On The Effects of Archiving, Elitism, And Density Based Selection in Evolutionary Multi-Objective Optimization

Marco Laumanns, Eckart Zitzler, and Lothar Thiele

ETH Zürich, Institut TIK, CH–8092 Zürich, Switzerland,
{laumanns,zitzler,thiele}@tik.ee.ethz.ch,
http://www.tik.ee.ethz.ch/aroma

Abstract. This paper studies the influence of what are recognized as key issues in evolutionary multi-objective optimization: archiving (to keep track of the current non-dominated solutions), elitism (to let the archived solutions take part in the search process), and diversity maintenance (through density dependent selection). Many proposed algorithms use these concepts in different ways, but a common framework does not exist yet. Here, we extend a unified model for multi-objective evolutionary algorithms so that each specific method can be expressed as an instance of a generic operator. This model forms the basis for a new type of empirical investigation regarding the effects of certain operators and parameters on the performance of the search process. The experiments of this study indicate that interactions between operators as well as between standard parameters (like the mutation intensity) cannot be neglected. The results lead not only to better insight into the working principle of multi-objective evolutionary algorithms but also to design recommendations that can help possible users in including the essential features into their own algorithms in a modular fashion.

1 Introduction

Evolutionary algorithms have shown to be a useful auxiliary tool for approximating the Pareto set of multi-objective optimization problems. Several surveys of evolutionary multi-objective algorithms can be found in the literature, e.g., [5, 17, 1], which reflects the large number of different evolutionary approaches to multi-objective optimization proposed to date. While most of these algorithms were designed with regard to two common goals, fast and reliable convergence to the Pareto set and a good distribution of solutions along the front, virtually each algorithm represents a unique combination of specific techniques to achieve these goals.

As a consequence, there are no common guidelines available, how to best tailor an evolutionary algorithm to an application involving multiple objectives. Recently, some researchers have tried to address this problem by carrying out extensive comparative case studies, which can roughly be divided into two different categories. The first group compares different algorithms [17, 8, 19, 18, 20], but as the algorithms usually differ in more that just one aspect, it is very difficult to identify the features which are mainly responsible for the better performance of one algorithm over another. On the contrary, a few other studies take one algorithm and focus on a specific operator or parameter to tune, e.g. the selection method [10, 12]. In this case the results are valid for the algorithm under concern and highly dependent on the other algorithmic parameters. Hence, it has remained open up to now

- how a certain parameter or a certain operator affects the overall performance independent of the specific implementation and the other techniques used, and
- how the parameters and operators influence each others' performance.

The reason why these questions cannot be answered easily is the vast number of details by which the various implementations differ. To overcome this difficulty, in a first step we have developed a unified model representing a general structure that most algorithms have in common; this allows to assess important algorithmic features by separating them from the implementations-specific details [9]. Based on this model, we here investigate archiving, elitism, and density estimation methods and present an appropriate statistical methodology to identify significant effects and interactions, which will be carried out for the example of the knapsack problem.

Section 2 starts with the classification of the key concepts in evolutionary multi-objective optimization and the description of the model. The experimental design is explained in section 3. In the remaining parts, experiments focus on the key parameters and operators in multi-objective evolutionary algorithms: Section 4 describes the notion of elitism and investigates the benefits of elitism and the interactions with mutation. In section 5 we deal with archiving methods and different possibilities to limit the size of the archive. In section 6 we investigate different density estimation techniques and how they can be used to guide the selection of parents for the next generation. Finally, we discuss the results and their implications for the design of multi-objective evolutionary algorithms.

2 Background

2.1 Issues in Evolutionary Multi-objective Optimization

Fig. 1 gives an overview on the different techniques that are applied in most multi-objective evolutionary algorithms. Though some are also used in single-objective optimization (e.g. fitness sharing), most of the features are especially devoted to the task of maintaining a diverse set of solutions in an multi-dimensional objective space.

Many multi-objective evolutionary algorithms maintain a secondary population – or archive – parallel to the normal offspring population [17, pp.3-26–3-28]. Generally, an archive can be used to store any kind of useful information gathered during the run, but in most cases it just contains non-dominated solutions and therefore approximates the Pareto set. If the archived solutions reproduce as well, we say the algorithm uses elitism. Recent studies suggest that the use of elitism improves multi-objective evolutionary algorithms [10, 12, 18].

Another issue is the assignment of fitness values to the individuals. In this study we concentrate on Pareto-based methods because of their acknowledged advantages over aggregation and population-based methods [5]. Different techniques inferring a scalar value from the partially ordered objective vectors include the dominance level (or non-dominated sorting [6, 15]), the dominance grade or rank [4] and the 'strength' measure [20]. Some algorithms use a further means to reach a better or more uniform distribution

