# Multi–objective Optimization with the Naive MIDEA

Peter A.N. Bosman[1] and Dirk Thierens[2]

[1] National Research Institute for Mathematics and Computer Science P.O. Box 94079 1090 GB Amsterdam, The Netherlands
Peter.Bosman@cwi.nl

[2] Institute of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
Dirk.Thierens@cs.uu.nl

**Summary.** EDAs have been shown to perform well on a wide variety of single-objective optimization problems, for binary and real-valued variables. In this chapter we look into the extension of the EDA paradigm to multi-objective optimization. To this end, we focus the chapter around the introduction of a simple, but effective, EDA for multi-objective optimization: the naive MIDEA (mixture-based multi-objective iterated density-estimation evolutionary algorithm). The probabilistic model in this specific algorithm is a mixture distribution. Each component in the mixture is a univariate factorization. As will be shown in this chapter, mixture distributions allow for wide-spread exploration of a multi-objective front, whereas most operators focus on a specific part of the multi-objective front. This wide-spread exploration aids the important preservation of diversity in multi-objective optimization. To further improve and maintain the diversity that is obtained by the mixture distribution, a specialized diversity preserving selection operator is used in the naive MIDEA. We verify the effectiveness of the naive MIDEA in two different problem domains and compare it with two other well-known efficient multi-objective evolutionary algorithms (MOEAs).

## 1 Introduction

In this chapter, we apply the EDA paradigm to multi-objective optimization. We put the focus on a specific EDA, which we call the naive mixture-based multi-objective iterated density-estimation evolutionary algorithm (naive MIDEA). The naive MIDEA is an instance of the MIDEA framework for multi-objective optimization using EDAs. We will show how the naive MIDEA can be implemented for both binary as well as real problem variables.

The remainder of this chapter is organized as follows. In Sect. 2, we first discuss multi-objective optimization. In Sect. 3 we develop the MIDEA framework and specifically focus on the naive MIDEA instance. In Sect. 4 we validate the performance of MIDEAs on eight test problems and compare the results with two

P.A.N. Bosman and D. Thierens: *Multi–objective Optimization with the Naive* MIDEA, StudFuzz **192**, 123–157 (2006)
www.springerlink.com

other state-of-the-art MOEAs and discuss our findings. We present our conclusions in Sect. 5.

## 2 Multi-objective Optimization

Multi-objective optimization differs from single-objective optimization in that we have a multiple of objectives that we wish to optimize simultaneously without an expression of weight or preference for any of the objectives. Often, these multiple objectives are conflicting. Such problems naturally arise in many real world situations. An example of conflicting objectives that often arises in industry, is when we want to minimize the costs of some production process while at the same time we also want to minimize the pollution caused by the same production process. Such conflicting objectives give rise to a key characteristic of multi-objective optimization problems, which is the existence of sets of solutions that cannot be ordered in terms of preference when only considering the objective function values simultaneously. To formalize this notion, four relevant concepts exist. Assuming that we have $m$ objectives $f_i(\boldsymbol{x}), i \in \boldsymbol{\mathcal{M}} = \{0, 1, \ldots, m-1\}$, that, without loss of generality, we seek to minimize, these four concepts can be defined as follows:

1. **Pareto dominance**
   A solution $\boldsymbol{x}$ is said to (Pareto) dominate a solution $\boldsymbol{y}$ (denoted $\boldsymbol{x} \succ \boldsymbol{y}$)
   **iff** $(\forall i \in \boldsymbol{\mathcal{M}} : f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y})) \wedge (\exists i \in \boldsymbol{\mathcal{M}} : f_i(\boldsymbol{x}) < f_i(\boldsymbol{y}))$
2. **Pareto optimal**
   A solution $\boldsymbol{x}$ is said to be Pareto optimal **iff** $\neg\exists \boldsymbol{y} : \boldsymbol{y} \succ \boldsymbol{x}$
3. **Pareto optimal set**
   The set $\boldsymbol{\mathcal{P}_S}$ of all Pareto optimal solutions: $\boldsymbol{\mathcal{P}_S} = \{\boldsymbol{x} | \neg\exists \boldsymbol{y} : \boldsymbol{y} \succ \boldsymbol{x}\}$
4. **Pareto optimal front**
   The set $\boldsymbol{\mathcal{P}_F}$ of all objective function values corresponding to the solutions in $\boldsymbol{\mathcal{P}_S}$: $\boldsymbol{\mathcal{P}_F} = \{(f_0(\boldsymbol{x}), f_1(\boldsymbol{x}), \ldots, f_{m-1}(\boldsymbol{x})) | \boldsymbol{x} \in \boldsymbol{\mathcal{P}_S}\}$

The Pareto optimal set $\boldsymbol{\mathcal{P}_S}$ is a definition of all trade-off optimal solutions in the parameter space. The Pareto optimal front $\boldsymbol{\mathcal{P}_F}$ is the same set of solutions, only regarded in the objective space. The size of either set can be infinite, in which case it is impossible to find the optimal set or front with a finite number of solutions. Regardless of the size of $\boldsymbol{\mathcal{P}_S}$ or $\boldsymbol{\mathcal{P}_F}$, it is commonly accepted that we are interested in finding a good representation of these sets with a finite number of solutions. The definition of a good representation, is difficult however. The reason for this is that it is desirable to obtain a diverse set of solutions as well as it is desirable to obtain a front or set that is close to the optimal one. Furthermore, it depends on the mapping between the parameter space and the objective space whether a good spread of the solutions in the parameter space is also a good spread of the solutions in the objective space. However, it is common practice [9] to search for a good diversity of the solutions along the Pareto *front*. The reason for this is that a decision-maker will ultimately have to pick a single solution. Therefore, it is often best to present a wide variety of trade-off solutions for the specified goals.

The notion of searching a space by maintaining a population of solutions is characteristic of evolutionary algorithms (EAs), which makes them natural candidates for multi-objective optimization aiming to cover a good approximation of the Pareto optimal front. A strongly increasing amount of research has indeed been done in the field of evolutionary multi-objective optimization in recent years [9] with very promising results.

## 3 The Naive 𝕄IDℍA

To obtain EDAs that are well-suited for multi-objective optimization, we propose to instantiate two steps in the framework. Firstly, to stimulate the preservation of diversity along the Pareto front, we instantiate the selection mechanism by using a diversity preserving truncation selection operator. Secondly, we partially instantiate the search for a probability distribution to use by enforcing the use of mixture distributions.

### 3.1 Diversity-preserving Truncation Selection

#### Background and Motivation

Selection in evolutionary algorithms is meant to select the better solutions of the population to perform variation with. In multi-objective optimization however, the notion of "a better solution" has two sides to it. On the one hand we want the solutions to be as close to the Pareto optimal front as possible. On the other hand, we want a good diverse representation of the Pareto optimal front. A good selection operator in a MOEA must thus exert selection pressure with respect to both of these aspects.

*Selection Pressure towards the Pareto Optimal Front*

In a practical application, we have no indication of how close we are to the Pareto optimal front. To ensure selection pressure towards the Pareto optimal front in the absence of such information, the best we can do is to find solutions that are dominated as little as possible by any other solution.

A straightforward way to obtain selection pressure towards non-dominated solutions is therefore to count for each solution in the population the number of times it is dominated by another solution in the population, which is called the domination count of a solution [3, 16]. The rationale behind the domination count approach is that ultimately we would like no solution to be dominated by any other solution, so the less times a solution is dominated, the better. A lower domination count is preferable. Using this value we can apply truncation selection or tournament selection to obtain solid pressure towards non-dominated solutions.

Another approach to ensuring a preference for solutions that are dominated as little as possible, is to assign a preference to different domination *ranks* [12, 17].

The solutions that are in the $j^{\text{th}}$ rank are those solutions that are non-dominated if the solutions of all ranks $i < j$ are disregarded. Note that the best domination rank contains all solutions that are non-dominated in the complete population. A lower rank is preferable. Using this value we can again apply for instance either truncation selection or tournament selection. Similar to the domination count approach, this approach effectively prefers solutions that are closer to the set of non-dominated solutions. It has been observed that in practice the difference between domination-counting and the domination-ranking schemes in practice is only very small [5].

*Selection Pressure towards Diversity*

In most multi-objective selection schemes, diversity is used as a second comparison key in selection. This prohibits tuning the amount of selection pressure towards diversity to the amount of selection pressure towards getting close to the Pareto optimal front. An example is the approach taken in the NSGA-II in which solutions are selected based on their non-domination rank using tournament selection [12]. If the ranks of two solutions are equal, the solution that has the largest total distance between its two neighbors summed over each objective, is preferred. This gives a preference to non-crowded solutions.

The explicit selection pressure towards diversity may serve more than just the purpose of ensuring that a diverse subset is selected from a certain set of non-dominated solutions. If we only apply selection pressure to finding the non-dominated solutions and enable diversity preservation only to find a good spread of solutions in current Pareto front, we increase the probability that we only find a subset of a discontinuous Pareto optimal front. Selection pressure towards diversity will most likely be too late in helping out to find the other parts of the discontinuous Pareto optimal front as well. Therefore, we may need to spend more attention on diversity preservation during optimization and perhaps even increase the amount of diversity preservation. Another reason why we may need to increase the selection pressure towards diversity is that a variation operator is used that can find many more non-dominated solutions, which could cause a MOEA to converge prematurely onto subregions of a Pareto optimal front or onto locally optimal sets of non-dominated solutions, unless the population size is increased. However, given a fixed number of evaluations, this can be a significant drawback in approaching the Pareto optimal front. This problem can be alleviated by placing more emphasis on selection pressure towards diversity and by consequently reducing the effort in the selection pressure towards getting close to the Pareto optimal front. By doing so, the variation operator is presented with a more diverse set of solutions from which a more diverse set of offspring will result. Furthermore, solutions that are close to each other will now have a smaller joint chance that they will both be selected, which improves the ability to approach the Pareto optimal front since premature convergence is less likely.

*Combining Selection Pressures*

Concluding, to ensure pressure towards the Pareto optimal front and towards diversity at the same time, the selection procedure must be provided with a component

that prefers a diverse selection of solutions. However, since the goal is to preserve diversity *along* the Pareto front, rather than to preserve diversity in general, the selection on the basis diversity should not precede selection on the basis of getting close to the Pareto optimal front.

**Selection Operator**

In the selection operator that we propose, the ratio of the amount of selection pressure towards the Pareto optimal front and the amount of selection pressure towards diversity can be tuned using a single parameter $\delta$ to better fit the specific needs of the problem solver. In most selection operators this ratio is fixed beforehand. Ultimately, the selection operator selects $\lfloor \tau n \rfloor$ solutions, where $n$ is the population size and $\tau \in [\frac{1}{n}; 1]$ is the selection percentile. Just as there are two forms of selection pressure to be exerted by the selection operator as discussed above, there are two phases in our selection operator.

