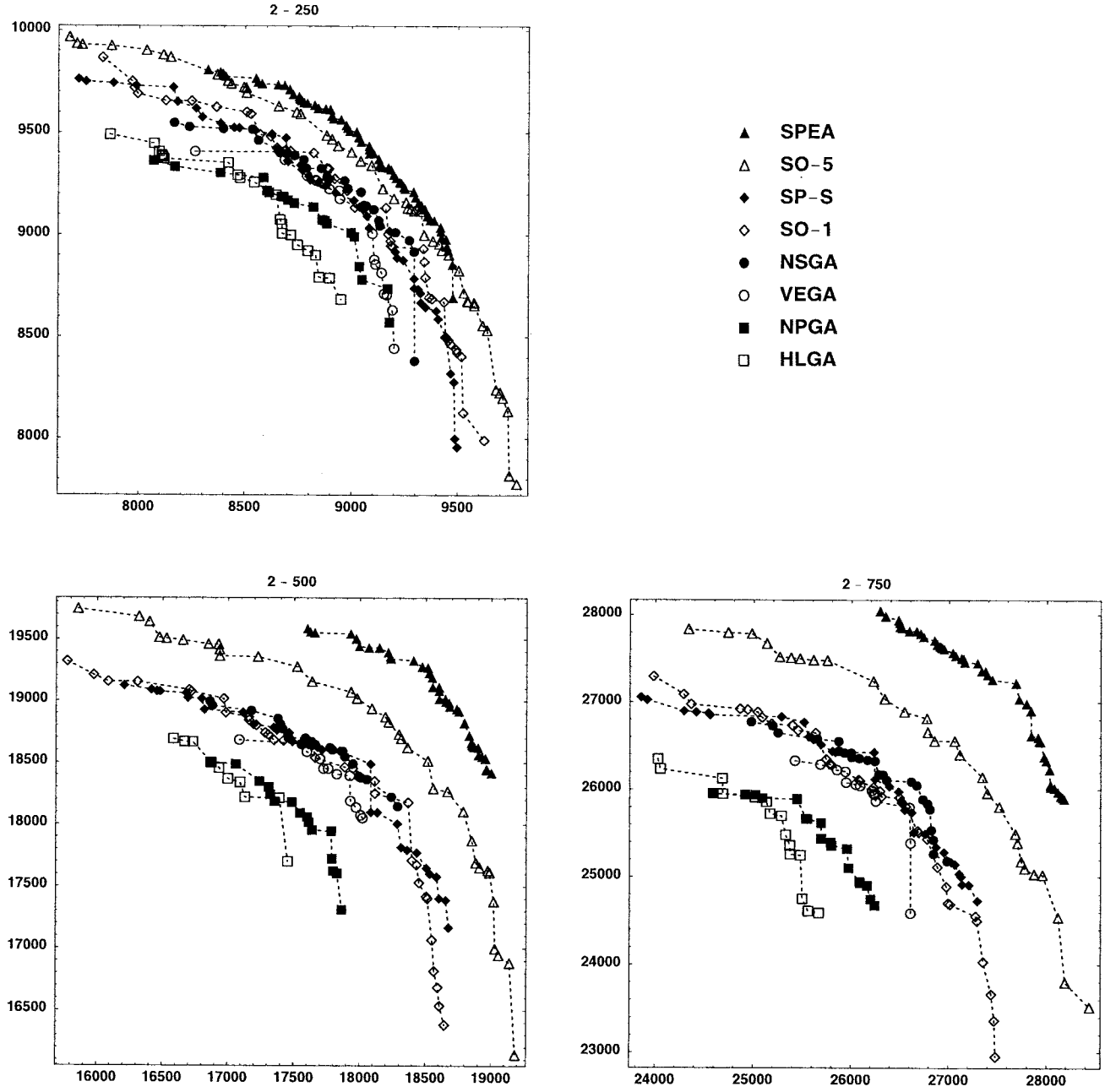


Fig. 1. Tradeoff fronts for two knapsacks: here, the nondominated solutions regarding the first five runs are plotted. For better visualization, the points achieved by a particular method are connected by dashed lines and RAND is not included in the figure. Note that SPEA and SP-S are described later.

$\mathcal{C}$, values per test problem according to the 30 runs performed. Here, *box plots* [35] are used to visualize the distribution of these samples. A box plot consists of a box summarizing 50% of the data. The upper and lower ends of the box are the upper and lower quartiles, while a thick line within the box encodes the median. Dashed appendages summarize the spread and shape of the distribution, and dots represent outside values.

Generally, the simulation results prove that all multiobjective EA's do better than the random search strategy. Fig. 2 shows that the tradeoff fronts achieved by RAND are entirely dominated by the fronts evolved by HLGA, NPGA, and NSGA (with regard to the same population). Concerning the $\mathcal{S}$ distributions, the RAND median is less by more than 20 quartile deviations than the medians associated with the EA's when the maximum quartile deviation of all samples is considered.

Among the multiobjective EA's, NSGA seems to provide the best performance. The median of the $\mathcal{S}$ values is for each test problem greater than the corresponding medians of the other three EA's by more than five quartile deviations. In addition, on eight of the nine test problems NSGA covers more than 70% of the fronts computed by HLGA, NPGA, and VEGA in more than 75% of the runs; in 99% of the runs it covers more than 50%. In contrast, those three EA's cover less than 10% of the NSGA outcomes in 75% of all runs and less than 25% in 99% of the runs (on eight of the nine problems).