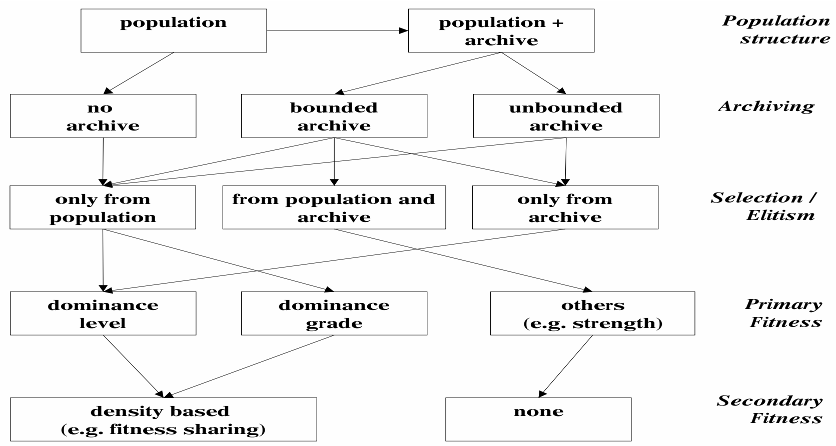

Fig. 1. A taxonomy of techniques in multi-objective evolutionary algorithms, the arrows indicate existing combinations.

of solutions: Individuals receive a secondary fitness value, mainly based on the density of search points near them. This density value can then be used to modify the primary fitness, e.g. via fitness sharing.

2.2 The Unified Model

The framework we use in this study is based on the Unified Model for Multi-objective Evolutionary Algorithms (UMMEA [9]), which is characterized by a decomposition of algorithms into abstract operators. Existing techniques can be classified, formalized, and mapped onto the operators. Hence, it is not only possible to emulate the behavior of specific algorithms, but also to combine the different techniques arbitrarily, which is important for our statistical analysis.

The main concept of this model is the coexistence of a 'normal' offspring population ($B, |B^t| = b$) and an archive of 'elite' individuals ($A, |A^t| \leq a$), which represents the current approximation of the Pareto set. All modifications of these multi-sets of individuals can be expressed by (stochastic) operators that can be arbitrarily combined in a modular fashion. Fig. 2 shows the generic algorithm based on this model (left) and a schematic view of the transition of one generation to the other (right). The $initialize$ operator usually sets the archive A^0 to the empty set and B^0 to a random sample of the search space. If the archive size is bounded ($1 \leq a < \infty$), a strategy must be defined to deal with more non-dominated solutions than can be stored. This is done by the $truncate$ operator.

The 'elitism intensity' is modeled by the parameter p_e, which represents the probability to select a parent from the archive instead of the normal population.

As long as the termination predicate is not satisfied, the archive and the population are evolved. In each iteration, the $update$ operator updates the archive with individuals from the current offspring population; usually, only non-dominated solutions are stored. In the $adapt$ operator, possible control mechanisms for the elitism intensity can be modeled, but in this study we leave p_e constant during the run.

$t := 0$
$(A^0, B^0, p_e^0) := initialize()$
while $terminate(A^t, B^t, t) = false$ **do**
  $t := t + 1$
  $A^t := truncate(update(A^{t-1}, B^{t-1}))$
  $p_e^t := adapt(A^t, B^{t-1}, p_e^{t-1})$
  $B^t := vary(sample(evaluate(A^t, B^{t-1}, p_e^t)))$
end while

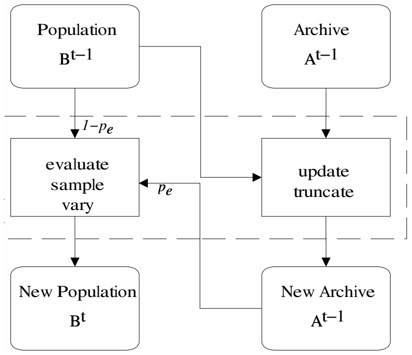

Fig. 2. Left: The general algorithm based on the unified model, A^t denotes the archive, B^t the population and p_e^t the elitism intensity at generation t. Right: A schematic view of the transition from one generation to the other.

The $evaluate$ operator calculates the target sampling rates for each individual depending on the individuals in the archive and the population. For example, the primary fitness of individuals is derived from their rank with respect to their dominance level, usually referred to as 'non-dominated sorting' [15]. From these ranks, the sampling rates of an individual b in a set of individuals (population) B are calculated to simulate binary tournament selection.

According to these sampling rates, the $sample$ operator then selects the required number of individuals from both the archive and the population as parents of the next generation. These parents are finally modified in the $vary$ operator, where both recombination and mutation take place. In this study we use use one-point crossover and constant mutation rates σ_{mut}.

3 Experimental Design

Our aim is to investigate how the quality of the Pareto set approximation of the algorithms depends on certain algorithmic variables or parameters. This is a challenging task since the design space of the algorithm is huge and the performance indicators are noisy. In this study we pursue a novel approach which will be explained next, followed by a short descriptions of the test problem and the performance measure.

3.1 Methodology

In the design of an algorithm and specifically in the instances of our unified model, we face different types of design variables. In order to investigate the effect of these (independent) variables on the performance of the algorithm, we want to deal with all of them simultaneously in a common framework. For this purpose we distinguish

- ordinal, continuous variables (like the elitism intensity p_e)
- ordinal, discrete variables (like the population size)
- categorical variables (like different operators)

Categorical variables are mainly given by different operator instances, e. g. the archive limitation strategy in $truncate$.