1. In the first phase, the domination count [16] of all solutions is first computed as mentioned above. Subsequently, a pre-selection $\boldsymbol{\mathcal{S}}^P$ is made of $\lfloor \delta\tau n \rfloor$ solutions ($\delta \in [1; \frac{1}{\tau}]$) using truncation selection on the domination count (select the best $\lfloor \delta\tau n \rfloor$ solutions). However, if the solution with the largest domination count to end up in $\boldsymbol{\mathcal{S}}^P$ by truncation selection has a domination count of 0, *all* solutions with a domination count of 0 are selected instead, resulting in $|\boldsymbol{\mathcal{S}}^P| \geq \lfloor \delta\tau n \rfloor$. This ensures that once the search starts to converge onto a certain Pareto front, we enforce diversity over all of the available solutions on the front.
2. In the second phase, the final selection $\boldsymbol{\mathcal{S}}$ is obtained from $\boldsymbol{\mathcal{S}}^P$. To do so, a nearest neighbor heuristic is used to promote diversity. First, a solution with an optimal value for a randomly chosen objective is deleted from $\boldsymbol{\mathcal{S}}^P$ and added to $\boldsymbol{\mathcal{S}}$. Note that the choice of objective is arbitrary as the key is to find a diverse selection of solutions. To stimulate this, we can select a solution that is optimal along any objective. For all solutions in $\boldsymbol{\mathcal{S}}^P$, the nearest neighbor distance is computed to the single solution in $\boldsymbol{\mathcal{S}}$. The distance that we use is the Euclidean distance scaled to the sample range in each objective. The solution in $\boldsymbol{\mathcal{S}}^P$ with the *largest* distance is then deleted from $\boldsymbol{\mathcal{S}}^P$ and added to $\boldsymbol{\mathcal{S}}$. The distances in $\boldsymbol{\mathcal{S}}^P$ are updated by investigating whether the distance to the newly added point in $\boldsymbol{\mathcal{S}}$ is smaller than the currently stored distance. These last two steps are repeated until $\lfloor \tau n \rfloor$ solutions are in the final selection.

An example application of this operator is presented in Fig. 1. This selection operator has a running time complexity of $\mathcal{O}(n^2)$. This is no worse than the minimum of $\mathcal{O}(n^2)$ for computing the domination counts which is required in all MOEAs.

### 3.2 Mixture Distributions

A mixture probability distribution is a weighted sum of $k > 1$ probability distributions. Each probability distribution in the mixture probability distribution is called

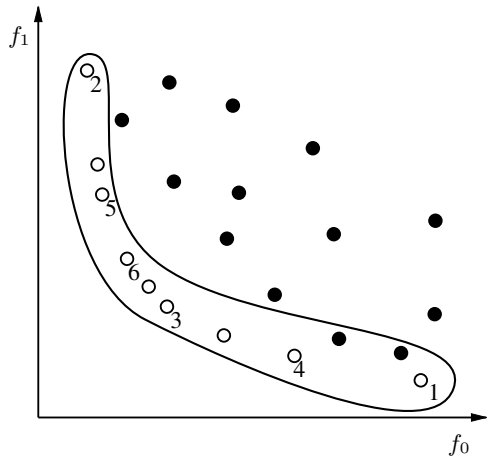


**Fig. 1.** An example of the application of the diversity preserving selection operator with $n = 22$, $\delta = \frac{5}{3}$, $\tau = \frac{3}{10}$, which gives $\lfloor \delta\tau n \rfloor = 11$ and $\lfloor \tau n \rfloor = 6$. Objectives $f_0$ and $f_1$ should both be minimized. The dominated solutions are black whereas the non-dominated solutions are white. The solutions that belong to the preselection are outlined. The solutions that are finally selected are numbered in the order in which they are chosen from the preselection. Here objective $f_0$ has been chosen to initiate the selection process

a mixture component. Let $\boldsymbol{\mathcal{Z}} = (Z_0, Z_1, \ldots, Z_{l-1})$ be a vector for all random variables involved in the EDA (i.e. $Z_i$ is a random variable associated with the $i^{\text{th}}$ problem variable). A mixture probability distribution for random variables $\boldsymbol{\mathcal{Z}}$ is then defined as follows:

$$P^{mixture}(\boldsymbol{\mathcal{Z}}) = \sum_{i=0}^{k-1} \beta_i P^i(\boldsymbol{\mathcal{Z}}) \tag{1}$$

where $\beta_i \geq 0$, $i \in \{0, 1, \ldots, k-1\}$, and $\sum_{i=0}^{k-1} \beta_i = 1$. The $\beta_i$ with which the mixture components are weighted in the sum are called mixing coefficients.

### The Benefit of Mixture Distributions

The general advantage of mixture probability distributions is that a larger class of independence relations between the random variables can be expressed than when using non-mixture probability distributions since a mixture probability distribution makes a combination of multiple probability distributions. In many cases, simple probability distributions can be estimated to get accurate descriptions of the data in different parts of the sample space. By adding the $k$ "simple" probability distributions into the mixture probability distribution, an accurate description of the data in the complete sample space can be obtained. This allows for the modelling of quite complex dependencies between the problem variables. By using mixture probability

distributions, a powerful, yet computationally tractable type of probability distribution can be used within EDAs, that provides for processing complicated interactions between a problem's variables.

For multi-objective optimization, mixture distributions can have a specific advantage that renders them particularly useful. The specific advantage is geometrical in nature. If we for instance cluster the solutions as observed in the objective space and then estimate a simpler probability distribution in each cluster, the probability distributions in these clusters can portray specific information about the different regions along the Pareto optimal front that we are ultimate interested in multi-objective optimization. Each simpler probability distribution to be used in each cluster can for instance be a factorized probability distribution as is used in most EDAs. Drawing new solutions from the resulting mixture probability distribution gives solutions that are more likely to be well spread along the front as each mixture component delivers a subset of new solutions. The use of such a mixture distribution thus results in a parallel exploration along the current Pareto front. This parallel exploration may very well provide a better spread of new solutions along the Pareto front than when a single non-mixture distribution is used to capture information about the complete Pareto front. In Fig. 2 an example is given of what the result of clustering the selected solutions in the objective space typically looks like. The effect of splitting up the solutions along the Pareto front, thereby facilitating parallel exploration along the front, can clearly be seen.

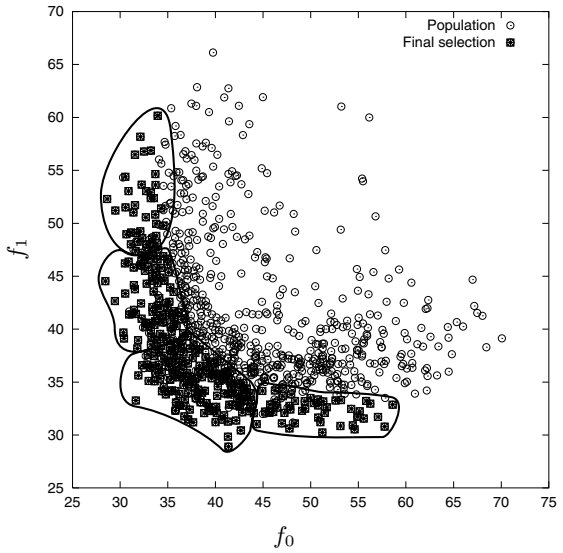


**Fig. 2.** An example of the breaking up the front of selected solutions using clustering. Objectives $f_0$ and $f_1$ should both be minimized. The four individual clusters that are defined in this example are outlined

**Estimating Mixture Distributions**

From the previous subsection describing the specific advantages of mixture probability distributions for multi-objective, we already have a straightforward manner to estimate mixture probability distributions from data using clustering. To actually build the mixture distribution from the simpler distributions, the mixing coefficients $\beta_i$ must still be chosen. This can be done in various ways. A common approach is to set $\beta_i$ to the proportion of the size of the $i^{\text{th}}$ cluster with respect to the sum of the sizes of all clusters. For the specific application of multi-objective optimization however, we propose to assign each cluster an equally large mixing coefficient, i.e. $\beta_i = 1/k$. The reason for this is that we want to distribute the solutions as good as possible along the Pareto front. Giving each cluster an equal probability of producing new solutions maximizes parallel exploration along the Pareto front. The only thing left to choose then is which clustering algorithm to use. Exact algorithms for partitioning (i.e. clustering into mutually disjoint subsets) exist [20], but the running times for these algorithms are of no practical use for building EDAs. What we require, is a fast approximate assessment of clusters such that we can estimate a relatively simple probability distribution in each cluster in a good way. Computationally efficient clustering algorithms exist that provide useful results [20]. Examples are the leader algorithm and the $K$-means algorithm.

A different approach to estimating a mixture probability distribution from data is to compute a maximum likelihood estimation. To this end, the Expectation Maximization (EM) algorithm [14] can be used. The EM algorithm is a general iterative approach to computing a maximum likelihood estimate. Although the EM algorithm is a valid approach to obtaining mixture probability distributions, it tends to be time-consuming, especially if the dimensionality of the data increases. Moreover, since we expect the specific benefit of mixture probability distributions to reside in dividing the data on the basis of its geometry in the objective space, using the EM algorithm seems less attractive because it builds a model completely based on the data as given in the parameter space.

### 3.3 Elitism

If elitism is used, the best solutions of the current generation are copied into the next generation. Alternatively, an external archive of a predefined maximum size $n_a$ may be used that contains only non-dominated solutions. This is actually similar to using elitism in a population, because this archive can be seen as the first few population members in a population for which the size is at least $n_p$ and at most $n_p + n_a$, where $n_p$ is the size of the population in an archive-based approach and $n_a$ is the size of the external archive.

Elitism plays an important role in multi-objective optimization since many solutions exist that are all equally preferable. It is important to have access to many of them during optimization to advance the complete set of non-dominated solutions further. An ideal variation operator is capable of generating solutions that are closer to the Pareto optimal front, but also spread out across the entire current set

of non-dominated solutions as well as possibly outside it to extend the diversity of the set of non-dominated solutions even further. However, obtaining new and diverse non-dominated solutions is hard, especially as the set of non-dominated solutions approaches the Pareto optimal front. If a non-dominated solution gets lost in a certain generation, it may take quite some effort before a new non-dominated solution in its vicinity is generated again. For this reason, elitism is commonly accepted [24, 36] to be a very important tool for improving the results obtained by any MOEA.

Elitism can be used within the $\mathbb{M}$IDEA framework in a straightforward manner because truncation selection is already used (Sect. 3.1). An elitist $\mathbb{M}$IDEA selects the best $\lfloor \tau n \rfloor$ solutions using the diversity-preserving truncation selection operator. Subsequently, only the worst $n - \lfloor \tau n \rfloor$ solutions are replaced with new offspring that result from sampling the estimated probability distribution. The best $\lfloor \tau n \rfloor$ solutions that were selected, are thus kept in the population.