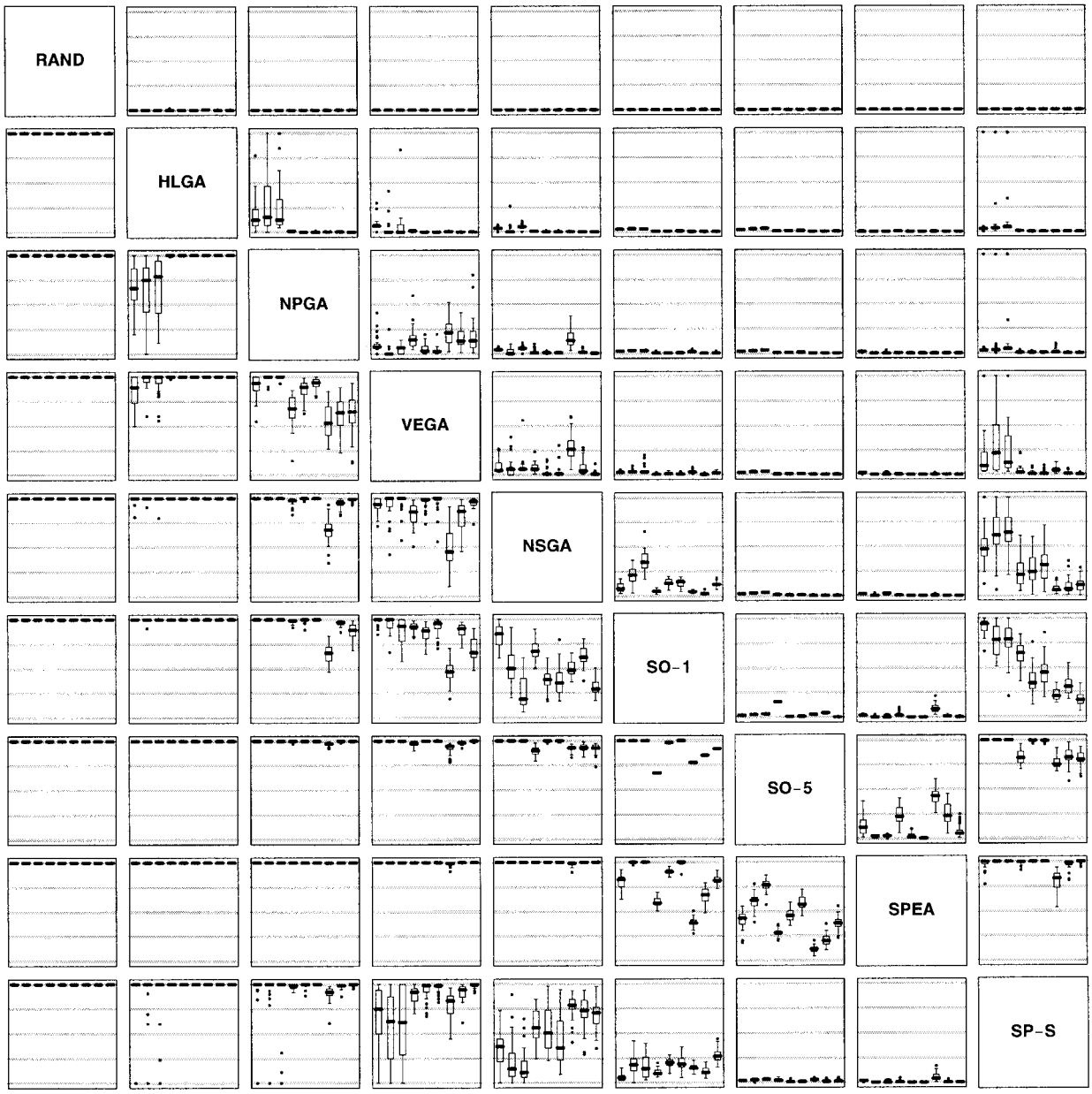


Fig. 2. Box plots based on the $\mathcal{C}$ measure. Each rectangle contains nine box plots representing the distribution of the $\mathcal{C}$ values for a certain ordered pair of algorithms; the three box plots to the left relate to two knapsacks and (from left to right) 250, 500, and 750 items; correspondingly the three middle box plots relate to three knapsacks and the three to the right to four knapsacks. The scale is 0 at the bottom and 1 at the top per rectangle. Furthermore, each rectangle refers to algorithm A associated with the corresponding row and algorithm B associated with the corresponding column and gives the fraction of B covered by A $(\mathcal{C}(A, B))$. Note that SPEA and SP-S are introduced later.

For four knapsacks and 250 items, the coverage rates scatter more, however, NSGA achieves higher $\mathcal{C}$ values in comparison with the other multiobjective EA's.

Comparing NPGA and VEGA, there is no clear evidence that one algorithm outperforms the other, although VEGA seems to be slightly superior to NPGA. Only on two of the test problems (two knapsacks, 500 and 750 items) do the medians of the $\mathcal{S}$ distributions of the two EA's deviate by more than three quartile deviations (in favor of VEGA). In the direct comparison based on the $\mathcal{C}$ measure, VEGA covers more than 50% of the NPGA outcomes on average, while NPGA achieves less than 25% coverage regarding VEGA on average. Furthermore, both algorithms generate better assessments in comparison with HLGA. With three and four knapsacks, the fronts produced by HLGA are dominated by the NPGA and VEGA fronts by 99% (cf. Fig. 2), and the medians of the $\mathcal{S}$ values associated with HLGA are more than ten quartile deviations less than the $\mathcal{S}$ medians related to NPGA and VEGA. For two knapsacks, the $\mathcal{S}$ distributions are closer together; however, the $\mathcal{C}$ measure indicates clear advantages of NPGA and VEGA over HLGA.

Finally, the fact that SO-5 covers on average more than 90% of the nondominated solutions computed by HLGA, NPGA, VEGA, and NSGA and achieves significantly greater

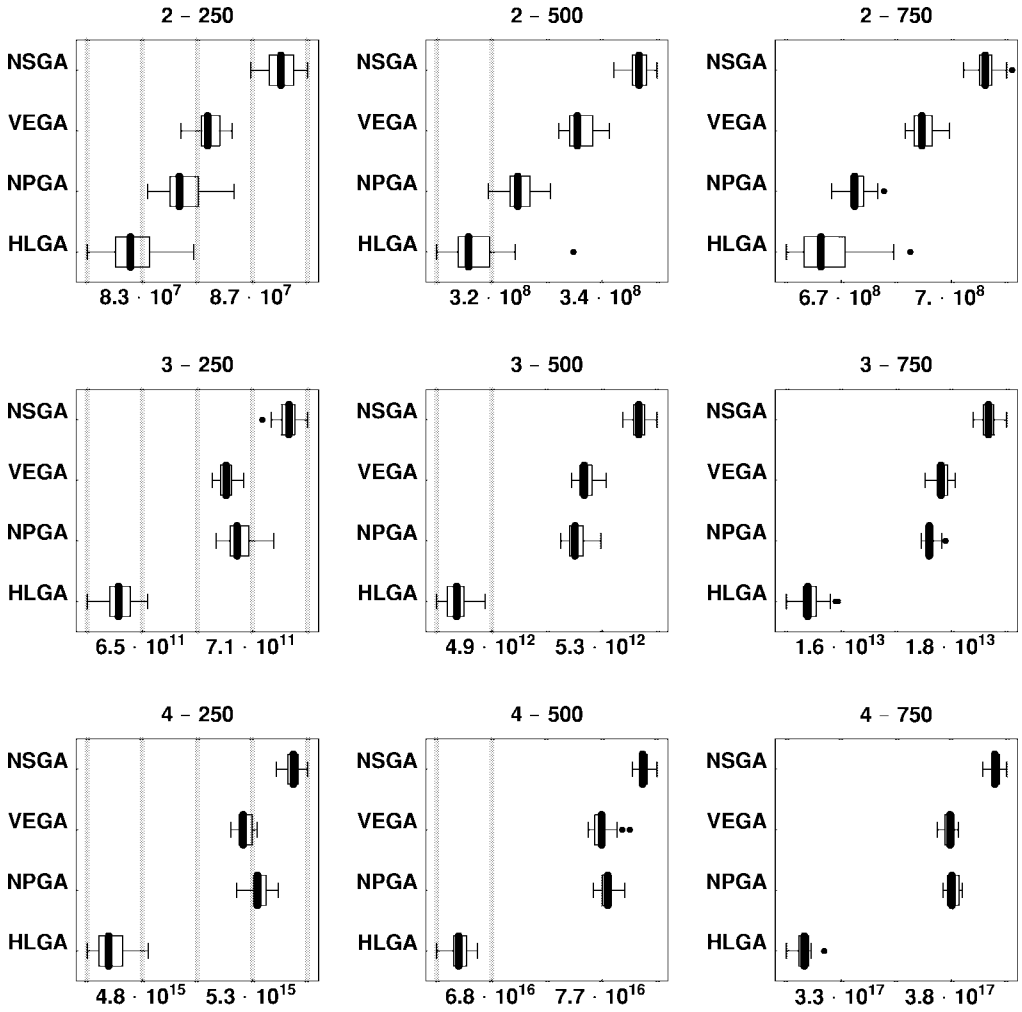


Fig. 3. Distribution of the $\mathcal{S}$ values for the nine test problems. Note that RAND, SO-1, and SO-5 are not considered in this figure, since the focus is on the four multiobjective EA's and otherwise the differences between those algorithms would be blurred.