For each experiment, we first decide on the variable(s) whose influence is to be assessed, and fix all other variables to some standard or optimal value from previous experiments (if those are available). The free variables are then varied randomly in their respective domain, and from each setting one observation of the performance indicator is drawn, i. e. the EA run once with with each setting.[1]

After a large number of data has been collected in this manner (usually 1000 observations per experiment), we look for a linear model that yields a good description of the data. The general form of this model for the i-th observation is

$$\mathcal{V}_i = a_0 + a_1 \cdot x_i^{(1)} + a_2 \cdot x_i^{(2)} + \ldots + a_n \cdot x_i^{(n)} + E_i \quad (1)$$

where $x^{(1)}, \ldots, x^{(n)}$ are the explanatorial variables and $\mathcal{V}_i$ is the response (or target variable) – in our case the performance value. E_i are the random error caused by the randomness in the algorithm itself.[2]

The variables $x^{(j)}, 1 \leq j \leq n$ of the model can be any transformation or combination of original variables. Thus, the model is only linear in the coefficients, and non-linear dependencies from the variables (which are very likely for complex systems like evolutionary algorithms) can easily be traced. It furthermore allows to include the categorical variables, which have no natural value or order. This is done by indicator variables, where one variable is used for each category. If an observation falls into a specific category, the respective indicator variable is set to one, otherwise to zero. The coefficient of this indicator variable then shows the difference in the response that is caused by this category.

If only categorical variables were used, this method would be equivalent to analysis of variance (ANOVA), which has been applied to parameter studies of evolutionary algorithms by Schaffer et at. [14]. Here, we try to keep variables in their respective domain and want to use the order of the variables wherever possible. From the models we can then identify, which variables significantly effects the algorithmic performance.

3.2 Multi-objective 0/1 Knapsack Problems

To study the performance of different algorithmic configurations the multi-objective 0/1 knapsack problem is used, which has been subject to recent empirical case studies, both in the evolutionary computation community [19, 8] and in the field of multiple criteria decision analysis [16].

The multi-objective 0/1 knapsack problem is a function of the form $k : \{0,1\}^n \mapsto \mathbb{R}^m$, where n is the number of decision variables and m the number of objectives. In this work we refer to the definition in [19] and use the same parameters for the weights

[1] We prefer this over doing replications with identical settings for it leads to a better distribution of samples in the design space of the algorithm.

[2] In order for a linear regression to be viable, the random errors are assumed to be independent and identically distributed by a normal distribution with zero mean and equal variance. In this study we verified these assumptions using graphical diagnosis tools for the residuals $R_i = V_i - \tilde{\mathcal{V}}_i$, like Normal plot and Tukey-Anscombe plot. Generally, an appropriate transformation of the target variable $\mathcal{V}$ led to the desired results.

and profits of the items and the same constraint handling technique. Here we restrict ourselves to the bi-objective case as a baseline for further comparison, an extension of the experiments to $m > 2$ is straightforward. The number of decision variables and the allowed number of objective function calls is depicted in Table 1. Each individual represents one decision alternative as a bit-sting of length n.

3.3 Performance Measure

For the quality or performance measure $\mathcal{V}$ we apply a volume based approach according to [20] with slight modifications. Here, a reference volume between the origin and an utopia point – defined by the profit sums of all items in each objective – is taken into account. The aim is to minimize the fraction of that space, which is not dominated by any of the final archive members. We consider this as the most appropriate scalar indicator since it combines both the distance of solutions (towards some utopian trade-off surface) and the spread of solutions.

4 Elitism

This section focuses on the influence of elitism and its possible interdependence with the mutation intensity. To isolate these effects, no truncation of the archive ($a = \infty$) and no density dependent selection is applied, i.e. the $evalutate$ operator assigns sampling rates only on the basis of the dominance level.

Here, elitism in the sense of [9] is assumed, i. e. the best individuals are not only stored permanently, they also take part in the selection of offspring. Our previous study indicated that there may be interaction between the mutation strength and the benefit of elitism. As in the original version of UMMEA, the elitism intensity is characterized by a parameter $p_e \in [0, 1]$ which basically represents the probability to select a parent individual from the archive instead of the previous offspring population. This concept allows us to emulate the different elitist multi-objective evolutionary algorithms which have been proposed so far.

4.1 Experiments

In the first experiment, we vary p_e randomly and uniformly in $[0, 1]$ and the mutation rate between $1/n$ and $10/n$. The regression analysis shows significant interactions between the elitism intensity p_e and the normalized mutation rate $\sigma = \sigma_{mut} \cdot n$, where σ_{mut} denotes the mutation probability for each of the n bits. σ now gives the expected number of changed bits per mutation of an individual. The interactions are clearly visible in Fig. 4: For low mutation rates $\sigma < 2$, the performance decreases monotonously with p_e, while for $\sigma > 5$ the performance increases with p_e. The region around the critical mutation rate $\sigma \approx 3$ has the lowest variance in performance. However, the best results are achieved on either sides: There is a local optimum at $\sigma \approx 1$ and $p_e \in [0, 0.1]$ and a global optimum around $\sigma \approx 6$ and $p_e = 1$, while their difference seems to increase with n (see Fig. 4). These results show that elitism is not always beneficial, it

Knapsack Problem	KN1	KN2	KN3
Decision variables n	250	500	750
Maximal objective function calls	40,000	80,000	120,000
Standard population size b	150	200	250

Table 1. Different instances of the knapsack problem and default parameters for this study.