### 3.4 The $\mathbb{M}$IDEA Framework

The $\mathbb{M}$IDEA variant that we use in our experiments is described in pseudo-code in Fig. 3.

| $\mathbb{M}$IDEA |
|---|
| 1 Initialize a population of $n$ random solutions and evaluate their objectives |
| 2 Iterate until termination |
| 2.1 Compute the domination counts |
| 2.2 Select $\lfloor \tau n \rfloor$ solutions with the diversity preserving selection operator |
| 2.3 Estimate a mixture probability distribution $P^{mixture}(\boldsymbol{\mathcal{Z}})$ |
| 2.4 Replace the non-selected solutions with new solutions drawn from $P^{mixture}(\boldsymbol{\mathcal{Z}})$ |
| 2.5 Evaluate the objectives of the new solutions |

**Fig. 3.** Pseudo-code for the $\mathbb{M}$IDEA framework

#### The Naive $\mathbb{M}$IDEA Instance

*Probability Distributions in Each Cluster*

In Sect. 3.2 we have argued that mixture distributions can play an important role in multi-objective optimization. Moreover, we have argued that a simple, but effective approach to estimating mixture distributions is to cluster the selected solutions on the bases of the geometry of their objective values. We therefore suggest keeping the probability distributions to be estimated in each cluster as simple as possible. This suggestion leads to the choice of using univariate factorized probability distributions in each cluster in the naive $\mathbb{M}$IDEA. In a factorized probability distribution, each random variable is regarded separately, meaning that a probability distribution is estimated for each random variable separately. For discrete random variables, this

amounts to repeatedly counting frequencies and computing proportions for a single random variable. For real-valued random variables this implies estimating for instance the mean and variance of a one-dimensional normal distribution repeatedly. The mathematical formulation of the univariate factorization is:

$$P^{univariate}(\boldsymbol{\mathcal{Z}}) = \prod_{i=0}^{l-1} P(Z_i) \tag{2}$$

Since in each cluster we thus disregard all dependencies between random variables, we call this specific $\mathbb{M}$IDE$\mathbb{A}$ instance naive in analogy with the well-known naive Bayes classifier. However, the clusters are expected to already provide a large benefit for multi-objective optimization. Moreover, algorithms such as UMDA [27] and the compact GA [19] that use same probability distribution as in (2) (without clustering) have provided good results on many interesting single-objective optimization problems. Hence, we already expect good optimization behavior for the naive $\mathbb{M}$IDE$\mathbb{A}$.

Clearly, non-naive instances of $\mathbb{M}$IDE$\mathbb{A}$ can be made directly by estimating more involved probability distributions in each cluster, such as Bayesian factorized probability distributions. Although we will present the results of some experiments with such more involved probability distributions for comparison reasons, we refer the interested reader for more details to the literature on either these probability distributions (e.g. [7, 10, 25]) or to the relevant literature on single-objective EDAs (e.g. [1, 2, 4, 18, 23, 28–33]).

*Clustering Algorithm*

Since we are interested in obtaining useful results in as little time as possible, we suggest the use of a fast clustering algorithm. Possibly this adds to the naiveness of our naive $\mathbb{M}$IDE$\mathbb{A}$ instance, but other clustering algorithms are easily implemented if required.

The algorithm that we propose to use is the leader algorithm. The leader algorithm is one of the fastest partitioning algorithms [20]. The use of it can thus be beneficial if the amount of overhead that is introduced by factorization mixture selection methods is desired to remain small. There is no need to specify in advance how many partitions there should be. The first solution to make a new partition is appointed to be its leader. The leader algorithm goes over the solutions exactly once. For each solution it encounters, it finds the first partition that has a leader being closer to the solution than a given threshold $\mathfrak{T}_d$. If no such partition can be found, a new partition is created containing only this single solution. To prevent the first partitions from becoming quite a lot larger than the later ones, we randomize the order in which the partitions are inspected. The asymptotic running time for finding the first partition with a leader closer than $\mathfrak{T}_d$ is the same as going over all partitions and finding the closest partition. Therefore, we prefer to find the closest partition.

One of the drawbacks of the (randomized) leader algorithm is that it is not invariant given the sequence of the input solutions. Most partitioning algorithms do not have this property, but not as strongly as the leader algorithm. Therefore, to be

sure that the ordering of the solutions is not subject to large repeating sequences of solutions, we randomize the ordering of the solutions each time the leader algorithm is applied.

*Pseudo-Code*

The naive $\mathbb{MIDEA}$ is an instance of the general $\mathbb{MIDEA}$ framework. Figure 4 shows how the naive $\mathbb{MIDEA}$ can be obtained from the general $\mathbb{MIDEA}$ framework by using a specific instantiation of lines 2.3 and 2.4.

naive $\mathbb{MIDEA}$

(*instantiation of steps 2.3 and 2.4 of the general* $\mathbb{MIDEA}$ *framework*)

1 $(\boldsymbol{c^0}, \boldsymbol{c^1}, \ldots, \boldsymbol{c^{k-1}}) \leftarrow$ LeaderAlgorithm$(\mathfrak{T}_d)$
2 **for** $i \leftarrow 0$ **to** $k-1$ **do**
  2.1 $\beta_i \leftarrow 1/k$
  2.2 **for** $j \leftarrow 0$ **to** $l-1$ **do**
    2.2.1 Estimate a one-dimensional probability distribution $P^{i,j}(Z_j)$ for random variable $Z_j$ from the solutions in the $i^{\text{th}}$ cluster (i.e. $\boldsymbol{c^i}$)
3 **for** $i \leftarrow \lfloor \tau n \rfloor$ **to** $n-1$ **do**
  3.1 Initialize a new solution $\boldsymbol{z}$
  3.2 Choose an index $q \in \{0, 1, \ldots, k-1\}$ with probability $\beta_q$
  3.3 **for** $j \leftarrow 0$ **to** $l-1$ **do**
    3.3.1 Draw a value for $\boldsymbol{z}_j$ from the one-dimensional probability distribution $P^{q,j}(Z_j)$ associated with the $q^{\text{th}}$-cluster
  3.4 Add $\boldsymbol{z}$ to the set of new offspring.

**Fig. 4.** Pseudo-code for the naive $\mathbb{MIDEA}$

## 4 Experiments

In this section we compare $\mathbb{MIDEA}$ instances to two well-known state-of-the-art MOEAs that aim at obtaining a diverse set of solutions along the Pareto front. The SPEA algorithm by Zitzler and Thiele [38] and the NSGA-II algorithm by Deb et al. [12] showed superior performance compared to most other MOEAs [12, 36]. The test suite we used consists of eight multi-objective optimization problems. We varied the dimensionality of these problems to get a total of sixteen problem instances to test the MOEAs on. The multi-objective optimization problems are described in Sect. 4.1. The performance measures we use to score the results of the algorithms with are described in Sect. 4.2. In Sect. 4.3 we present our experiment setup. In Sect. 4.4 we discuss the obtained results. Finally, in Sect. 4.5 we give a short summary for the EA practitioner.

### 4.1 Multi-objective Optimization Problems

Our test suite consists of problems with real-valued variables as well as with binary variables. To make a clear distinction between these two cases, we write real-valued variables as $y_i$ and binary variables as $x_i$. In both cases we have used four different optimization problems and two different dimensionalities for these problems to obtain a total test suite size of 16 problems. In the following we give a brief description of the problems in our test suite.

#### Real-valued Multi-objective Optimization Problems

A variety of test problems for real-valued variables has been proposed that may cause different types of problems for multi-objective optimization algorithms [11, 13, 36]. From this set of problems, we have selected three problems that are commonly used to benchmark multi-objective optimization algorithms. The fourth real-valued test problem is a new test problem we have designed to test the performance of MOEAs if there are strong interactions between the problem variables. These problems represent a spectrum of multi-objective problem difficulty as they make it difficult for a multi-objective optimization algorithm to progress towards the global optimal front and to maintain a diverse spread of solutions due to properties such as discontinuous fronts and multi-modality. The problems with real-valued variables that we use in our experiments are all defined for two objectives. An overview of our test problems is given in Fig. 5.

$BT_1$

Function $BT_1$ differs from the other three functions in that it has multivariate (linear) interactions between the problem variables. Therefore, more complex factorizations are required to exploit these interactions, whereas the other functions are well-suited to be optimized using the univariate factorization. The Pareto optimal front is given by $f_1(\boldsymbol{y}) = 1 - y_0$.

$ZDT_4$

Function $ZDT_4$ was introduced by Zitzler et al. [36]. It is very hard to obtain the optimal front $f_1(\boldsymbol{y}) = 1 - \sqrt{y_0}$ in $ZDT_4$ since there are many local fronts. Moreover, the number of local fronts increases as we get closer to the Pareto optimal front. The main problem that a MOEA should be able to overcome to optimize this problem is thus strong multi-modality.

$ZDT_6$

Function $ZDT_6$ was also introduced by Zitzler et al. [36]. The density of solutions in $ZDT_6$ increases as we move away from the Pareto optimal front. Furthermore, $ZDT_6$ has a non-uniform density of solutions *along* the Pareto optimal front as there are more solutions as $f_0(\boldsymbol{y})$ goes up to 1. Therefore, a good diverse spread of solutions along the Pareto front is hard to obtain. The Pareto front for $ZDT_6$ is given by $f_1(\boldsymbol{y}) = 1 - f_0(\boldsymbol{y})^2$ with $f_0(\boldsymbol{y}) \in [1 - e^{-1/3}; 1]$.

| Name | Definition | Range |
|---|---|---|
| $BT_1$ | **Minimize** $(f_0(\boldsymbol{y}), f_1(\boldsymbol{y}))$<br>**Where** • $f_0(\boldsymbol{y}) = y_0$<br>• $f_1(\boldsymbol{y}) = 1 - f_0(\boldsymbol{y}) + 10^7 - \frac{100}{\left(10^{-5} + \sum_{i=1}^{l-1} \lvert \sum_{j=1}^{i} y_i \rvert\right)}$ | • $y_0 \in [0; 1]$<br>• $y_i \in [-3; 3]$<br>$(1 \leq i < l)$ |
| $ZDT_4$ | **Minimize** $(f_0(\boldsymbol{y}), f_1(\boldsymbol{y}))$<br>**Where** • $f_0(\boldsymbol{y}) = y_0$<br>• $f_1(\boldsymbol{y}) = \gamma\left(1 - \sqrt{\frac{f_0(\boldsymbol{y})}{\gamma}}\right)$<br>• $\gamma = 1 + 10(l-1) + \sum_{i=1}^{l-1}\left(y_i^2 - 10\cos(4\pi y_i)\right)$ | • $y_0 \in [0; 1]$<br>• $y_i \in [-5; 5]$<br>$(1 \leq i < l)$ |
| $ZDT_6$ | **Minimize** $(f_0(\boldsymbol{y}), f_1(\boldsymbol{y}))$<br>**Where** • $f_0(\boldsymbol{y}) = 1 - e^{-4y_0}\sin^6(6\pi y_0)$<br>• $f_1(\boldsymbol{y}) = \gamma\left(1 - \left(\frac{f_0(\boldsymbol{y})}{\gamma}\right)^2\right)$<br>• $\gamma = 1 + 9\left(\sum_{i=1}^{l-1} \frac{y_i}{9}\right)^{0.25}$ | • $y_i \in [0; 1]$<br>$(0 \leq i < l)$ |
| $CTP_7$ | **Minimize** $(f_0(\boldsymbol{y}), f_1(\boldsymbol{y}))$<br>**Where** • $f_0(\boldsymbol{y}) = y_0$<br>• $f_1(\boldsymbol{y}) = \gamma\left(1 - \frac{f_0(\boldsymbol{y})}{\gamma}\right)$<br>• $\gamma = 1 + 10(l-1) + \sum_{i=1}^{l-1}\left(y_i^2 - 10\cos(4\pi y_i)\right)$<br>**Such that** • $\cos(-\frac{5\pi}{100})f_1(\boldsymbol{y}) - \sin(-\frac{5\pi}{100})f_0(\boldsymbol{y}) \geq 40\lvert\sin(5\pi\left[\sin(-\frac{5\pi}{100})f_1(\boldsymbol{y}) + \cos(-\frac{5\pi}{100})f_0(\boldsymbol{y})\right])\rvert^6$ | • $y_0 \in [0; 1]$<br>• $y_i \in [-5; 5]$<br>$(1 \leq i < l)$ |

**Fig. 5.** Real-valued multi-objective optimization test problems

### $CTP_7$

Function *CTP*$_7$ was introduced by Deb et al. [13]. Its Pareto optimal front differs slightly from that of *ZDT*$_4$, but otherwise shares the multi-modal front problem. In addition, this problem has constraints in the objective space, which makes finding a diverse representation of the Pareto front more difficult since the Pareto front is discontinuous and it is hard to obtain an approximation that has a few solutions in each feasible part of that front.