$\mathcal{S}$ values (the median is greater by more than 21 quartile deviations than the other medians per test problem) suggests that none of the multiobjective EA's converge to the Pareto-optimal front using the chosen parameter settings. This can also be observed in Fig. 1, where the tradeoff fronts obtained in five runs are plotted for the 2-D problems. Note that the computational effort needed by SO-5 to produce the depicted fronts is 20 times higher than the one for the multiobjective EA's.

IV. The Strength Pareto Approach

We propose a new approach to multiobjective optimization, the *Strength Pareto Evolutionary Algorithm* (SPEA). SPEA uses a mixture of established and new techniques in order to find multiple Pareto-optimal solutions in parallel. On one hand, similarly to other multiobjective EA's, it:

- stores the nondominated solutions found so far externally (e.g., [10], [12], [19]);
- uses the concept of Pareto dominance in order to assign scalar fitness values to individuals;
- performs clustering to reduce the number of nondominated solutions stored without destroying the characteristics of the tradeoff front [20].

On the other hand, SPEA is unique in four respects.

- It combines the above three techniques in a single algorithm.
- The fitness of an individual is determined only from the solutions stored in the external nondominated set; whether members of the population dominate each other is irrelevant.
- All solutions in the external nondominated set participate in the selection.
- A new niching method is provided in order to preserve diversity in the population; this method is Pareto-based and does not require any distance parameter (like the niche radius for sharing).

A. Algorithm

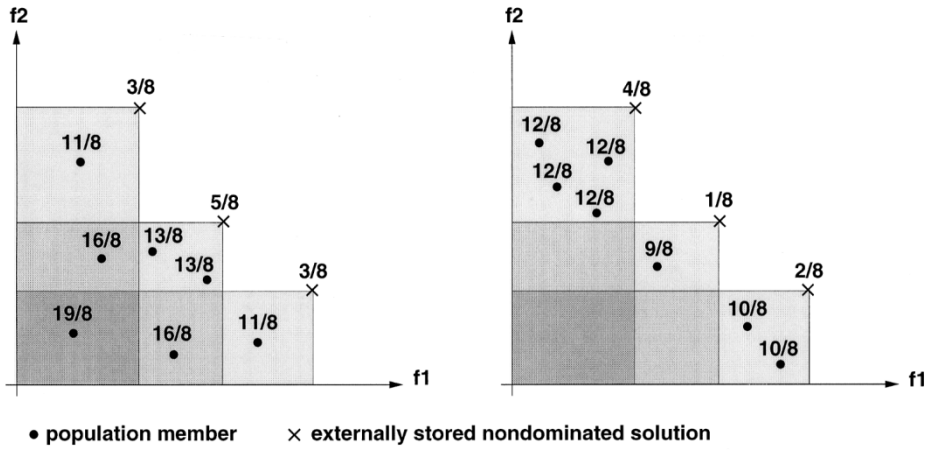
The flow of the algorithm is as follows.

Step 1) Generate an initial population P and create the empty external nondominated set P'.

Step 2) Copy nondominated members of P to P'.

Step 3) Remove solutions within P' which are covered by any other member of P'.

Step 4) If the number of externally stored nondominated solutions exceeds a given maximum N', prune P' by means of clustering.



(a) (b)

Fig. 4. Two scenarios for a maximization problem with two objectives. The number associated with each solution gives the fitness (and strength in case of nondominated points).

Step 5) Calculate the fitness of each individual in P as well as in P'.

Step 6) Select individuals from $P + P'$ (multiset union), until the mating pool is filled. In this study, binary tournament selection with replacement is used.

Step 7) Apply problem-specific crossover and mutation operators as usual.

Step 8) If the maximum number of generations is reached, then stop, else go to Step 2.

In the next two subsections, the fitness assignment as well as the clustering procedure are described in detail.

1) Fitness Assignment: The fitness assignment procedure is a two-stage process. First, the individuals in the external nondominated set P' are ranked. Afterwards, the individuals in the population P are evaluated.

Step 1) Each solution $i \in P'$ is assigned a real value $s_i \in [0, 1)$, called *strength*;[4] s_i is proportional to the number of population members $j \in P$ for which $i \succeq j$. Let n denote the number of individuals in P that are covered by i and assume N is the size of P. Then s_i is defined as $s_i = \frac{n}{N+1}$. The fitness f_i of i is equal to its strength: $f_i = s_i$.

Step 2) The fitness of an individual $j \in P$ is calculated by summing the strengths of all external nondominated solutions $i \in P'$ that cover j. We add one to the total in order to guarantee that members of P' have better fitness than members of P (note that fitness is to be minimized, i.e., small fitness values correspond to high reproduction probabilities)

$$f_j = 1 + \sum_{i, i \succeq j} s_i, \qquad \text{where } f_j \in [1, N).$$

To make the effect of this ranking method clear, take a look at Fig. 4. The objective space which is covered by the three nondominated solutions is divided into distinct rectangles. Each subset of P' defines one such area that all members of the subset cover in common. For instance, the dark-shaded rectangle in the lower-left corner is covered by all three nondominated points, while the upper left bright-shaded rectangle is only covered by one nondominated point. We consider these areas as niches, and the goal is to distribute the individuals over this "grid" such that:

a) (brighter shaded) areas covered by only a few nondominated points contain more individuals than (darker shaded) rectangles that are covered by many nondominated points, and

b) an area comprises as many individuals as the other (equally shaded) rectangles that are covered by the same number of nondominated points.

This mechanism intuitively reflects the idea of preferring individuals near the Pareto-optimal front and distributing them at the same time along the tradeoff surface. In Fig. 4(a), the first aspect is illustrated: Individuals located in the bright areas achieve better fitness values than the remaining population members. Fig. 4(b) provides an example for the second aspect and directly visualizes the strength principle: Individuals having many neighbors in their niche are penalized due to the high strength value of the associated nondominated point; the "stronger" a nondominated solution, the less "fitter" are the covered individuals.

The main difference to fitness sharing is that niches are not defined in terms of distance but Pareto dominance. This renders the setting of a distance parameter superfluous, although the parameter N' influences the niching capability as we will discuss in the next section. Furthermore, it has to be mentioned that this kind of fitness assignment using two interacting populations has been inspired by [37]–[41]. Paredis [41] studied the use of cooperating populations in EA's and showed that symbiotic evolution can speed up the search process. In [37]–[40], a similar concept was applied to immune system models where two cooperative populations were used to maintain population diversity; [39] reported that this method has emergent properties that are similar to fitness sharing.

[4]This term is adopted from [36] where it was introduced in the context of classifier systems; it stands for a quantity summarizing the usefulness of a rule. Here, it reflects the usefulness of a nondominated point.