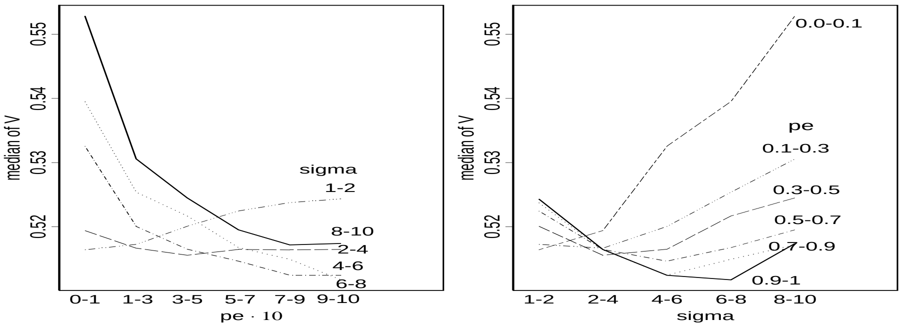

Fig. 3. Interaction plots for the combined influence of the mutation rate σ and the elitism intensity p_e on $\mathcal{V}$ for the knapsack problem with $n = 750$. The median for each combination is shown.

can even be harmful, if the mutation rate is too low, which is surprisingly the case for the popular rule $\sigma_{mut} = 1/n$. Nevertheless, the best results are achievable for $p_e = 1$.

At this point we refer to two other studies, which compared different algorithms on the same test problems. In [20], a constant mutation rate of $\sigma_{mut} = 0.01$ was used for all algorithms, and the advantage of the elitist SPEA (Strength Pareto EA, [20]) became more visible with increasing problem size. This may be explained as follows: A mutation rate of 0.01 leads for $n = 250$ to $\sigma = 2.5$, which is still not high enough to make the influence of elitism strong. For $n = 500$ and $n = 750$, however, the values are $\sigma = 5$ and $\sigma = 7.5$, respectively, and thereby in a range where elitism is very important. In [8] the performance of SPEA was improved by increasing the archive size relative to the population and (due to the specific selection method used) implicitly the elitism intensity from 0.36 to 0.96. According to our findings this is crucial especially when a mutation rate of 0.01 is used on the knapsack problem with $n = 750$.

These observations support the assumption that non-elitist EA only work well, if the mutation strength lies under a certain threshold [11]. Otherwise selection is no longer able to compensate the low success rate of mutation to create dominating individuals, which usually decreases as the mutation rate increases. For multi-objective problems, where the success rate also decreases with the number of objectives, this can pose a further problem: For high dimensional objective spaces, no mutation rate may exist for which non-elitist algorithms lead to positive progress, unless the population size is very large and a high selection pressure is used. On the other hand, elitist EAs have a significantly higher optimal mutation rate, which in turn increases with the elitism intensity. Here, the archived solutions preclude divergence.

4.2 Key results

The main observation is that the usefulness of elitism strongly depends on the mutation strength:

- If the usually recommended mutation rate $\sigma = 1$ is taken, the best performance is achieved without elitism (p_e=0).
- The best overall performance can be reached with strong elitism ($p_e > 0.7$) in combination with high mutation rates ($\sigma \approx 5$).
- The combination of strong elitism with low mutation rates or weak elitism with high mutation rates shows the worst performance.

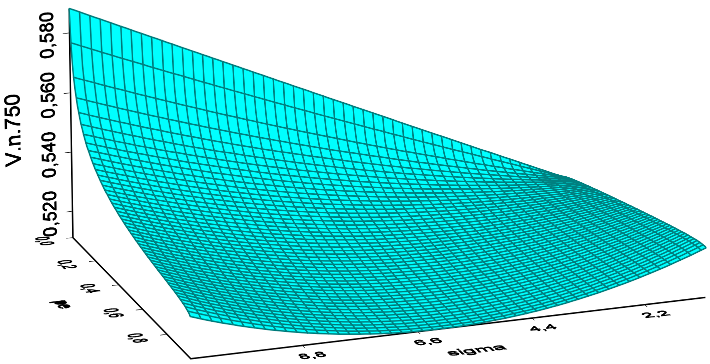

Fig. 4. Estimated response surface of the performance measure $\mathcal{V}$ to the normalized mutation rate σ and the elitism intensity p_e for $n = 750$. Two local minima arise at $(p_e = 1, \sigma \approx 6)$ and $(p_e = 0, \sigma = 1)$. A third order orthogonal polynomial was used as the model, the coefficients were estimated via linear regression.

5 Bounding the Archive Size

In the previous section the influence of the mutation rate and the elitism intensity has been explored under the assumption that the archive is unbounded and can contain all non-dominated individuals found so far. In some cases, however, it may be desirable – or even mandatory – to limit its size for several reasons:

- The size of the true non-dominated set of the multi-objective problem may be exponentially large or even infinite.
- All implementations are restricted to limited resources (i. e. storage space).
- The complexity of the archive updating operator increases with the archive size.
- Genetic drift can occur since over-represented regions of the search space are favored in the (uniform) sampling process.