### Binary Multi-objective Optimization Problems

In Fig. 6, we have specified four binary multi-objective optimization problems. Next to being binary, these problems are also multi-objective variants of well-known com-

| Name | Definition |
|---|---|
| *MS* (Maximum Satisfiability) | **Maximize** $(f_0(\boldsymbol{x}), f_1(\boldsymbol{x}), \ldots, f_{m-1}(\boldsymbol{x}))$<br>**Where** • $\forall_{i \in \boldsymbol{\mathcal{M}}} : f_i(\boldsymbol{x}) = \sum_{j=0}^{c_i - 1} \text{sgn}\left(\left[\sum_{k=0}^{l-1} (C_i)_{jk} \otimes x_k\right]\right)$<br>• $\text{sgn}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$<br>• $\begin{array}{c\|cc} \otimes & 0 & 1 \\ \hline -1 & 1 & 0 \end{array}$ $\begin{array}{c\|cc} \otimes & 0 & 1 \\ \hline 0 & 0 & 0 \end{array}$ $\begin{array}{c\|cc} \otimes & 0 & 1 \\ \hline 1 & 0 & 1 \end{array}$ |
| *KN* (Knapsack) | **Maximize** $(f_0(\boldsymbol{x}), f_1(\boldsymbol{x}), \ldots, f_{m-1}(\boldsymbol{x}))$<br>**Where** • $\forall_{i \in \boldsymbol{\mathcal{M}}} : f_i(\boldsymbol{x}) = \sum_{j=0}^{l-1} P_{ij} x_j$<br>**Such that** • $\forall_{i \in \boldsymbol{\mathcal{M}}} : \sum_{j=0}^{l-1} W_{ij} x_j \leq c_i$ |
| *SC* (Set Covering) | **Minimize** $(f_0(\boldsymbol{x}), f_1(\boldsymbol{x}), \ldots, f_{m-1}(\boldsymbol{x}))$<br>**Where** • $\forall_{i \in \boldsymbol{\mathcal{M}}} : f_i(\boldsymbol{x}) = \sum_{j=0}^{l-1} C_{ij} x_j$<br>**Such that** • $\forall_{i \in \boldsymbol{\mathcal{M}}} : \forall_{0 \leq j < r} : \sum_{k=0}^{l-1} (A_i)_{jk} x_k \geq 1$ |
| *MST* (Minimal Spanning Tree) | **Minimize** $(f_0(\boldsymbol{x}), f_1(\boldsymbol{x}), \ldots, f_{m-1}(\boldsymbol{x}))$<br>**Where** • $\forall_{i \in \boldsymbol{\mathcal{M}}} : f_i(\boldsymbol{x}) = \sum_{j=0}^{l-1} W_{ij} x_j$<br>**Such that** • $\forall_{S \subseteq V} : \sum_{x_j \in (S \times (V-S))} x_j \geq 1$<br>• $\forall_{S \subseteq V} : \sum_{x_j \in (S \times S)} x_j \leq \lvert S \rvert - 1$ |

**Fig. 6.** Binary multi-objective combinatorial optimization test problems

binatorial optimization problems. The number of objectives for these problems is not restricted to two and is denoted by $m$.

It is important to note that we have used random instances for the combinatorial optimization problems. In the case of only a single objective, random instances may on average be easy for some combinatorial problems. However, in the case of multiple objectives, finding the Pareto front is usually much more difficult, even if efficient algorithms are available for the single-objective case [15]. Therefore, the instances used in our test suite are not expected to be over-easy. Furthermore, the problems also serve to indicate differences between the different multi-objective algorithmic approaches other than the fact that dependencies between problem variables can be exploited. This relative performance of the algorithms may be well observed using our proposed test-suite. On the other hand, the degree of interaction between the problem variables in randomly generated problem instances may not be too large, which may cause optimization algorithms that regard the problem variables independently of each other to be the most efficient.

*Maximum Satisfiability*

In the maximum satisfiability problem, we are given a propositional formula in conjunctive normal form. The goal is to satisfy as many clauses as possible. The solution string is a truth assignment to the involved literals. These formulas can be represented by a matrix in which row $i$ specifies what literals appear either positive (1) or negative $(-1)$ in clause $i$. In the multi-objective variant of this problem, we have $m$ of such matrices and only a single solution to satisfy as many clauses as possible in each objective at the same time.

*Knapsack*

The multi-objective knapsack problem was first used to test MOEAs on by Zitzler and Thiele [38]. We are given $m$ knapsacks with a specified capacity and $n$ items. Each item can have a different weight and profit in every knapsack. Selecting item $i$ in a solution implies placing it in every knapsack. A solution may not cause exceeding the capacity of any knapsack.

*Set Covering*

In the set covering problem, we are given $l$ locations at which we can place some service at a specified cost. Furthermore, associated with each location is a set of regions $\subseteq \{0, 1, \ldots r-1\}$ that can be serviced from that location. The goal is to select locations such that *all* regions are serviced against minimal costs. In the multi-objective variant of set covering, $m$ services are placed at a location. Each service however covers its own set of regions when placed at a certain location and has its own cost associated with a certain location. A binary solution indicates at which locations the services are placed.

*Minimal Spanning Tree*

In the minimal spanning tree problem we are given an undirected graph $(V, E)$ such that each edge has a certain weight. We are interested in selecting edges $E_T \subseteq E$ such that $(V, E_T)$ is a spanning tree. The objective is to find a spanning tree such that the weight of all its edges is minimal. In the multi-objective variant of this problem, each edge can have a different weight in each objective.

### 4.2 Performance Indicators

To measure the performance of a MOEA we only consider the subset of all non-dominated solutions that is contained in the final population that results from running the MOEA. We call such a subset an approximation set and denote it by $\mathcal{S}$. The size of the approximation set depends on the settings used to run the MOEA with.

To actually measure performance, performance indicators are used. A performance indicator is a function that, given an approximation set $\mathcal{S}$, returns a real value that indicates how good $\mathcal{S}$ is with respect to a certain feature that is measured by the

performance indicator. Performance indicators are commonly used to determine the performance of a MOEA and to compare this performance with other MOEAs if the number of evaluations is fixed beforehand. More detailed information regarding the importance of using good performance indicators to evaluate MOEAs may be found in dedicated literature [5, 22, 37].

Since we are interested in performance as measured in the objective space, we define the distance between two multi-objective solutions $\boldsymbol{z}^0$ and $\boldsymbol{z}^1$ to be the Euclidean distance between their objective values $f(\boldsymbol{z}^0)$ and $f(\boldsymbol{z}^1)$:

$$d(\boldsymbol{z}^0, \boldsymbol{z}^1) = \sqrt{\sum_{i=0}^{m-1} (f_i(\boldsymbol{z}^1) - f_i(\boldsymbol{z}^0))^2} \tag{3}$$

If we only want to measure diversity, we can use the **FS** (*Front Spread*) indicator. This performance indicator was first used by Zitzler [35]. The **FS** indicator indicates the size of the objective space covered by an approximation set. A larger **FS** indicator value is preferable. The **FS** indicator for an approximation set $\mathcal{S}$ is defined to be the maximum Euclidean distance inside the smallest $m$-dimensional bounding-box that contains $\mathcal{S}$. This distance can be computed using the maximum distance among the solutions in $\mathcal{S}$ in each dimension separately:

$$\mathbf{FS}(\mathcal{S}) = \sqrt{\sum_{i=0}^{m-1} \max_{(\boldsymbol{z}^0, \boldsymbol{z}^1) \in \mathcal{S} \times \mathcal{S}} \{(f_i(\boldsymbol{z}^0) - f_i(\boldsymbol{z}^1))^2\}} \tag{4}$$

In combination with the **FS** indicator, it is also important to know how many points are available in the set of non-dominated solutions, because a larger set of trade-off points is more desirable. This quantity is called the **FO** (*Front Occupation*) indicator and was first used by Van Veldhuizen [34]. A larger **FO** indicator value is preferable.

$$\mathbf{FO}(\mathcal{S}) = |\mathcal{S}| \tag{5}$$

The ultimate goal is to cover the Pareto optimal front. An intuitive way to define the distance between an approximation set $\mathcal{S}$ and the Pareto optimal front is to average the minimum distance between a solution and the Pareto optimal front over each solution in $\mathcal{S}$. We refer to this distance as the distance from a set of non-dominated solutions to the Pareto optimal front and it serves as an indicator of how close an approximation set has come to the Pareto optimal front. We denote it by $\boldsymbol{D}_{\mathcal{S} \rightarrow \boldsymbol{\mathcal{P}}_{\boldsymbol{F}}}$. This performance indicator was first used by Van Veldhuizen [34]. A smaller value for this performance indicator is preferable.

$$\boldsymbol{D}_{\mathcal{S} \rightarrow \boldsymbol{\mathcal{P}}_{\boldsymbol{F}}}(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{\boldsymbol{z}^0 \in \mathcal{S}} \min_{\boldsymbol{z}^1 \in \boldsymbol{\mathcal{P}}_{\boldsymbol{S}}} \{d(\boldsymbol{z}^0, \boldsymbol{z}^1)\} \tag{6}$$

An approximation set with a good $\boldsymbol{D}_{\mathcal{S} \rightarrow \boldsymbol{\mathcal{P}}_{\boldsymbol{F}}}$ indicator value does not imply that a good diverse representation of the Pareto optimal set has been obtained, since the

indicator only reflects how far away the obtained points are from the Pareto optimal front on average. An approximation set consisting of only a single solution can already have a low value for this indicator. To include the goal of diversity, the reverse of the $\boldsymbol{D}_{\mathcal{S}\rightarrow\boldsymbol{\mathcal{P}_F}}$ indicator is a better guideline for evaluating MOEAs. In the reverse distance indicator, we compute for each solution in the Pareto optimal set the distance to the closest solution in an approximation set $\mathcal{S}$ and take the average as the indicator value. We denote this indicator by $\boldsymbol{D}_{\boldsymbol{\mathcal{P}_F}\rightarrow\mathcal{S}}$ and refer to it as the distance from the Pareto optimal front to an approximation set. A smaller value for this performance indicator is preferable. In the definition of this indicator, we must realize that the Pareto optimal front may be continuous. For an exact definition, we therefore have to use a line integration over the entire Pareto front. For a 2-dimensional multi-objective problem we obtain the following expression:

$$\boldsymbol{D}_{\boldsymbol{\mathcal{P}_F}\rightarrow\mathcal{S}}(\mathcal{S}) = \int_{\boldsymbol{\mathcal{P}_F}} \min_{\boldsymbol{z}^0\in\mathcal{S}}\{d(\boldsymbol{z}^0,\boldsymbol{z}^1)\}df(\boldsymbol{z}^1) \tag{7}$$