2) Reducing the Pareto Set by Clustering: In certain problems, the Pareto-optimal set can be extremely large or even contain an infinite number of solutions. However, from the decision maker's point of view, presenting all nondominated solutions found is useless when their number exceeds reasonable bounds. Moreover, the size of the external nondominated set influences the behavior of SPEA. On the one hand, since P' participates in selection, too many nondominated solutions might reduce selection pressure and slow down the search [20]. On the other hand, the strength niching mechanism relies on a uniform granularity of the "grid" defined by the nondominated solutions (cf. Fig. 4); if the points in P' are not distributed uniformly, the fitness assignment method is possibly biased toward certain regions of the search space, leading to an unbalanced distribution in the population. Thus pruning the external nondominated set while maintaining its characteristics might be necessary or even mandatory.

A method that has been applied to this problem successfully and studied extensively in the same context is cluster analysis [42], [43]. In general, cluster analysis partitions a collection of m elements into n groups of relatively homogeneous elements, where $n < m$. The *average linkage method* [42], a clustering approach that has proven to perform well on this problem (cf. [42]), has been chosen in this paper.

Step 1) Initialize cluster set C; each external nondominated point $i \in P'$ constitutes a distinct cluster: $C = \bigcup_i \{\{i\}\}$.

Step 2) If $|C| \leq N'$, go to Step 5, else go to Step 3.

Step 3) Calculate the distance of all possible pairs of clusters. The distance d of two clusters c_1 and $c_2 \in C$ is given as the average distance between pairs of individuals across the two clusters

$$d = \frac{1}{|c_1| \cdot |c_2|} \cdot \sum_{i_1 \in c_1, i_2 \in c_2} \|i_1 - i_2\|$$

where the metric $\|\cdot\|$ reflects the distance between two individuals i_1 and i_2 (in this study an Euclidean metric on the objective space is used).

Step 4) Determine two clusters c_1 and c_2 with minimal distance d; the chosen clusters amalgamate into a larger cluster: $C = C \setminus \{c_1, c_2\} \cup \{c_1 \cup c_2\}$. Go to Step 2.

Step 5) Compute the reduced nondominated set by selecting a representative individual per cluster. We consider the centroid (the point with minimal average distance to all other points in the cluster) as representative solution.

Cunha *et al*. [20] also combined a multiobjective EA with a clustering approach in order to achieve reasonably sized Pareto sets. This algorithm, however, uses a different clustering method which has been proposed in [43]; thereby, for each objective, a tolerance value has to be specified. Moreover, it differs from SPEA with regard to the following two aspects: a) The nondominated solutions are not stored externally, and b) fitness sharing is incorporated to preserve diversity in the population.

B. A Simple Test Function: Schaffer's f_2

A very simple test function for multiobjective optimizers is the well-known function f_2 used by Schaffer [44]. It is defined as follows:

$$\begin{aligned} \text{minimize } & f_2(x) = (g(x), h(x)) \\ \text{where } & g(x) = x^2 \\ & h(x) = (x-2)^2. \end{aligned} \tag{8}$$

Obviously, the Pareto-optimal points are located in the range $x \in [0, 2]$. Outside this interval, g as well as h are increasing, while within the interval, there is a tradeoff between the two functions (one is increasing, the other one is decreasing).

To test SPEA on f_2, we used a 14-bit chromosome which is decoded to a real number between -6 and 6. The bit string 00000000000000 encodes $x = -6$ and 11111111111111 stands for $x = 6$. Furthermore, the following parameters were used for SPEA:

Population size (N):	95/70/30
Size of external nondominated set (N'):	5/30/70
Crossover probability:	1.0
Mutation probability:	0.0
Number of generations:	100

Altogether, we tried three different combinations of N and N', where $N + N'$ equaled 100 in each case. In order to examine the effectiveness of SPEA alone, no mutation operator was applied to the individuals. Instead, we used a crossover probability of 1.0. In addition, VEGA ran on this problem with identical parameters $(N = 100)$. In order to guarantee a fair comparison, the offline performance of VEGA is considered here, i.e., the final tradeoff front is formed by the nondominated solutions found *during* a run, not only by the Pareto-optimal points in generation 100.

The results produced by the algorithms using the same initial population are shown in Fig. 5.[5] It can be observed that SPEA is able to well approximate the Pareto-optimal front, depending on the size of the external nondominated set. In comparison to VEGA, it evolved more Pareto-optimal solutions (VEGA:20, SPEA:5/30/70) and distributed them more uniformly along the tradeoff front.

C. Performance on the 0/1 Knapsack Problem

The same parameters as for the other multiobjective EA's were used for SPEA on the 0/1 knapsack problem. For reasons of fairness, N was set to $4/5$ and N' to $1/4$ of the population size given in Table I. In addition, a slightly modified version of SPEA was examined (SP-S) where P' does not participate in the selection phase; there, the population size was the same as for the other EA's, and the size of the external nondominated set was restricted to $1/4 \cdot N$.

The results concerning the $\mathcal{S}$ measure (size of the space covered) are depicted in Fig. 6, the direct comparison of SPEA

[5] Certainly, only limited weight can be given to a single run per algorithm. Nevertheless, the results were similar when the experiments were repeated with different initial populations.