While the first three points mean that one has to limit the archive size for practical considerations though it would ideally be unlimited, the last point indicates that a (useful) truncation of the archive may also lead to a performance gain. In the following we set the maximum archive size $a := b$ and examine how different archive truncation methods affect the algorithmic performance.

5.1 Truncation Operators

Rudolph and Agapie [13] provide theoretical results about convergence properties of different archiving and selection strategies for multi-objective optimizers in finite search spaces. The authors state that algorithms with unlimited archive sizes do have the desired convergence properties provided that the variation operators match certain preconditions. However, they are usually not of practical relevance because of limited resources. Instead, they propose an updating operator that respects a maximum archive size via a strong elite preserving strategy, thus keeping the desired convergence properties. In our experiment this operator is named $truncate_1$, and it assures that for each dominated former archive member, at least one dominating individual must be included in the new archive. In contrast, the operator $truncate_2$ just makes a random choice which individuals to delete.

Another possibility to reduce the archive size is clustering: The possible members of the archive are grouped into distinct clusters based on some similarity measure. Then, each cluster will be represented in the archive by certain individuals. Clustering-based approaches are not strongly elite preserving, and they can be very time consuming. Many algorithms are based on iterative melioration of a given partitioning according to a predefined necessary optimality condition and can therefore lead to partitions which are only locally optimal. Clustering-based archive reduction is used in SPEA [20]: The individuals are clustered by the average linkage method into a distinct clusters, from which the centroid individual is included in the new archive. Here, we implement this as the $truncate_3$ operator.

Other approaches to limit the archive size can roughly be categorized as density dependent ranking techniques. We will discuss density estimation later in the context of biasing selection. The idea for this concept is quite intuitive: Though the archive must be truncated, one would like it to be as 'diverse' as possible. Hence, individuals in a densely populated area receive lower values and are discarded from the archive in favor of others. Different implementations of this concept are applied in [12], [7], [8], [2], or [3]. In this study we represent the method used in NSGA-II [3] by the operator $truncate_4$: For each objective coordinate the absolute difference of its predecessor and successor is aggregated for each individual, higher total values lead to better ranks.

Table 2 gives an overview of these different techniques and the implementations we use in our experiments. As a baseline, the $truncate_0$ operator is included, which represents the unlimited archive.

Method	No Reduction	Conservative	Random	Clustering	Density-based
Operator	$truncate_0$	$truncate_1$	$truncate_2$	$truncate_3$	$truncate_4$
Examples	VV [13] PR [13]	AR-1 [13] AR-2 [13]		SPEA [20]	PAES [7] M-PAES [8] PESA [2] NSGA-II [3]
Features	may grow very large, genetic drift	efficiency preserving, unreachable points	easy implementation, low complexity, genetic drift	good discrimination, adaptive metrics, high complexity	good discrimination, medium complexity

Table 2. Archive truncation methods in multi-objective evolutionary algorithms and operator instances for this study.

5.2 Experiments

At first two experiments were carried out where the normalized mutation rate was fixed at $\sigma = 1$ and $\sigma = 4$, respectively, while for each trial p_e was again varied randomly in $[0, 1]$ and one of the four $truncate$ operators was picked with equal probability.

For $\sigma = 4$ the $truncate$ operator had no significant effect on the results at all. This is not surprising since in these cases the archive size rarely exceeds a. For the small mutation rate $\sigma = 1$, where we had very large archives in the experiments before, the

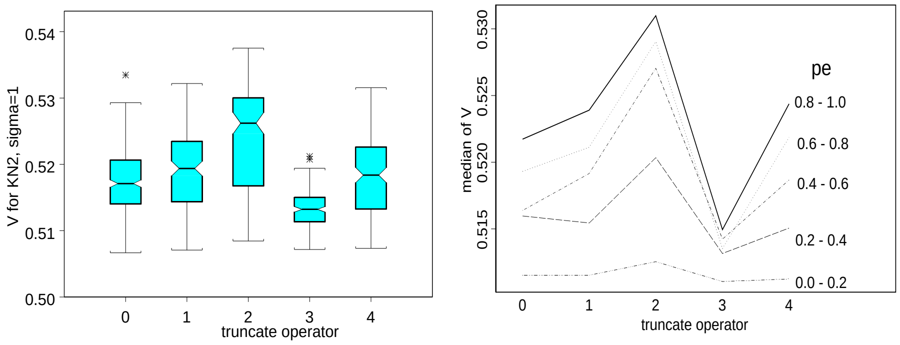

Fig. 5. Left: Box plots for $n = 500$, $\sigma = 1$ and different $truncate$ operators. The boxes range from the lower to the upper quartile of the observations, the median is marked with a bold line, outliers with an asterisk. The notches show confidence intervals for the group medians, by not overlapping they indicate a difference in location on a rough 5% significance level. Right: Interaction plots for the combined influence of the $truncate$ operator and the p_e level on $\mathcal{V}$. The median for each combination is shown.

effect of the $truncate$ operator becomes significant. Fig. 5 (left) shows the box plots for the different operators. However, there is a strong interaction with the elitism intensity: The effect of the $truncate$ operator increases with p_e. Once again, if the archive is not used in the reproduction of offspring ($p_e = 0$), the effect of the $truncate$ operator is the weakest, this can be visualized by the interaction plots in Fig. 5 (right).