In most practical experiments, it is easier to compute a uniformly sampled set of many solutions along the Pareto optimal front and to use this discretized representation of $\boldsymbol{\mathcal{P}_F}$ instead. A discretized version of the Pareto optimal front is also available if a discrete multi-objective optimization problem is being solved. In the discrete case, the $\boldsymbol{D}_{\mathcal{S}\rightarrow\boldsymbol{\mathcal{P}_F}}$ indicator is defined by:

$$\boldsymbol{D}_{\boldsymbol{\mathcal{P}_F}\rightarrow\mathcal{S}}(\mathcal{S}) = \frac{1}{|\boldsymbol{\mathcal{P}_S}|}\sum_{\boldsymbol{z}^1\in\boldsymbol{\mathcal{P}_S}} \min_{\boldsymbol{z}^0\in\mathcal{S}}\{d(\boldsymbol{z}^0,\boldsymbol{z}^1)\} \tag{8}$$

An illustration of the $\boldsymbol{D}_{\boldsymbol{\mathcal{P}_F}\rightarrow\mathcal{S}}$ indicator is presented in Fig. 7. The $\boldsymbol{D}_{\boldsymbol{\mathcal{P}_F}\rightarrow\mathcal{S}}$ indicator represents both the goal of getting close to the Pareto optimal front as well as

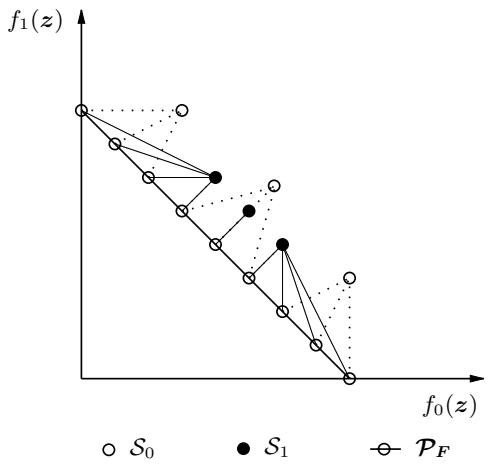


**Fig. 7.** The approximation set $\mathcal{S}_1$ is closer to the (discretized) Pareto optimal front but has less diversity, while approximation set $\mathcal{S}_0$ is further away from the front but has greater diversity: both sets have approximately the same $\boldsymbol{D}_{\boldsymbol{\mathcal{P}_F}\rightarrow\mathcal{S}}$ indicator value though

the goal of getting a diverse, wide-spread front of solutions. The $\boldsymbol{D}_{\boldsymbol{\mathcal{P}}_{\boldsymbol{F}}\rightarrow\mathcal{S}}$ indicator for an approximation set $\mathcal{S}$ is zero if and only if all points in $\boldsymbol{\mathcal{P}}_{\boldsymbol{F}}$ are contained in $\mathcal{S}$ as well. Furthermore, a single solution from the Pareto optimal set will lead to the same $\boldsymbol{D}_{\boldsymbol{\mathcal{P}}_{\boldsymbol{F}}\rightarrow\mathcal{S}}$ indicator as a more diverse set of solutions that has objective values that are slightly further away from the Pareto optimal front. Moreover, a similarly diverse approximation set of solutions that is closer to the Pareto optimal front, will have a lower $\boldsymbol{D}_{\boldsymbol{\mathcal{P}}_{\boldsymbol{F}}\rightarrow\mathcal{S}}$ indicator value. However, an approximation set of solutions that is extremely diverse but far away from the Pareto optimal front, such as the non-dominated solutions of a randomly generated set of solutions, has a bad $\boldsymbol{D}_{\boldsymbol{\mathcal{P}}_{\boldsymbol{F}}\rightarrow\mathcal{S}}$ indicator value. This underlines the important point that diversity is *not* equally important as is getting close to the Pareto optimal front because a larger diversity is often not hard to come by. What is important is the diversity *along* the objectives of a set of non-dominated solutions that is as close as possible to the Pareto optimal front.

A performance indicator that is closely related to the $\boldsymbol{D}_{\boldsymbol{\mathcal{P}}_{\boldsymbol{F}}\rightarrow\mathcal{S}}$ indicator, is the hypervolume indicator by Knowles and Corne [22]. In the hypervolume indicator, a point in the objective space is picked such that it is dominated by all points in the approximation sets that need to be evaluated. The indicator value is then equal to the hypervolume of the multi-dimensional region enclosed by the approximation set and the picked reference point. This value is an indicator of the region in the objective space that is dominated by the approximation set. The main difference between the hypervolume indicator and the $\boldsymbol{D}_{\boldsymbol{\mathcal{P}}_{\boldsymbol{F}}\rightarrow\mathcal{S}}$ indicator is that for the hypervolume indicator a reference point has to be chosen. Different reference points lead to different indicator values. Moreover, different reference points can lead to indicator values that indicate a preference for different approximation sets. Since in the $\boldsymbol{D}_{\boldsymbol{\mathcal{P}}_{\boldsymbol{F}}\rightarrow\mathcal{S}}$ indicator the true Pareto optimal front is used, the $\boldsymbol{D}_{\boldsymbol{\mathcal{P}}_{\boldsymbol{F}}\rightarrow\mathcal{S}}$ indicator does not suffer from this drawback. Of course, a major drawback of the $\boldsymbol{D}_{\boldsymbol{\mathcal{P}}_{\boldsymbol{F}}\rightarrow\mathcal{S}}$ indicator is that in a real application the true Pareto optimal front is not known beforehand. In that case, the Pareto front of all approximation sets could be used as a substitute for the actual Pareto optimal front.

### 4.3 Experiment Setup

#### Optimization Problem Dimensionalities

*Real-Valued Multi-Objective Optimization Problems*

For the real-valued problems, we tested all algorithms with both $l = 10$ and $l = 100$ problem variables.

*Binary Multi-Objective Optimization Problems*

For the binary problems, we used test instances with $l = 100$ and $l = 1000$. For the maximum satisfiability problem, we generated the test instances by generating 2500 clauses for $l = 100$ and 12500 clauses for $l = 1000$ with a random number

of literals between 1 and 5. For the knapsack problem, we generated instances by generating random weights in $[1; 10]$ and random profits in $[1; 10]$. The capacity of a knapsack was set at half of the total weight of all the items, weighted according to that knapsack objective. For set covering, the costs were generated at random in $[1; 10]$. We used 250 regions and 2500 regions to be serviced for $l = 100$ and $l = 1000$ respectively. We varied the problem difficulty through the region-location adjacency relation. This relation was generated by making each location adjacent to 70 and 50 randomly selected regions for $l = 100$ and $l = 1000$ respectively. Finally, for the minimum spanning tree problem, we used full graphs with 105 edges (15 vertices) and 1035 edges (46 vertices). The dimensionality of these problems is therefore not precisely 100 and 1000. The weights of the edges were generated randomly in $[1; 10]$.

**Optimization Problem Constraints**

Problems $CTP_7$, set covering, knapsack and minimal spanning tree have constraints. To deal with them, we can use a repair mechanism to transform infeasible solutions into feasible solutions. Another approach is based on the notion of constraint-domination introduced by Deb et al. [13]. This notion allows to deal with constrained multi-objective problems in a general fashion. A solution $\boldsymbol{z}^0$ is said to constraint-dominate solution $\boldsymbol{z}^1$ if any of the following is true:

1. Solution $\boldsymbol{z}^0$ is feasible and solution $\boldsymbol{z}^1$ is infeasible
2. Solutions $\boldsymbol{z}^0$ and $\boldsymbol{z}^1$ are both infeasible, but $\boldsymbol{z}^0$ has a smaller overall constraint violation
3. Solutions $\boldsymbol{z}^0$ and $\boldsymbol{z}^1$ are both feasible and $\boldsymbol{z}^0 \succ \boldsymbol{z}^1$

The overall constraint violation is the amount by which a constraint is violated, summed over all constraints. We have used this principle for problems $CTP_7$ and set covering. For the knapsack problem, an elegant repair mechanism was proposed earlier by Zitzler and Thiele [38]. For the minimal spanning tree problem, the number of constraints grows exponentially with the problem size $l$. We therefore propose to use repair mechanisms for these latter two problems.

*Knapsack Repair Mechanism*

If a solution violates a constraint, the repair mechanism iteratively removes items until all constrains are satisfied. The order in which the items are investigated, is determined by the maximum profit/weight ratio. The items with the lowest profit/weight ratio are removed first.

*Minimal Spanning Tree Repair Mechanism*

First the edges are removed from the currently constructed graph and they are sorted according to their weight. Next, they are added to the graph so that no cycles are

introduced. This is done by only allowing edges to be introduced between the connected components in the graph. If after this phase, the number of connected components has not been reduced to 1, all edges between the connected components are regarded in increasing weight and again the connected components are merged until a single component is left.

### General Algorithmic Setup

We ran every algorithm 50 times on each problem. In any single run we chose to allow a maximum of $20 \cdot 10^3$ evaluations for the real-valued problems of dimensionality $l = 10$ and the binary problems of dimensionality $l = 100$ and a maximum of $100 \cdot 10^3$ evaluations for the real-valued problems of dimensionality $l = 100$ and the binary problems of dimensionality $l = 1000$. As a result of imposing the restriction of a maximum of evaluations, a value for the population size $n$ exists for each MOEA such that the MOEA will perform best. For too large population sizes, the search will move towards a random search and for too small population sizes, there is not enough information to perform adequate model selection and induction. We therefore increased the population size in steps of 25 to find the best results. To actually select the best population size, we selected the result with the lowest value for the $\boldsymbol{D}_{\boldsymbol{\mathcal{P}_F} \rightarrow \mathcal{S}}$ indicator.

### Algorithms

We tested a few variants of three MOEAs. In the following we will describe the details that are required in addition to the details given in earlier sections for constructing the actual MOEAs that we will use for testing.

*SPEA*

For SPEA, we used uniform crossover and one-point crossover with a probability of 0.8. Bit-flipping mutation was used in combination with either of these recombination operators with a probability of 0.01. These settings were used previously by the SPEA authors [36]. We allowed the size of the external storage in SPEA to become as large as the population size. For the real problems, we encoded every variable with 30 bits.

*NSGA-II*

For NSGA-II, we used the same crossover and mutation operators and the same encoding for the real variables.

$\mathbb{M}$IDE$\mathbb{A}$

For $\mathbb{M}$IDE$\mathbb{A}$, we used the leader clustering algorithm in the objective space such that four clusters were constructed on average. If the number of clusters becomes too large, the requirements for the population size increases in order to facilitate

proper factorization selection in each cluster. We do not suggest that the number of clusters we use is optimal, but it will serve to indicate the effectiveness of parallel exploration along the Pareto front as well as diversity preservation. In each cluster, we either used the univariate factorization (i.e. naive $\mathbb{MIDEA}$) or we estimated a Bayesian factorization based upon normal distributions in the case of real variables. For details on how the Bayesian factorization is learned, see [1]. However, in the case of 100-dimensional real-valued problems, we allowed only at most a single parent for any variable. In the case of binary variables, we used the optimal dependency tree algorithm by Chow and Liu [8] to estimate a tree factorization in each cluster. To further investigate the influence of the different components in the $\mathbb{MIDEA}$ algorithm, we also performed tests in which only a single cluster is used. Furthermore, we also replaced the use of estimating probability distributions by the use of one-point crossover and uniform crossover with mutation as used in the SPEA and NSGA-II algorithms. In the case of clustering in combination with the use of crossover operators, restricted mating was employed in order to ensure clustered exploration along the front. In restricted mating crossover, an offspring is produced using two parent solutions that are picked from the same cluster. For the truncation percentile, we used the rule of thumb by Mühlenbein and Mahnig [26] and set $\tau$ to $0.3$. Furthermore, for the comparison benchmarks, we set the diversity preservation parameter to $\delta = 1.5$, which was experimentally determined to give good results both with respect to diversity preservation as well as selective pressure. For an investigation of the influence of $\delta$ on the performance of $\mathbb{MIDEA}$, we also varied $\delta$ and observed the results in some additional experiments, the results of which are reported below.