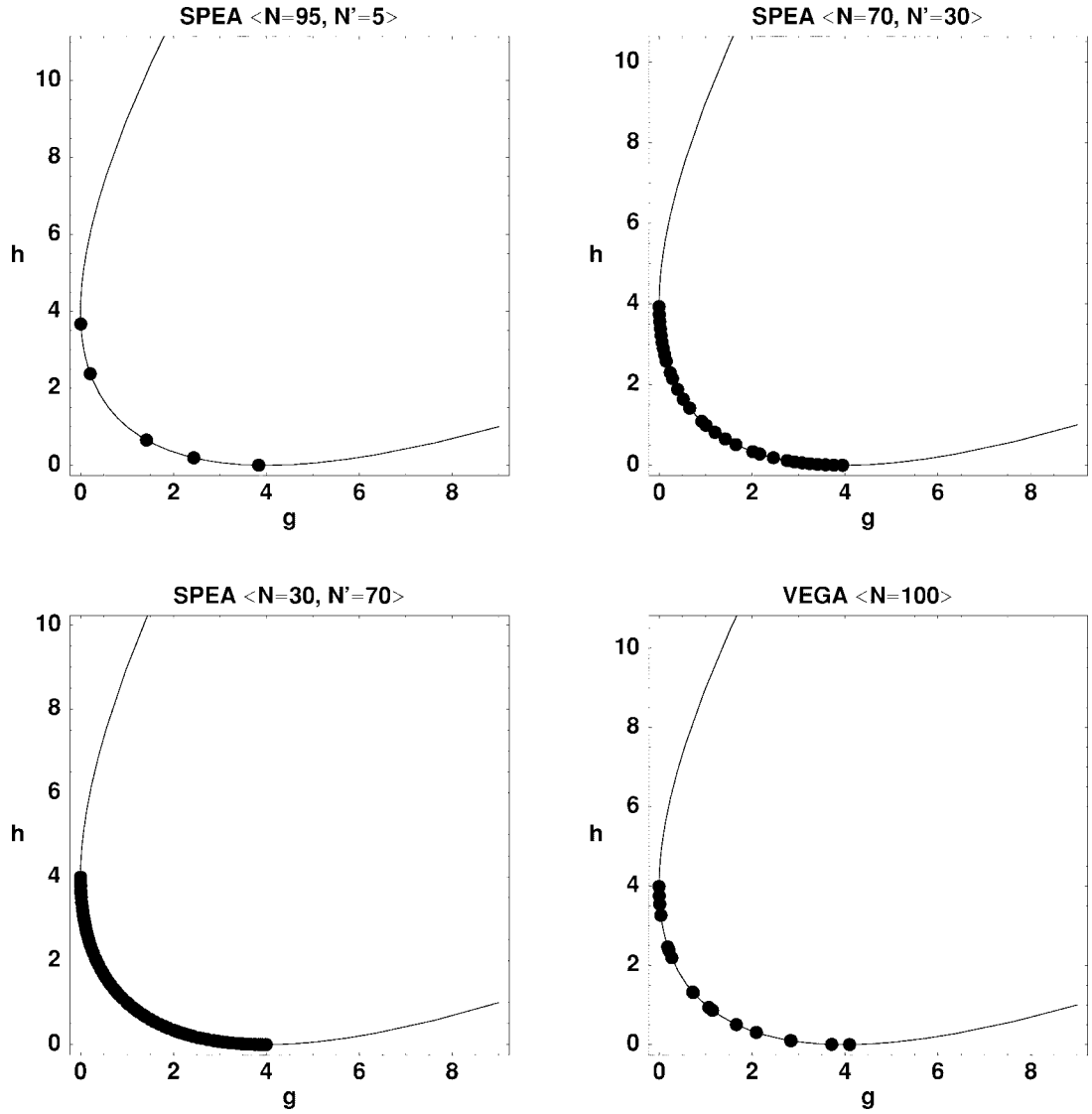


Fig. 5. Performance of SPEA and VEGA on Schaffer's f_2.

with the other algorithms based on the $\mathcal{C}$ measure (coverage) is shown in Fig. 2. Furthermore, Fig. 1 gives the plots of the 2-D tradeoff fronts achieved by SPEA and the other EA's. The main observations can be summarized as follows.

- SPEA achieves the best assessments among the multiobjective EA's. It covers 100% of the nondominated solutions found by HLGA, NPGA, VEGA, and NSGA with eight of the nine test problems; for four knapsacks and 250 items at least 87% are covered. On the other hand, those algorithms cover less than 5% of the SPEA outcomes in all 270 runs. Concerning the size of the covered space, the medians of the $\mathcal{S}$ distributions related to SPEA are greater than the corresponding medians of the other multiobjective EA's by more than ten quartile deviations. Although the Pareto-optimal fronts of the test problems considered here are all convex, we have shown recently [45] that SPEA also has advantages over the other EA's for different types of problems (e.g., nonconvex functions).
- As Fig. 1 indicates, SPEA can find solutions that are closer to the Pareto-optimal front than those produced by SO-5 in spite of less computational effort. This observation is supported by the fact that SO-5 covers only 48% of the SPEA front with eight of the nine test problems (SO-1 less than 12%). However, the fronts found by multiple single-objective searches contain many more solutions and are wider in the sense that the size of the covered space is significantly greater (cf. Fig. 6). Whether SPEA can outperform a single-objective EA with substantially less computation time is the subject of future work; however, it was shown recently [45] that this is the case for 2-D problems of different characteristics.
- Elitism seems to be important for the effectiveness of the search, as SP-S performs substantially worse than SPEA. Nevertheless, SP-S appears to do slightly better than NSGA on the three- and four-dimensional problems. Both the $\mathcal{S}$ values (the median distance to NSGA is greater than three quartile deviations) and the $\mathcal{C}$ values suggest a slight advantage for SP-S over NSGA. For two knapsacks, the results are ambiguous and do not allow a final conclusion to be made.

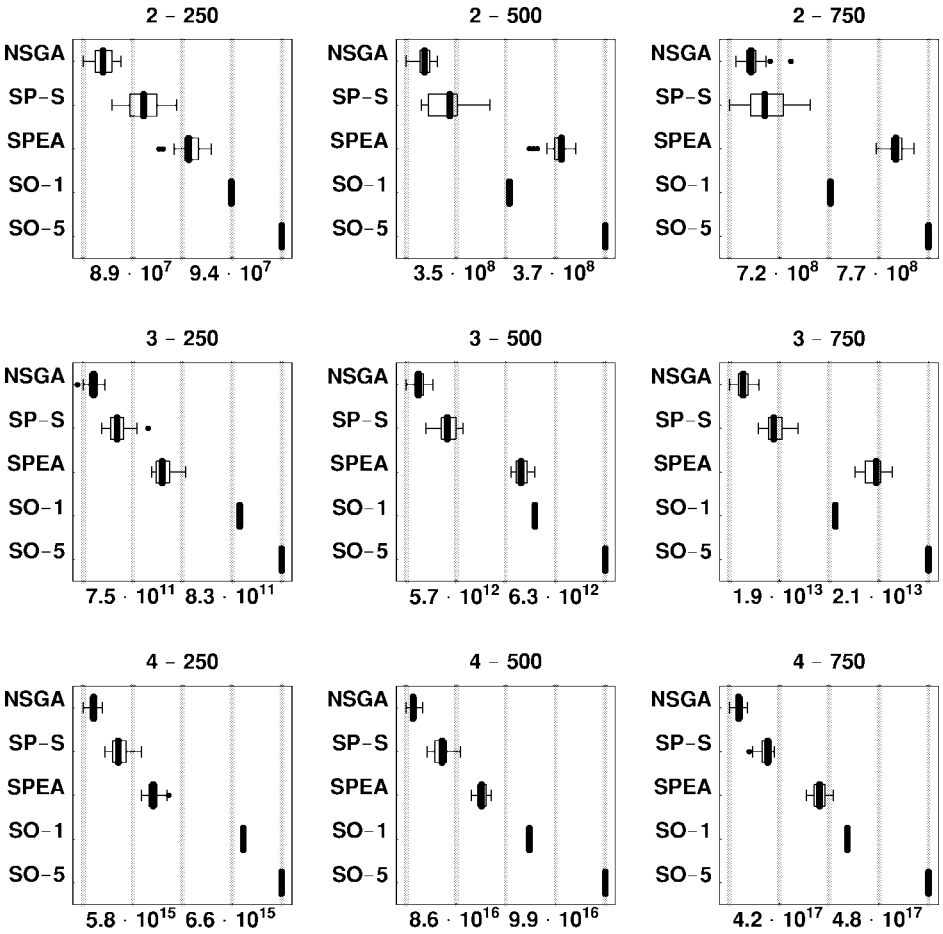


Fig. 6. SPEA in comparison with the other algorithms with regard to the size of the covered space. The box plots represent the distributions of the $\mathcal{S}$ values achieved in the 30 optimization runs.

D. Application to System-Level Synthesis

The third application is a larger problem in the domain of computer engineering that is concerned with computer-based system-level synthesis. Blickle *et al*. [24], [46], [47] have presented an evolutionary approach to this problem which we use as the basis for the SPEA implementation.

1) Problem Description: In [47], system-level synthesis is considered as the problem of optimally mapping a task-level specification onto a heterogeneous hardware/software architecture. The input consists of three parts.