As a result we can claim that, if a reduction of the archive is necessary, is should be done carefully to minimize information loss. The random choice is always worst. In no case was the clustering approach significantly inferior to the unbounded archive, while it was the best for large p_e and low $\sigma = 1$. This shows that a 'good' reduction method is able to bias the search in favor of under-represented regions. Surprisingly, the density-based method does not reach significantly higher $\mathcal{V}$ values than the conservative method. The reason may be found in the specific technique, which has difficulties to deal with identical objective vectors, and not in this type of reduction method.

Nonetheless, there are other methods which directly influence the sampling rate of individuals based on their density, these will be discussed in the next section. One of the questions will be whether density dependent selection itself will lead to a higher performance gain than archive reduction.

5.3 Key Results

Depending on the choice of the truncation operator, archive reduction can either decrease or increase the performance:

– The clustering-based method leads to an increase of the average performance over the whole range of p_e and σ values. In other words: Clustering reduces the algorithm's sensitivity to suboptimal parameter settings without, however, a further improvement for the optimal settings.
– Random truncation lead to a significantly worse performance than the conservative method, which in turn was not significantly worse than the unbounded archive.

6 Density Dependent Selection

In multi-objective optimization, a uniform distribution of efficient points may be desirable in general. Unfortunately, the randomness in the evolutionary operators (genetic drift), the granularity of the search space or the topology of the objective function can make the archived solutions as well as the population itself exhibit a rather non-uniform distribution. A uniform sampling from the archive even reinforces these effects: Over-represented regions will be sampled by parents more often, and - given at least a certain locality of the variation operators - more offspring will be produced there.

One way to tackle this problem indirectly can be through a reduction of the archive, as described in the last section. A more direct approach, however, would be to bias the selection process in favor of under-represented regions. This will then be independent of the actual archive size and is much more flexible.

6.1 Density Estimation Methods

The density of individuals in a set A can serve as an a posteriori estimate of the probability density for the creation of the individuals in this set. This probability distribution is usually implicitly defined by the stochastic process which governs the evolutionary algorithm. However, it can easily be estimated (using standard probability density estimation techniques) and then be used to bias the sampling rates accordingly.

The relevance of density estimation in the context of (multi-objective) evolutionary algorithms has been put forward by [5], where the authors noted that the standard fitness sharing concept is essentially the application of a kernel density estimator. In [4], existing results from kernel density estimation were used to derive guidelines for the fitness sharing parameters.

Many advanced multi-objective evolutionary algorithms use some form of density dependent selection. Furthermore, nearly all techniques can be expressed in terms of density estimation, a classification is given in Table 3. We will make use of this as a further step towards a common framework of evolutionary multi-objective optimizers, and present the relevant enhancement of the unified model.

A straightforward density estimate is the histogram: The (multi-variate) space is divided into equally sized cuboids, and the number of points inside a cuboid is the estimate of the density for this subspace. Here we apply this technique in the $evaluate_1$ operator. Kernel density estimation is represented in $evaluate_2$ by the niche count function known from standard fitness sharing. The $evaluate_3$ operator uses the technique from NSGA-II, where the distances to neighboring individuals is used to calculate a volume around each individual. The reciprocal value of this volume then serves as an estimate of the density, which is typical for nearest neighbor density estimates.

For the calculation of the target sampling rates, the normalized density estimate for each solution is added to its basic cost. If the minimal basic cost difference of differently valued solutions is at least 1 (like in our case of the dominance level), it will be assured that lower valued individuals always receive lower sampling rates regardless of their density. Thus, this method is used to bias selection *only between equally valued individuals*. Finally, a rank based assignment of target sampling rates is performed as before.

Method	None	Histogram	Kernel	Nearest Neighbor
Operator	$evaluate_0$	$evaluate_1$	$evaluate_2$	$evaluate_3$
Examples		PAES M-PAES PESA	all algorithms with fitness sharing	NSGA-II
Features:				
continuous		no	as kernel function	no
smoothing control		bin width	window width	local density

Table 3. Density estimation techniques in multi-objective evolutionary algorithms and operators used in this study.

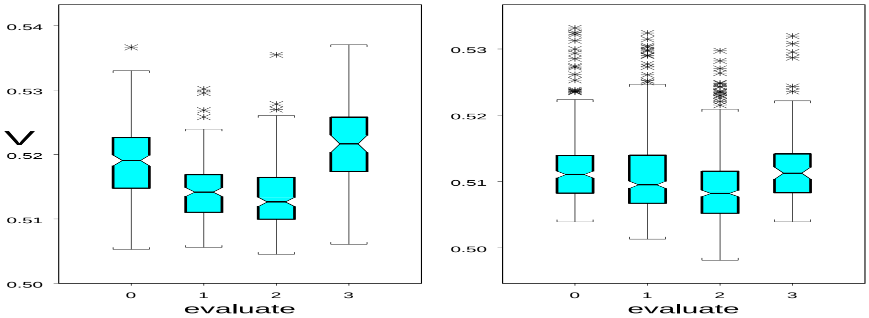

Fig. 6. Box plots for $n = 500$, $\sigma = 1$ (left) and $\sigma = 4$ (right) for different $evaluate$ operators. For $n = 250$ and $n = 750$ the plots a similar.