*Overview of Abbreviations*

In presenting the results, the different evolutionary algorithms that were tested are abbreviated to save space. For reference, a list of abbreviations that we have used is presented in Fig. 8.

| **Abbrev.** | **Meaning** |
|---|---|
| UX | Uniform crossover (prob. 1) + bit-flipping mutation (prob. 0.01) |
| 1X | One-point crossover (prob. 1) + bit-flipping mutation (prob. 0.01) |
| Univariate | The univariate factorization (2) |
| Learning | A more advanced Bayesian factorization is learned |
| 1 Cluster | No clustering because everything is placed in a single cluster |
| Par. Clust. | Clustering in the parameter space |
| Obj. Clust. | Clustering in the objective space |
| $\mathbb{M}$ | An instance of the $\mathbb{MIDEA}$ framework |

**Fig. 8.** List of abbreviations used in the presentation of the results

### 4.4 Results

To compare the MOEAs, we investigate their average performance with respect to performance indicators introduced in Sect. 4.2. The performance indicators that we use are the $\boldsymbol{D}_{\mathcal{P}_F \rightarrow \mathcal{S}}$ indicator, the **FS** indicator and the **FO** indicator. For the $\boldsymbol{D}_{\mathcal{P}_F \rightarrow \mathcal{S}}$ performance indicator, we used different sets to represent the Pareto optimal front for the real-valued optimization problems and the binary optimization problems. For the real-valued optimization problems we used a uniformly sampled set of $5000$ solutions along the Pareto optimal front. Since we do not know the Pareto optimal front for the binary optimization problems, we used the Pareto front over all results obtained by all MOEAs.

For each of the performance indicators, we computed their average and standard deviation over the 50 runs to get an assessment of their performance. The averages are tabulated in Figs. 9 through 14. The best results are written in boldface. For each algorithm, the type of variation is indicated as a superscript. The $\mathbb{M}$IDE$\mathbb{A}$ algorithms are indicated by a single $\mathbb{M}$ symbol. For all tested $\mathbb{M}$IDE$\mathbb{A}$ algorithms, the subscript indicates whether only a single cluster was used or whether clustering was performed in either the parameter space or the objective space. The population sizes that led to the best performance, are tabulated in Figs. 15 and 16. For the standard deviations, we refer the interested reader to a technical report [6]. Although the average behavior is the most interesting, the standard deviations are vital to determine whether the differences in the average behavior of the different algorithms are significant. To investigate these significances, we have performed Aspin-Welch-Satterthwaite (AWS) statistical hypothesis $T$-tests at a significance level of $\alpha = 0.05$. The AWS $T$-test is a statistical hypothesis test for the equality of means in which the equality of variances is not assumed [21]. For each problem, we verified for each pair of algorithms whether the average obtained performance indicator values differ significantly. We assigned a value of $1$ if an algorithm scored significantly better and a value of $-1$ if an algorithm scored significantly worse. We summed the so obtained matrices over all problems to get the statistically significant improvement matrices that are shown in Figs. 17 through 19. We also computed the sum for each algorithm of its significant improvement values over all other algorithms to indicate the summed relative statistically significant performance of the algorithms. A less detailed summary of the statistical significance tests is shown in Fig. 21. In this figure histograms are used to indicate the sum of the results of the statistical significance tests for each algorithm compared with all other algorithms. The histogram represents the sums for the real-valued problems and the combinatorial problems for the different tested dimensionalities and the average of these four sums.

*Influence of Problem Dimensionality*

Although the $\mathbb{M}$IDE$\mathbb{A}$ variants already mostly outperform the other tested algorithms in the case in which the dimensionality of the problem is smaller ($l = 10$ for the real-valued problems, $l = 100$ for the binary problems), they perform even better in the case in which the dimensionality of the problem is larger. This is most likely due to

| | $D_{\mathcal{P}_F \to \mathcal{S}}$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **EA** | $BT_1^{10}$ | $ZDT_4^{10}$ | $ZDT_6^{10}$ | $CTP_7^{10}$ | $BT_1^{100}$ | $ZDT_4^{100}$ | $ZDT_6^{100}$ | $CTP_7^{100}$ |
| SPEA$^{\text{UX}}$ | $100 \cdot 10^5$ | 4.62 | 0.193 | 7.97 | $100 \cdot 10^5$ | 470 | 7.64 | 499 |
| SPEA$^{\text{1X}}$ | $100 \cdot 10^5$ | 3.90 | 0.172 | 7.31 | $100 \cdot 10^5$ | 447 | 7.06 | 476 |
| NSGA-II$^{\text{UX}}$ | $100 \cdot 10^5$ | 4.39 | 0.303 | 7.25 | $100 \cdot 10^5$ | 360 | 5.99 | 348 |
| NSGA-II$^{\text{1X}}$ | $100 \cdot 10^5$ | **1.40** | 0.328 | **3.32** | $100 \cdot 10^5$ | 297 | 6.59 | 303 |
| $\mathbb{M}^{\text{UX}}_{\text{1 Cluster}}$ | $100 \cdot 10^5$ | 4.43 | 0.358 | 6.63 | $100 \cdot 10^5$ | 374 | 6.72 | 378 |
| $\mathbb{M}^{\text{1X}}_{\text{1 Cluster}}$ | $100 \cdot 10^5$ | 1.89 | 0.291 | 4.13 | $100 \cdot 10^5$ | 336 | 6.81 | 345 |
| $\mathbb{M}^{\text{UX}}_{\text{Par. Clust.}}$ | $100 \cdot 10^5$ | 4.01 | 0.368 | 6.42 | $100 \cdot 10^5$ | 400 | 6.98 | 394 |
| $\mathbb{M}^{\text{1X}}_{\text{Par. Clust.}}$ | $100 \cdot 10^5$ | 1.65 | 0.298 | 3.77 | $100 \cdot 10^5$ | 332 | 7.01 | 340 |
| $\mathbb{M}^{\text{UX}}_{\text{Obj. Clust.}}$ | $100 \cdot 10^5$ | 3.98 | 0.354 | 7.27 | $100 \cdot 10^5$ | 311 | 5.96 | 326 |
| $\mathbb{M}^{\text{1X}}_{\text{Obj. Clust.}}$ | $100 \cdot 10^5$ | 2.03 | 0.311 | 3.95 | $100 \cdot 10^5$ | 328 | 6.74 | 335 |
| $\mathbb{M}^{\text{Univariate}}_{\text{1 Cluster}}$ | $100 \cdot 10^5$ | 14.0 | 1.08 | 16.5 | $100 \cdot 10^5$ | 774 | 3.06 | 875 |
| $\mathbb{M}^{\text{Learning}}_{\text{1 Cluster}}$ | $100 \cdot 10^5$ | 11.2 | **0.00239** | 15.3 | $100 \cdot 10^5$ | 597 | **0.434** | 600 |
| $\mathbb{M}^{\text{Univariate}}_{\text{Par. Clust.}}$ | $999 \cdot 10^4$ | 5.36 | 0.798 | 7.93 | $100 \cdot 10^5$ | 168 | 3.70 | 192 |
| $\mathbb{M}^{\text{Learning}}_{\text{Par. Clust.}}$ | $999 \cdot 10^4$ | 14.0 | 0.159 | 17.1 | $100 \cdot 10^5$ | 416 | 0.470 | 523 |
| naive $\mathbb{M}$IDEA | $100 \cdot 10^5$ | 5.00 | 0.306 | 8.64 | $100 \cdot 10^5$ | 157 | 4.60 | **161** |
| $\mathbb{M}^{\text{Learning}}_{\text{Obj. Clust.}}$ | $\mathbf{998 \cdot 10^4}$ | 11.5 | 0.287 | 12.6 | $\mathbf{100 \cdot 10^5}$ | **144** | 1.30 | 165 |

**Fig. 9.** Average of the $D_{\mathcal{P}_F \to \mathcal{S}}$ performance indicator on all real-valued problems. Note: naive $\mathbb{M}$IDEA could also have been abbreviated as $\mathbb{M}^{\text{Univariate}}_{\text{Obj. Clust.}}$

| | $D_{\mathcal{P}_F \to \mathcal{S}}$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **EA** | $MS^{100}$ | $KN^{100}$ | $SC^{100}$ | $MST^{105}$ | $MS^{1000}$ | $KN^{1000}$ | $SC^{1000}$ | $MST^{1035}$ |
| SPEA$^{\text{UX}}$ | 12.7 | 10.4 | 2.93 | 2.10 | 181 | 83.9 | 550 | 6.78 |
| SPEA$^{\text{1X}}$ | 11.8 | 9.14 | 2.99 | 2.12 | 270 | 105 | 484 | 6.40 |
| NSGA-II$^{\text{UX}}$ | 11.5 | 8.29 | 1.79 | 1.88 | 180 | 76.4 | 289 | 7.15 |
| NSGA-II$^{\text{1X}}$ | 11.7 | 9.33 | 2.64 | 2.22 | 283 | 114 | 360 | 6.60 |
| $\mathbb{M}^{\text{UX}}_{\text{1 Cluster}}$ | 9.65 | 6.20 | **0.931** | 2.76 | 80.4 | 52.3 | **72.4** | 5.14 |
| $\mathbb{M}^{\text{1X}}_{\text{1 Cluster}}$ | 12.4 | 7.34 | 1.9 | 2.72 | 135 | 93.0 | 109 | 4.66 |
| $\mathbb{M}^{\text{UX}}_{\text{Par. Clust.}}$ | 10.6 | 6.96 | 1.23 | 2.69 | 104 | 58.8 | 75.4 | 5.42 |
| $\mathbb{M}^{\text{1X}}_{\text{Par. Clust.}}$ | 13.4 | 8.13 | 1.54 | 2.86 | 169 | 107 | 101 | 4.96 |
| $\mathbb{M}^{\text{UX}}_{\text{Obj. Clust.}}$ | 7.50 | **3.71** | 1.49 | **1.30** | 69.0 | **18.8** | 189 | 3.33 |
| $\mathbb{M}^{\text{1X}}_{\text{Obj. Clust.}}$ | 10.5 | 5.98 | 1.89 | 1.54 | 116 | 46.3 | 305 | 3.11 |
| $\mathbb{M}^{\text{Univariate}}_{\text{1 Cluster}}$ | 18.8 | 16.4 | 1.48 | 3.18 | 141 | 117 | 76.5 | 9.60 |
| $\mathbb{M}^{\text{Learning}}_{\text{1 Cluster}}$ | 11.4 | 7.25 | 1.50 | 2.70 | 262 | 77.6 | 94.2 | 5.89 |
| $\mathbb{M}^{\text{Univariate}}_{\text{Par. Clust.}}$ | 18.3 | 13.2 | 1.54 | 3.26 | 168 | 118 | 105 | 9.68 |
| $\mathbb{M}^{\text{Learning}}_{\text{Par. Clust.}}$ | 12.5 | 7.56 | 1.85 | 2.54 | 262 | 115 | 269 | 7.69 |
| naive $\mathbb{M}$IDEA | **7.20** | 4.32 | 1.24 | 1.54 | **36.9** | 28.1 | 181 | 3.58 |
| $\mathbb{M}^{\text{Learning}}_{\text{Obj. Clust.}}$ | 9.37 | 5.91 | 2.52 | 1.72 | 52.4 | 37.4 | 650 | **2.64** |