1) A behavioral description of a hardware/software system to synthesize. The behavior is defined in terms of functional objectives such as algorithms, tasks, procedures, or processes together with their data interdependencies.
2) A structural specification of the system (= a class of possible architectures) where structural objects are general- or special-purpose processors, application-specific integrated circuits (ASIC's), buses, and memories. With each structural object, a fixed cost is associated that arises when the particular resource is realized.
3) A Boolean function m of the set of functional objects to the set of structural objects that defines the space of possible mappings; when $m(a,b) = 1$, the task a can be mapped to the resource b, otherwise it cannot. Additionally, a latency function l gives the estimated time $l(a,b)$ that is necessary to execute task a on resource b.

The optimization goal is to find an implementation which simultaneously minimizes cost and execution time; thereby, an implementation is described by:

1) the set of the selected resources and structural objects (*allocation*);
2) the mapping of the algorithm onto the selected architecture (*binding*);
3) the schedule that defines the start times of the tasks on the selected resources.

An example that visualizes the relations between input and output is provided in Fig. 7. The behavioral specification described by means of a directed graph contains seven functional objects, where shaded nodes stand for communication operations. The architecture, which includes a RISC processor, a digital signal processor (DSP), and an application-specific integrated circuit (ASIC), interconnected by two buses, is also modeled by a directed graph. Finally, the function m is represented by edges between nodes of the two graphs. For instance, Algorithm 4 can be mapped to any chip while Algorithm 1 has to be executed on the RISC processor. On the right-hand side of Fig. 7, a sample implementation is depicted. All resources except Bus 2 are selected, thus all communications are handled by Bus 1 (this is also reflected

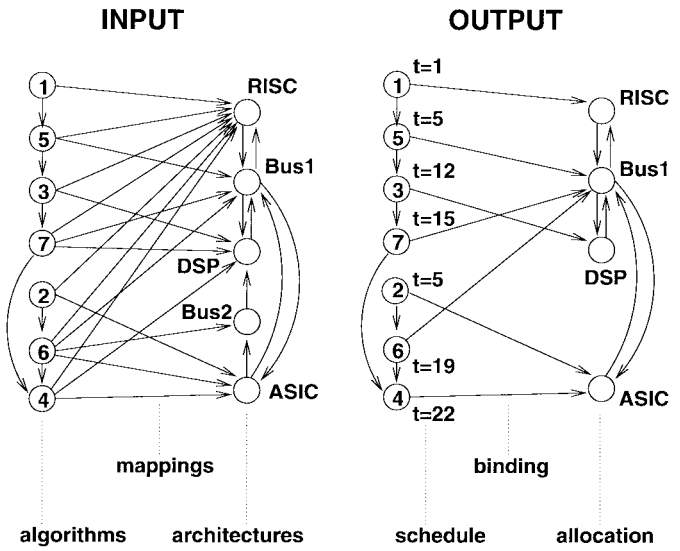


Fig. 7. System-level synthesis: problem statement (slightly modified example from [24]). Given is a set of algorithms together with their data interdependencies, a superset of possible architectures, and a set of possible mappings of algorithms to computing resources. The goal is to find an implementation that is described by the selected architecture (allocation), the selected mapping (binding), and a schedule for the algorithms to be executed on the architecture.

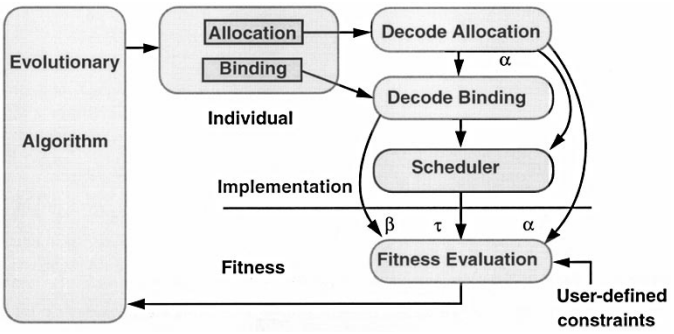


Fig. 8. An evolutionary algorithm for system-level synthesis (diagram taken from [24, p. 174]). Depicted is the process of fitness evaluation. In the first step, an allocation is derived from the information encoded in the individual. The binding, which is computed in the second step, depends on both the encoded information and the allocation. Afterwards, the schedule is determined heuristically and the resulting implementation is assessed concerning the design criteria, possibly taking user-defined constraints into account.

by the binding that maps the communication nodes 5, 7, and 6 to Bus 1). For each functional object, a start time is given (schedule).

2) EA Implementation: The overall picture of the EA is depicted in Fig. 8. Each individual encodes both allocation and binding, whereas the schedule is computed deterministically by a heuristic list-scheduling algorithm incorporating loop pipelining. An allocation is intuitively represented as a binary string, the length of which corresponds to the number of specified resources in the set of possible architectures. In order to reduce the number of unfeasible solutions, allocations are partially repaired by a heuristic whenever an individual is decoded. For the same reason, bindings are not encoded directly using one chromosome but rather indirectly based on several chromosomes: one chromosome including a permutation of all tasks in the behavioral description determines the order in which the tasks are mapped to the resources with respect to the repaired allocation. Further lists, permutations of the set of resources, define separately for each task which resource is to be checked next for mapping.

To obtain the entire Pareto-optimal front (design space exploration), [24] used the same Pareto ranking method proposed in [14]: An individual's fitness is equal to the number of population members that dominate it. For the purpose of a diverse population, he incorporated a niching technique which has been rather seldom used: *restricted tournament selection* (RTS) [48]. RTS is a special binary tournament selection for steady-state EA's where two individuals hold tournament with the most similar individual of a randomly chosen group; winners replace inferior individuals in the population.

3) Experimental Results: The presented EA has been implemented with the Strength Pareto approach for multiobjective optimization and compared both to a single-objective EA and to the algorithm proposed in [24]. The synthesis of a video codec, based on the H.261 standard (cf. [24, ch. 9]), was chosen as test problem; the search space of this problem contains about $1.9 \cdot 10^{27}$ possible bindings.

TABLE II
VIDEO CODEC: NONDOMINATED SOLUTIONS FOUND BY THE THREE DIFFERENT METHODS. IN EACH COLUMN, THE PAIRS SET IN ITALIC MARK POINTS THAT ARE INFERIOR TO ANY POINT IN THE OTHER TWO COLUMNS. THE OUTCOMES OF THE SINGLE-OBJECTIVE EA ARE TAKEN FROM [24, p. 203]

SPEA	single-objective EA	RTS + Pareto ranking
(180,166)	(180,166)	(180,166)
(230,114)	(230,114)	(230,114)
(280,78)	(280,78)	(280,78)
(330,48)	*(330,54)*	*(330,54)*
(340,36)	*(340,42)*	*(350,23)*
(350,22)	(350,22)	*(370,22)*

All algorithms ran with a population size of 30 (SPEA: 20 with ten externally stored nondominated solutions), a crossover probability of 0.5, and a mutation probability of 0.2. In case of the two multiobjective EA's, the offline performance over ten independent runs with 100 generations each was considered. The single-objective EA was used to optimize each objective separately; for the other objective, a maximum value, a constraint, was defined. We examined 11 different latency constraints when minimizing cost and 11 cost constraints in the case of latency optimization. For each constraint, the best result out of ten independent runs (100 generations each) was taken, and the nondominated solutions of all 22 single-objective results constituted the final Pareto set.