6.2 Experiments

In order to first investigate the effect of the density estimation alone, the elite preserving $truncate_1$ operator is used[3] and the maximal archive size is set to $a := b$. $\sigma \in [1, 10]$ and $p_e \in [0, 1]$ are again chosen randomly. The $evaluate_0$ operator is the reference case, where no density estimation is applied.

In contrast to the previous section, the influence of the $evaluate$ operator is now significant for the whole range of $\sigma \in [1, 10]$. Figure 6 shows the box plots, again for the medium size problem with $n = 500$. Notice that for $\sigma = 1$ the location differences of the $\mathcal{V}$ distribution between the first three operators are stronger than for $\sigma = 4$, while for $\sigma = 4$ the differences in the minimal $\mathcal{V}$ values are stronger.

Fig. 7 visualizes how the behavior of the different $evaluate$ operators interacts with different p_e and σ settings. Obviously, the kernel density estimator as well as the histogram method lead to lower $\mathcal{V}$ values in all groups. The NSGA-II based method, however, does not change performance at lower mutation rates and only improves a few groups with stronger mutation. The corresponding fronts show that the performance gain of density dependent selection is only due to a broadening of the current non-dominated set, while approximation does not come closer in the middle. Comparisons to the real efficient set show that the evolutionary algorithm still finds solutions only in the central region of the real efficient set, even with the density dependent selection operators considered here.

[3] We chose this as the common baseline since it did not show significant differences to the unlimited archive.

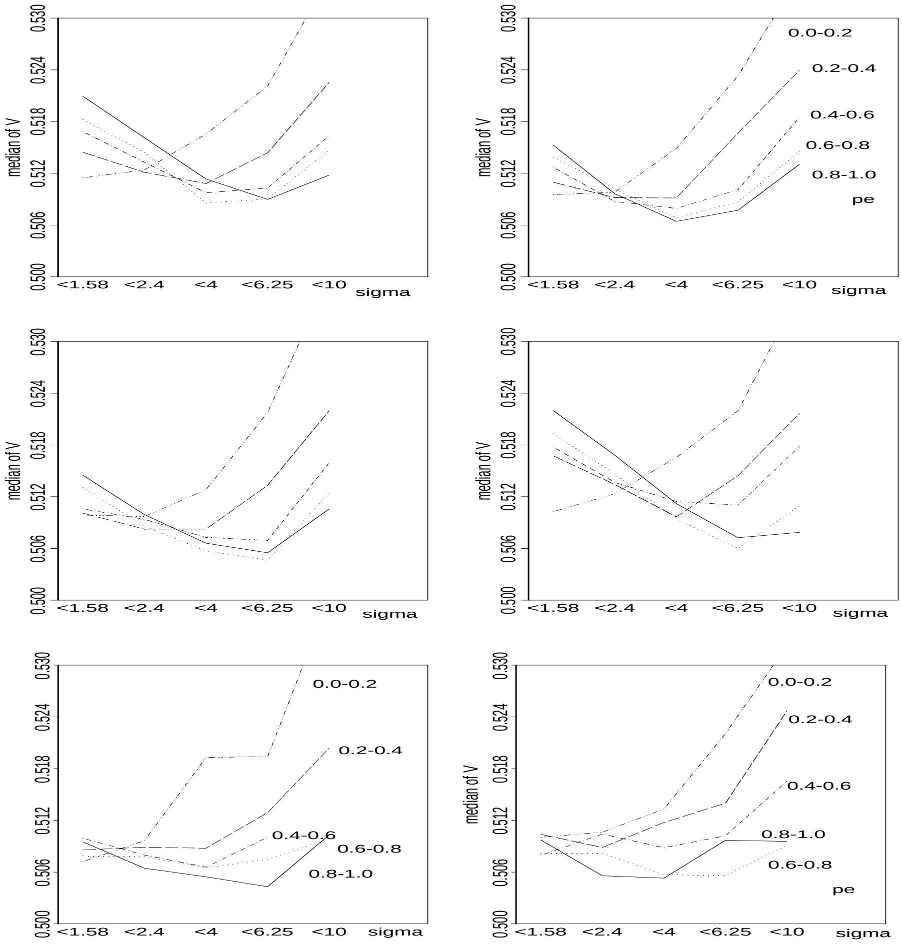

Fig. 7. Interaction plots for $n = 500$ for different $evaluate$ operators. The upper row shows no density estimation (left) and the histogram method (right), the middle row the kernel density estimator (left) and the NSGA-II-based method (right), the lower row the clustering based truncation operator combined with the kernel density estimator (left) and the histogram method (right).