**Fig. 10.** Average of the $D_{\mathcal{P}_F \to \mathcal{S}}$ performance indicator on all combinatorial problems. Note: naive $\mathbb{M}$IDEA could also have been abbreviated as $\mathbb{M}^{\text{Univariate}}_{\text{Obj. Clust}}$

| *Front Spread* **FS** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **EA** | $BT_1^{10}$ | $ZDT_4^{10}$ | $ZDT_6^{10}$ | $CTP_7^{10}$ | $BT_1^{100}$ | $ZDT_4^{100}$ | $ZDT_6^{100}$ | $CTP_7^{100}$ |
| SPEA$^{\text{UX}}$ | 225 | 51.4 | 5.22 | 44.9 | 2.06 | 692 | 1.85 | 733 |
| SPEA$^{\text{IX}}$ | 369 | 55.8 | 5.26 | 46.3 | 2.31 | 736 | 3.02 | 773 |
| NSGA-II$^{\text{UX}}$ | 179 | 3.60 | 1.09 | 1.76 | 0.413 | 35.2 | 0.756 | 29.3 |
| NSGA-II$^{\text{IX}}$ | 23.4 | 8.93 | 1.03 | 1.31 | 1.02 | 33.4 | 0.665 | 13.9 |
| $\mathbb{M}^{\text{UX}}_{\text{1 Cluster}}$ | 655 | 8.55 | 2.90 | 39.1 | 2.18 | 395 | 3.43 | 365 |
| $\mathbb{M}^{\text{IX}}_{\text{1 Cluster}}$ | 78.6 | 2.46 | 1.92 | 1.41 | 2.27 | 94.0 | 1.40 | 88.6 |
| $\mathbb{M}^{\text{UX}}_{\text{Par. Clust.}}$ | 357 | 12.2 | 5.05 | 4.85 | 2.11 | 384 | 3.10 | 345 |
| $\mathbb{M}^{\text{IX}}_{\text{Par. Clust.}}$ | 199 | 2.45 | 5.33 | 1.66 | 2.31 | 129 | 1.53 | 93.1 |
| $\mathbb{M}^{\text{UX}}_{\text{Obj. Clust.}}$ | 685 | 40.8 | 4.11 | 41.8 | 2.15 | 740 | 4.75 | 737 |
| $\mathbb{M}^{\text{IX}}_{\text{Obj. Clust.}}$ | 262 | 3.38 | 3.94 | 58.9 | 2.29 | 359 | 2.30 | 371 |
| $\mathbb{M}^{\text{Univariate}}_{\text{1 Cluster}}$ | 293 | 70.8 | 1.15 | 84.7 | 1.82 | 393 | 0.180 | 347 |
| $\mathbb{M}^{\text{Learning}}_{\text{1 Cluster}}$ | $129 \cdot 10^1$ | 84.9 | 3.00 | 87.4 | 2.12 | 635 | 2.20 | 342 |
| $\mathbb{M}^{\text{Univariate}}_{\text{Par. Clust.}}$ | $508 \cdot 10^1$ | 24.0 | 2.47 | 28.8 | 2.19 | 231 | 0.05 | 306 |
| $\mathbb{M}^{\text{Learning}}_{\text{Par. Clust.}}$ | $112 \cdot 10^2$ | 142 | 5.15 | 116 | 1.91 | 577 | 7.01 | 588 |
| naive $\mathbb{M}$IDEA | $209 \cdot 10^1$ | 90.4 | **5.29** | 114 | 2.45 | 636 | **8.10** | 619 |
| $\mathbb{M}^{\text{Learning}}_{\text{Obj. Clust.}}$ | $\mathbf{164 \cdot 10^2}$ | **197** | 3.68 | **188** | **3.28** | $\mathbf{175 \cdot 10^1}$ | 3.97 | $\mathbf{183 \cdot 10^1}$ |

**Fig. 11.** Average of the **FS** performance indicator on all real-valued problems. Note: naive $\mathbb{M}$IDEA could also have been abbreviated as $\mathbb{M}^{\text{Univariate}}_{\text{Obj. Clust.}}$

| *Front Spread* **FS** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **EA** | $MS^{100}$ | $KN^{100}$ | $SC^{100}$ | $MST^{105}$ | $MS^{1000}$ | $KN^{1000}$ | $SC^{1000}$ | $MST^{1035}$ |
| SPEA$^{\text{UX}}$ | 116 | 69.5 | **64.6** | 30.6 | 288 | 254 | 631 | 52.1 |
| SPEA$^{\text{IX}}$ | 126 | 82.6 | 50.1 | **32.3** | 399 | 308 | **636** | 50.8 |
| NSGA-II$^{\text{UX}}$ | 120 | 78.3 | 17.3 | 26.3 | 370 | 288 | 144 | 33.7 |
| NSGA-II$^{\text{IX}}$ | 129 | 79.0 | 12.8 | 23.9 | 364 | 291 | 107 | 36.1 |
| $\mathbb{M}^{\text{UX}}_{\text{1 Cluster}}$ | 132 | 92.6 | 20.7 | 17.8 | 304 | 285 | 112 | 40.1 |
| $\mathbb{M}^{\text{IX}}_{\text{1 Cluster}}$ | 141 | 91.9 | 18.3 | 19.3 | 329 | 247 | 105 | 47.9 |
| $\mathbb{M}^{\text{UX}}_{\text{Par. Clust.}}$ | 129 | 90.8 | 20.1 | 18.4 | 265 | 289 | 125 | 40.7 |
| $\mathbb{M}^{\text{IX}}_{\text{Par. Clust.}}$ | 132 | 91.4 | 17.3 | 20.1 | 277 | 261 | 112 | 46.8 |
| $\mathbb{M}^{\text{UX}}_{\text{Obj. Clust.}}$ | 187 | 119 | 21.9 | 30.1 | 600 | 483 | 199 | 58.7 |
| $\mathbb{M}^{\text{IX}}_{\text{Obj. Clust.}}$ | 183 | 103 | 21.1 | 26.0 | 579 | 430 | 155 | 58.0 |
| $\mathbb{M}^{\text{Univariate}}_{\text{1 Cluster}}$ | 79.2 | 43.3 | 16.1 | 16.9 | 122 | 98.4 | 10.8 | 22.7 |
| $\mathbb{M}^{\text{Learning}}_{\text{1 Cluster}}$ | 143 | 90.0 | 18.2 | 19.7 | 124 | 214 | 135 | 37.5 |
| $\mathbb{M}^{\text{Univariate}}_{\text{Par. Clust.}}$ | 90.8 | 57.4 | 16.7 | 16.7 | 72.9 | 85.2 | 10.7 | 23.1 |
| $\mathbb{M}^{\text{Learning}}_{\text{Par. Clust.}}$ | 143 | 106 | 18.4 | 20.5 | 124 | 109 | 19.2 | 32.1 |
| naive $\mathbb{M}$IDEA | **192** | 116 | 27.6 | 32.1 | 665 | 503 | 313 | **65.2** |
| $\mathbb{M}^{\text{Learning}}_{\text{Obj. Clust.}}$ | 191 | **125** | 22.4 | 30.3 | **784** | **512** | 66.2 | 60.2 |

**Fig. 12.** Average of the **FS** performance indicator on all combinatorial problems. Note: naive $\mathbb{M}$IDEA could also have been abbreviated as $\mathbb{M}^{\text{Univariate}}_{\text{Obj. Clust.}}$

| *Front Occupation* **FO** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **EA** | $BT_1^{10}$ | $ZDT_4^{10}$ | $ZDT_6^{10}$ | $CTP_7^{10}$ | $BT_1^{100}$ | $ZDT_4^{100}$ | $ZDT_6^{100}$ | $CTP_7^{100}$ |
| SPEA$^{\text{UX}}$ | **60.9** | 99.0 | 50.0 | 43.5 | 49.8 | 27.6 | 18.7 | 26.7 |
| SPEA$^{\text{IX}}$ | 38.7 | **187** | 49.6 | 43.2 | 48.8 | 27.4 | 29.3 | 26.8 |
| NSGA-II$^{\text{UX}}$ | 5.42 | 59.7 | 47.5 | 59.3 | 100 | 5.80 | 6.00 | 4.00 |
| NSGA-II$^{\text{IX}}$ | 29.5 | 32.7 | 31.2 | 9.98 | 75.0 | 5.00 | 6.60 | 3.00 |
| $\mathbb{M}^{\text{UX}}_{\text{1 Cluster}}$ | 9.92 | 41.7 | 8.06 | 9.00 | 14.4 | 12.8 | 14.4 | 12.6 |
| $\mathbb{M}^{\text{IX}}_{\text{1 Cluster}}$ | 13.4 | 30.3 | 6.52 | 11.9 | 16.5 | 7.10 | 6.64 | 5.94 |
| $\mathbb{M}^{\text{UX}}_{\text{Par. Clust.}}$ | 7.46 | 25.4 | 8.02 | 18.2 | 15.4 | 12.9 | 15.2 | 12.4 |
| $\mathbb{M}^{\text{IX}}_{\text{Par. Clust.}}$ | 9.78 | 24.7 | 7.80 | 11.9 | 17.5 | 7.20 | 8.12 | 6.68 |
| $\mathbb{M}^{\text{UX}}_{\text{Obj. Clust.}}$ | 13.9 | 10.0 | 8.48 | 8.62 | 19.1 | 20.0 | 19.6 | 21.7 |
| $\mathbb{M}^{\text{IX}}_{\text{Obj. Clust.}}$ | 9.94 | 31.4 | 7.32 | 15.6 | 17.4 | 12.2 | 9.76 | 12.2 |
| $\mathbb{M}^{\text{Univariate}}_{\text{1 Cluster}}$ | 5.74 | 6.88 | 4.90 | 4.14 | 36.7 | 6.9 | 2.55 | 3.20 |
| $\mathbb{M}^{\text{Learning}}_{\text{1 Cluster}}$ | 6.06 | 8.36 | **258** | 4.96 | 13.1 | 5.25 | **369** | 3.75 |
| $\mathbb{M}^{\text{Univariate}}_{\text{Par. Clust.}}$ | 29.6 | 98.8 | 30.0 | **82.0** | 33.4 | 69.4 | 3.70 | 18.3 |
| $\mathbb{M}^{\text{Learning}}_{\text{Par. Clust.}}$ | 52.7 | 65.4 | 104 | 69.2 | **149** | 105 | 92.0 | **112** |
| naive $\mathbb{M}$IDEA | 12.5 | 68.7 | 56.3 | 34.0 | 64.5 | **106** | 27.7 | 78.9 |
| $\mathbb{M}^{\text{Learning}}_{\text{Obj. Clust.}}$ | 30.1 | 26.4 | 197 | 32.1 | 111 | 50.8 | 163 | 43.0 |