SPEA covers 100% and dominates 50% of the solutions found by the combination of RTS and Pareto ranking as shown in Table II. Although Blickle [24] ran the algorithm with a population size of 100 and a maximum number of 200 generations, the results he reported are the same as generated by the single-objective EA (Table II, second column). Moreover, in spite of significantly lower computational effort, SPEA covers 100% and dominates 33% of the nondominated front achieved by the single-objective EA.

V. Conclusion

This study compared four multiobjective EA's on a multiobjective 0/1 knapsack problem with nine different problem settings. The quality of the Pareto-optimal sets achieved was measured quantitatively by the size of the covered space. Additionally, the approaches were compared directly by evaluating the outcomes regarding the concept of Pareto dominance.

All multiobjective EA's clearly outperformed a pure random search strategy which randomly generates new points in the search space without exploiting similarities between solutions. Among these multicriteria EA's, the nondominated sorting genetic algorithm [6] achieved the best results on all test problems. It was followed by VEGA [9] which seems to have slight advantages over the niched Pareto genetic algorithm [18], [26] on this type of problem. Compared with Hajela's and Lin's weighted-sum approach [11], both VEGA and NPGA were assessed as better regarding the two performance measures considered here.

Furthermore, a new evolutionary approach to multiobjective optimization has been provided (SPEA) that differs from existing multicriteria EA's in the kind of fitness assignment based on principles of coevolution and the niching technique founded on the concept of Pareto dominance. As shown on three applications, SPEA is capable of efficiently guiding the search toward the Pareto-optimal front. On the 0/1 knapsack problem, it outperformed the other four multiobjective EA's by a wide margin. Moreover, the experimental results indicate that SPEA can even find solutions that are closer to the globally optimal trade-off surface than solutions evolved by a single-objective EA optimizing a linear combination of the objectives.

With regard to future perspectives, it may be worthwhile to investigate the following issues.

- If possible, other probabilistic search algorithms like simulated annealing, hill climbing, tabu search, etc., as well as "exact" methods (e.g., integer linear programming, branch-and-bound) and deterministic heuristics (cf. [31]) should be tested on the multiobjective 0/1 knapsack problem. This would permit a more precise assessment of the performance of the EA's.
- The distribution of the obtained nondominated sets should be included in the comparison. Although the size of the covered space is a performance measure that takes this property into account, it does not allow separate evaluation of the distribution.
- Comparative studies should also be performed on the basis of other test problems with different characteristics (e.g., nonconvexity). First steps in this direction have already been made [45].

Finally, as stated in [1], a theory of evolutionary multiobjective optimization is still required, examining different fitness assignment methods in combination with different selections schemes.

Acknowledgment

The authors wish to thank D. B. Fogel and the anonymous reviewers for their helpful comments and suggestions.

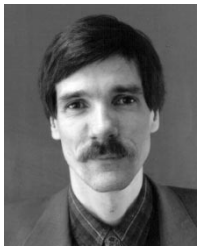
References

[1] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evol. Comput.*, vol. 3, no. 1, pp. 1–16, 1995.
[2] M. Valenzuela-Rendón and E. Uresti-Charre, "A nongenerational genetic algorithm for multiobjective optimization," in *Proc. 7th Int. Conf. Genetic Algorithms*, T. Bäck, Ed. San Francisco, CA: Morgan Kaufmann, 1997, pp. 658–665.
[3] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 67–82, Apr. 1997.
[4] J. Horn, "F1.9 Multicriteria decision making," in *Handbook of Evolutionary Computation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. Bristol, U.K.: Inst. Phys. Pub., 1997.
[5] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms—A comparative case study," in *5th Int. Conf. Parallel Problem Solving from Nature (PPSN-V)*, A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 292–301.
[6] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1994.
[7] R. E. Steuer, *Multiple Criteria Optimization: Theory, Computation, and Application*. New York: Wiley, 1986.
[8] J. L. Ringuest, *Multiobjective Optimization: Behavioral and Computational Considerations*. Boston, MA: Kluwer, 1992.
[9] M. P. Fourman, "Compaction of symbolic layout using genetic algorithms," in *Proc. Int. Conf. Genetic Algorithms and Their Applications*, J. J. Grefenstette, Ed., Pittsburgh, PA, July 24–26, 1985, pp. 141–153, sponsored by Texas Instruments and U.S. Navy Center for Applied Research in Artificial Intelligence (NCARAI).
[10] F. Kursawe, "Evolution strategies for vector optimization," in *Proc. 10th Int. Conf. Multiple Criteria Decision Making*, G.-H. Tzeng and P. L. Yu, Eds., National Chiao Tung University, Hsinchu, Taiwan, 1992, pp. 187–193.
[11] P. Hajela and C.-Y. Lin, "Genetic search strategies in multicriterion optimal design," in *Structural Optimization*, vol. 4. New York: Springer, June 1992, pp. 99–107.
[12] H. Ishibuchi and T. Murata, "Multi-objective genetic local search algorithm," in *Proc. 1996 IEEE Int. Conf. Evolutionary Computation (ICEC'96)*, Piscataway, NJ, May 20–22, 1996, pp. 119–124.
[13] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[14] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Proc. 5th Int. Conf. Genetic Algorithms*, S. Forrest, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 416–423.
[15] G. W. Greenwood, X. S. Hu, and J. G. D'Ambrosio, "Fitness functions for multiple objective optimization problems: Combining preferences with pareto rankings," in *Foundations of Genetic Algorithms 4 (FOGA-96)*, R. K. Belew and M. D. Vose, Eds. San Francisco, CA: Morgan Kaufmann, 1996, pp. 437–455.
[16] S. W. Mahfoud, "Niching methods for genetic algorithms," Ph.D. dissertation, Univ. Illinois, Urbana-Champaign, 1995.
[17] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, J. J. Grefenstette, Ed. Hillsdale, NJ: Lawrence Erlbaum, 1987, pp. 41–49.
[18] J. Horn and N. Nafpliotis, "Multiobjective optimization using the niched pareto genetic algorithm," IlliGAL Report 93005, Illinois Genetic Algorithms Lab., Univ. Illinois, Urbana-Champaign, July 1993.
[19] D. S. Todd and P. Sen, "A multiple criteria genetic algorithm for containership loading," in *Proc. 7th Int. Conf. Genetic Algorithms*, T. Bäck, Ed. San Francisco, CA: Morgan Kaufmann, 1997, pp. 674–681.
[20] A. G. Cunha, P. Oliviera, and J. Covas, "Use of genetic algorithms in multicriteria optimization to solve industrial problems," in *Proc. 7th Int. Conf. Genetic Algorithms*, T. Bäck, Ed. San Francisco, CA: Morgan Kaufmann, 1997, pp. 682–688.
[21] D. H. Loughlin and S. Ranjithan, "The neighborhood constraint-method: A genetic algorithm-based multiobjective optimization technique," in *Proc. 7th Int. Conf. Genetic Algorithms*, T. Bäck, Ed. San Francisco, CA: Morgan Kaufmann, 1997, pp. 666–673.
[22] K. A. De Jong, "An analysis of the bevavior of a class of genetic adaptive systems," Ph.D. dissertation, Univ. Michigan, Ann Arbor, 1975.
[23] C. Ryan, "Niche and species formation in genetic algorithms," in *Practical Handbook of Genetic Algorithms*, vol. 1, L. Chambers, Ed. Boca Raton, FL: CRC, 1995, ch. 2, pp. 57–74.