Finally, the different density estimation techniques are combined with the clustering based $truncate$ operator. Though this does not improve much on the top performance, the lower row of Fig. 7 shows that the truncation method reduces the sensitivity to suboptimal parameter settings, especially for low mutation rates.

6.3 Key Results

The performance gain of density dependent selection strongly depends on the accuracy of the applied density estimator:

- In general, the kernel density estimator as well as the histogram method improves the performance for all p_e and mutation rate settings.

– The combination of these techniques with the clustering-based archive truncation method leads to synergy effects in the sense that the algorithm becomes even less sensitive to suboptimal parameter settings than without density based selection.

7 Conclusion and Outlook

In this study we identified and analyzed the key elements of evolutionary multi-objective optimizers. From the results we can derive some design guidelines:

– Strong elitism should be used to achieve best performance, but in connection with a high mutation rate. The right combination of elitism intensity and mutation rate is the decisive factor for the performance.
– Density based selection can further improve the algorithmic performance by a broader distribution of solutions along the trade-off surface. Here, enough effort should be put in the density estimation technique since only a good estimate brings forth the desired results; fitness sharing as a simple kernel density estimator is a good choice.
– A good archive reduction method like clustering should be incorporated to make the algorithm robust concerning suboptimal parameter settings. However, the truncation operator has to be chosen carefully as inappropriate methods can decrease the performance.

As for the methodology of comparing different algorithms it should be noted that

– In comparisons of elitist against non-elitist algorithm not only the influence of the mutation rate must be considered but also the intensity of the elitism, which is often defined implicitly.
– When new algorithms are developed, the effect of all new features should be examined separately.

At present, these results are only valid for the bi-objective knapsack problem, but we believe that some fundamental characteristics have been found. It is subject to ongoing research to verify this for other problem classes and higher objective space dimensions. It can be expected, however, that performance differences increase, when more objectives and more difficult problems are considered.

References

1. Carlos A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization. *Knowledge and Information Systems*, 1(3):269–308, 1999.
2. D. W. Corne, J. D. Knowles, and M. J. Oates. The pareto envelope-based selection algorithm for multiobjective optimisation. In Marc Schoenauer et al., editor, *Parallel Problem Solving from Nature – PPSN VI*, Berlin. Springer.
3. K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In Marc Schoenauer et al., editor, *Parallel Problem Solving from Nature – PPSN VI*, Berlin. Springer.

4. C. M. Fonseca and P. J. Fleming. Multiobjective genetic algorithms made easy: Selection, sharing and mating restrictions. In *First Int'l Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA 95)*, pages 45–52, London, UK, 1995. The Institution of Electrical Engineers.
5. C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.
6. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts, 1989.
7. J. D. Knowles and D. W. Corne. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In *Congress on Evolutionary Computation (CEC99)*, volume 1, pages 98–105, Piscataway, NJ, 1999. IEEE Press.
8. J. D. Knowles and D. W. Corne. M-PAES: A memetic algorithm for multiobjective optimization. In *Congress on Evolutionary Computation (CEC 2000)*, volume 1, pages 325–332, Piscataway, NJ, 2000. IEEE Press.
9. M. Laumanns, E. Zitzler, and L. Thiele. A unified model for multi-objective evolutionary algorithms with elitism. In *Congress on Evolutionary Computation (CEC 2000)*, volume 1, pages 46–53, Piscataway, NJ, 2000. IEEE Press.
10. S. Obayashi, S. Takahashi, and Y. Takeguchi. Niching and elitist models for MOGAs. In A. E. Eiben et al., editor, *Parallel Problem Solving from Nature – PPSN V*, pages 260–269, Berlin, 1998. Springer.
11. G. Ochoa. Consensus sequence plots and error thresholds: Tools for visualising the structure of fitness landscapes. In M. Schoenauer et al., editor, *Parallel Problem Solving from Nature – PPSN VI*, pages 129–138, Berlin, 2000. Springer.
12. G. T. Parks and I. Miller. Selective breeding in a multiobjective genetic algorithm. In A. E. Eiben et al., editor, *Parallel Problem Solving from Nature – PPSN V*, pages 250–259, Berlin, 1998. Springer.
13. G. Rudolph and A. Agapie. Convergence properties of some multi-objective evolutionary algorithms. In *Congress on Evolutionary Computation (CEC 2000)*, volume 2, pages 1010–1016, Piscataway, NJ, 2000. IEEE Press.
14. J. D. Schaffer, R. A. Caruana, L. J. Eshelman, and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In J. D. Schaffer, editor, *Proceedings of the third international conference on genetic algorithms*, pages 51–60, San Mateo, CA, 1989. Morgan Kaufmann.
15. N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.
16. E.L. Ulungu, J. Teghem, P.H. Fortemps, and D. Tuyttens. Mosa method: A tool for solving multiobjective compinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8(4):221–236, 1999.
17. D. A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Graduate School of Engineering of the Air Force Institute of Technology, Air University, June 1999.
18. E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.
19. E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms — a comparative case study. In Agoston E. Eiben et al., editor, *Parallel Problem Solving from Nature – PPSN V*, pages 292–301, Berlin, 1998. Springer.
20. 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.