[24] T. Blickle, "Theory of evolutionary algorithms and application to system-synthesis," Ph.D. dissertation, Swiss Federal Inst. Technol. (ETH), Zürich, Switzerland, 1996, ETH diss no. 11894.
[25] H. Tamaki, H. Kita, and S. Kobayashi, "Multi-objective optimization by genetic algorithms: A review," in *Proc. 1996 IEEE Int. Conf. Evolutionary Computation (ICEC'96)*, Piscataway, NJ, May 20–22, 1996, pp. 517–522.
[26] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," in *Proc. 1st IEEE Conf. Evolutionary Computation, IEEE World Congr. Computational Computation*, Piscataway, NJ, June 27–29, 1994, vol. 1, pp. 82–87.
[27] S. Khuri, T. Bäck, and J. Heitkötter, "The zero/one multiple knapsack problem and genetic algorithms," in *Proc. 1994 ACM Symp. Applied Computing*, E. Deaton, D. Oppenheim, J. Urban, and H. Berghel, Eds. New York: ACM-Press, 1994, pp. 188–193.
[28] Z. Michalewicz and J. Arabas, "Genetic algorithms for the 0/1 knapsack problem," in *Methodologies for Intelligent Systems (ISMIS'94)*, Z. W. Raś and M. Zemankova, Eds. Berlin, Germany-Verlag: Springer, 1994, pp. 134–143.
[29] R. Spillman, "Solving large knapsack problems with a genetic algorithm," in *IEEE Int. Conf. Systems, Man and Cybernetics*, Piscataway, NJ, Oct. 22–25, 1995, vol. 1, pp. 632–637.
[30] M. Sakawa, K. Kato, and T. Shibano, "An interactive fuzzy satisficing method for multiobjective multidimensional 0-1 knapsack problems through genetic algorithms," in *Proc. 1996 IEEE Int. Conf. Evolutionary Computation (ICEC'96)*, Piscataway, NJ, May 20–22, 1996, pp. 243–246.
[31] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. Chichester, U.K.: Wiley, 1990.
[32] T. Blickle and L. Thiele, "A comparison of selection schemes used in evolutionary algorithms," *Evol. Comput.*, vol. 4, no. 4, pp. 361–394, 1996.
[33] C. K. Oei, D. E. Goldberg, and S.-J. Chang, "Tournament selection, niching, and the preservation of diversity," IlliGAL Rep. 91011, Univ. Illinois, Urbana-Champaign, Urbana, IL 61801, Dec. 1991.
[34] K. Deb, and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 42–50.
[35] J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey, *Graphical Methods for Data Analysis*. Pacific Grove, CA: Wadsworth & Brooks/Cole, 1983.
[36] J. H. Holland, *Adaption in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.
[37] S. Forrest and A. S. Perelson, "Genetic algorithms and the immune system," in *Parallel Problem Solving from Nature (PPSN I)*, H.-P. Schwefel and R. Männer, Eds. Berlin, Germany: Springer-Verlag, 1991, pp. 320–325.
[38] R. E. Smith and S. Forrest, "Population diversity in an immune system model: Implications for genetic search," *Foundations of Genetic Algorithms 2 (FOGA-92)*, L. D. Whitley, Ed. San Mateo, CA: Morgan Kaufmann, 1992.
[39] R. E. Smith, S. Forrest, and A. S. Perelson, "Searching for diverse, cooperative populations with genetic algorithms," *Evol. Comput.*, vol. 1, no. 2, pp. 127–149, 1993.
[40] S. Forrest, B. Javornik, R. E. Smith, and A. S. Perelson, "Using genetic algorithms to explore pattern recognition in the immune system," *Evol. Comput.*, vol. 1, no. 3, pp. 191–211, 1993.
[41] J. Paredis, "The symbiotic evolution of solutions and their representations," in *Proc. 6th Int. Conf. Genetic Algorithms*, L. J. Eshelman, Ed. San Francisco, CA: Morgan Kaufmann, 1995, pp. 359–365.
[42] J. N. Morse, "Reducing the size of the nondominated set: Pruning by clustering," *Comput. Oper. Res.*, vol. 7, nos. 1–2, 1980.
[43] M. A. Rosenman and J. S. Gero, "Reducing the pareto optimal set in multicriteria optimization," *Eng. Optim.*, vol. 8, pp. 189–206, 1985.
[44] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," Ph.D. dissertation, Vanderbilt Univ., Nashville, TN, 1984.
[45] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," Tech. Rep. 70, Comput. Eng. Networks Lab. (TIK), Swiss Federal Inst. Technol. (ETH) Zurich, Switzerland, Feb. 1999.
[46] J Teich, T. Blickle, and L. Thiele, "System-level synthesis using evolutionary algorithms," in *Proc. Codes/CASHE'97, 5th Int. Worksh. Hardware/Software Codesign*. Los Alamitos, CA: IEEE Comput. Soc. Press, 1997, pp. 167–171.
[47] T. Blickle, J. Teich, and L. Thiele, "System-level synthesis using evolutionary algorithms," *Des. Automat. Embedded Syst.*, vol. 3, no. 1, pp. 23–58, 1998.
[48] G. R. Harik, "Finding multimodal solutions using restricted tournament selection," in *Proc. 6h Int. Conf. Genetic Algorithms*, L. J. Eshelman, Ed. San Francisco, CA: Morgan Kaufmann, 1995, pp. 24–31.

Eckart Zitzler received the Diploma degree in computer science in 1996 from University of Dortmund, Dortmund, Germany.

Since 1996, he has been a Research and Teaching Assistant at the Computer Engineering Group at the Electrical Engineering Department of ETH Zurich, Switzerland. His main research interests are in the areas of evolutionary computation, multiobjective optimization, and computer engineering.



Lothar Thiele recieved the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering from Technical University of Munich, Munich, Germany, in 1981 and 1985, respectively.

Since 1981, he has been a Research Associate with Prof. R. Saal at the Institute of Network Theory and Circuit Design of the Technical University Munich. After finishing his Habilitation thesis, he joined the group of Prof. T. Kailath at the Information Systems Laboratory, Stanford University, Stanford, CA, in 1987. In 1988, he became Chair of Microelectronics at the faculty of engineering, University of Saarland, Saarbrücken, Germany. He joined ETH Zürich, Switzerland, as a Full Professor in Computer Engineering in the Fall of 1994. His research interests include models, methods and software tools for the design of hardware/software systems, and array processors as well as the development of parallel algorithms for signal and image processing, combinatorial optimization, and cryptography. He authored and co-authored more than 100 papers.

In 1986, Dr. Thiele received the award of the Technical University of Munich for his Ph.D. dissertation. He received the 1987 Outstanding Young Author Award of the IEEE Circuits and Systems Society. In 1988, he was the recipient of the 1988 Browder J. Thompson Memorial Prize Award of the IEEE.