**Fig. 13.** Average of the **FO** performance indicator on all real-valued problems. Note: naive $\mathbb{M}$IDEA could also have been abbreviated as $\mathbb{M}^{\text{Univariate}}_{\text{Obj. Clust.}}$

| *Front Occupation* **FO** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **EA** | $MS^{100}$ | $KN^{100}$ | $SC^{100}$ | $MST^{105}$ | $MS^{1000}$ | $KN^{1000}$ | $SC^{1000}$ | $MST^{1035}$ |
| SPEA$^{\text{UX}}$ | 46.8 | 46.5 | **25.1** | 42.8 | 49.4 | 49.5 | 26.2 | 48.8 |
| SPEA$^{\text{IX}}$ | 46.1 | **77.6** | 24.3 | **93.2** | 49.9 | 49.7 | **26.5** | **95.0** |
| NSGA-II$^{\text{UX}}$ | 33.5 | 35.5 | 12.0 | 32.3 | 35.4 | 33.1 | 7.50 | 64.7 |
| NSGA-II$^{\text{IX}}$ | 41.1 | 35.4 | 6.80 | 24.5 | 42.0 | 36.4 | 7.20 | 64.8 |
| $\mathbb{M}^{\text{UX}}_{\text{1 Cluster}}$ | 100 | 28.1 | 11.3 | 20.8 | 197 | 46.8 | 12.4 | 25.4 |
| $\mathbb{M}^{\text{IX}}_{\text{1 Cluster}}$ | 130 | 43.8 | 14.9 | 20.3 | 212 | 43.1 | 16.1 | 38.5 |
| $\mathbb{M}^{\text{UX}}_{\text{Par. Clust.}}$ | 112 | 32.2 | 10.6 | 23.7 | 171 | 46.9 | 13.0 | 26.0 |
| $\mathbb{M}^{\text{IX}}_{\text{Par. Clust.}}$ | 136 | 50.2 | 13.2 | 24.5 | 179 | 44.1 | 17.8 | 37.1 |
| $\mathbb{M}^{\text{UX}}_{\text{Obj. Clust.}}$ | **165** | 48.4 | 11.1 | 29.3 | 269 | 78.1 | 15.0 | 44.2 |
| $\mathbb{M}^{\text{IX}}_{\text{Obj. Clust.}}$ | 160 | 61.1 | 16.2 | 33.5 | 325 | 52.3 | 13.2 | 48.5 |
| $\mathbb{M}^{\text{Univariate}}_{\text{1 Cluster}}$ | 56.9 | 15.6 | 8.56 | 17.6 | 37.5 | 20.6 | 3.92 | 16.7 |
| $\mathbb{M}^{\text{Learning}}_{\text{1 Cluster}}$ | 105 | 37.5 | 10.0 | 20.9 | 48.5 | 64.2 | 19.2 | 61.0 |
| $\mathbb{M}^{\text{Univariate}}_{\text{Par. Clust.}}$ | 59.8 | 21.9 | 8.87 | 16.6 | 85.4 | 15.9 | 4.90 | 16.3 |
| $\mathbb{M}^{\text{Learning}}_{\text{Par. Clust.}}$ | 104 | 40.9 | 9.60 | 20.9 | 48.5 | 47.5 | 8.67 | 58.7 |
| naive $\mathbb{M}$IDEA | 147 | 36.1 | 11.9 | 25.9 | 129 | 65.1 | 16.1 | 41.9 |
| $\mathbb{M}^{\text{Learning}}_{\text{Obj. Clust.}}$ | 143 | 51.8 | 10.0 | 25.3 | **411** | **101** | 8.0 | 65.9 |

**Fig. 14.** Average of the **FO** performance indicator on all combinatorial problems. Note: naive $\mathbb{M}$IDEA could also have been abbreviated as $\mathbb{M}^{\text{Univariate}}_{\text{Obj. Clust.}}$

| | *Population Size* $n$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **EA** | $BT_1^{10}$ | $ZDT_4^{10}$ | $ZDT_6^{10}$ | $CTP_7^{10}$ | $BT_1^{100}$ | $ZDT_4^{100}$ | $ZDT_6^{100}$ | $CTP_7^{100}$ |
| SPEA$^{\mathrm{UX}}$ | 50 | 50 | 25 | 25 | 25 | 25 | 25 | 25 |
| SPEA$^{\mathrm{1X}}$ | 25 | 100 | 25 | 25 | 25 | 25 | 25 | 25 |
| NSGA-II$^{\mathrm{UX}}$ | 200 | 200 | 100 | 100 | 100 | 200 | 200 | 150 |
| NSGA-II$^{\mathrm{1X}}$ | 200 | 375 | 75 | 300 | 75 | 200 | 150 | 300 |
| $\mathbb{M}^{\mathrm{UX}}_{\text{1 Cluster}}$ | 75 | 100 | 25 | 25 | 100 | 125 | 200 | 125 |
| $\mathbb{M}^{\mathrm{1X}}_{\text{1 Cluster}}$ | 100 | 450 | 25 | 300 | 125 | 325 | 100 | 175 |
| $\mathbb{M}^{\mathrm{UX}}_{\text{Par. Clust.}}$ | 175 | 75 | 25 | 100 | 75 | 125 | 150 | 175 |
| $\mathbb{M}^{\mathrm{1X}}_{\text{Par. Clust.}}$ | 125 | 450 | 25 | 275 | 150 | 175 | 100 | 175 |
| $\mathbb{M}^{\mathrm{UX}}_{\text{Obj. Clust.}}$ | 225 | 25 | 25 | 25 | 125 | 200 | 200 | 300 |
| $\mathbb{M}^{\mathrm{1X}}_{\text{Obj. Clust.}}$ | 150 | 475 | 25 | 725 | 125 | 200 | 100 | 150 |
| $\mathbb{M}^{\mathrm{Univariate}}_{\text{1 Cluster}}$ | 150 | 50 | 75 | 50 | 100 | 75 | 375 | 50 |
| $\mathbb{M}^{\mathrm{Learning}}_{\text{1 Cluster}}$ | 150 | 75 | 425 | 75 | 175 | 100 | 700 | 100 |
| $\mathbb{M}^{\mathrm{Univariate}}_{\text{Par. Clust.}}$ | 175 | 125 | 175 | 125 | 225 | 150 | 450 | 150 |
| $\mathbb{M}^{\mathrm{Learning}}_{\text{Par. Clust.}}$ | 400 | 250 | 275 | 250 | 200 | 150 | 550 | 125 |
| naive $\mathbb{M}$ID$\mathbb{E}$A | 275 | 125 | 200 | 125 | 250 | 200 | 800 | 200 |
| $\mathbb{M}^{\mathrm{Learning}}_{\text{Obj. Clust.}}$ | 450 | 200 | 250 | 150 | 225 | 300 | 400 | 250 |

**Fig. 15.** Population sizes used for the real-valued problems. Note: naive $\mathbb{M}$ID$\mathbb{E}$A could also have been abbreviated as $\mathbb{M}^{\mathrm{Univariate}}_{\text{Obj. Clust.}}$

| | *Population Size* $n$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **EA** | $MS^{100}$ | $KN^{100}$ | $SC^{100}$ | $MST^{105}$ | $MS^{1000}$ | $KN^{1000}$ | $SC^{1000}$ | $MST^{1035}$ |
| SPEA$^{\mathrm{UX}}$ | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 |
| SPEA$^{\mathrm{1X}}$ | 25 | 50 | 25 | 125 | 25 | 25 | 25 | 50 |
| NSGA-II$^{\mathrm{UX}}$ | 350 | 325 | 300 | 200 | 200 | 200 | 200 | 250 |
| NSGA-II$^{\mathrm{1X}}$ | 100 | 325 | 250 | 200 | 150 | 250 | 150 | 200 |
| $\mathbb{M}^{\mathrm{UX}}_{\text{1 Cluster}}$ | 575 | 350 | 550 | 1250 | 775 | 775 | 325 | 1000 |
| $\mathbb{M}^{\mathrm{1X}}_{\text{1 Cluster}}$ | 550 | 400 | 300 | 1200 | 800 | 625 | 500 | 1050 |
| $\mathbb{M}^{\mathrm{UX}}_{\text{Par. Clust.}}$ | 500 | 525 | 500 | 2600 | 650 | 775 | 350 | 1100 |
| $\mathbb{M}^{\mathrm{1X}}_{\text{Par. Clust.}}$ | 525 | 575 | 425 | 2375 | 650 | 650 | 475 | 1200 |
| $\mathbb{M}^{\mathrm{UX}}_{\text{Obj. Clust.}}$ | 550 | 425 | 550 | 1975 | 750 | 775 | 775 | 1800 |
| $\mathbb{M}^{\mathrm{1X}}_{\text{Obj. Clust.}}$ | 475 | 425 | 825 | 1400 | 825 | 500 | 650 | 1750 |
| $\mathbb{M}^{\mathrm{Univariate}}_{\text{1 Cluster}}$ | 700 | 200 | 450 | 5000 | 1375 | 800 | 225 | 800 |
| $\mathbb{M}^{\mathrm{Learning}}_{\text{1 Cluster}}$ | 850 | 700 | 700 | 1850 | 1350 | 850 | 500 | 1600 |
| $\mathbb{M}^{\mathrm{Univariate}}_{\text{Par. Clust.}}$ | 750 | 600 | 525 | 7000 | 300 | 375 | 250 | 700 |
| $\mathbb{M}^{\mathrm{Learning}}_{\text{Par. Clust.}}$ | 1075 | 950 | 1050 | 1850 | 1350 | 700 | 700 | 2900 |
| naive $\mathbb{M}$ID$\mathbb{E}$A | 500 | 300 | 900 | 2500 | 875 | 750 | 900 | 1850 |
| $\mathbb{M}^{\mathrm{Learning}}_{\text{Obj. Clust.}}$ | 1000 | 925 | 1050 | 4000 | 1400 | 1500 | 1100 | 2350 |

**Fig. 16.** Population sizes used for the combinatorial problems. Note: naive $\mathbb{M}$ID$\mathbb{E}$A could also have been abbreviated as $\mathbb{M}^{\mathrm{Univariate}}_{\text{Obj. Clust.}}$

| $D_{\mathcal{P}_F \to \mathcal{S}}$ | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Statistically Significant Improvement Matrix* | SPEA$^{UX}$ | SPEA$^{IX}$ | NSGA-II$^{UX}$ | NSGA-II$^{IX}$ | $\mathbb{M}^{UX}_{1\text{ Cluster}}$ | $\mathbb{M}^{IX}_{1\text{ Cluster}}$ | $\mathbb{M}^{UX}_{\text{Par. Clust.}}$ | $\mathbb{M}^{IX}_{\text{Par. Clust.}}$ | $\mathbb{M}^{UX}_{\text{Obj. Clust.}}$ | $\mathbb{M}^{IX}_{\text{Obj. Clust.}}$ | $\mathbb{M}^{\text{Univariate}}_{1\text{ Cluster}}$ | $\mathbb{M}^{\text{Learning}}_{1\text{ Cluster}}$ | $\mathbb{M}^{\text{Univariate}}_{\text{Par. Clust.}}$ | $\mathbb{M}^{\text{Learning}}_{\text{Par. Clust.}}$ | naive $\mathbb{MIDEA}$ | $\mathbb{M}^{\text{Learning}}_{\text{Obj. Clust.}}$ | **Sum** |
| SPEA$^{UX}$ | 0 | -8 | -7 | -4 | -9 | 3 | -10 | 3 | -11 | 3 | 4 | -4 | -2 | -1 | -12 | -7 | -62 |
| SPEA$^{IX}$ | 8 | 0 | -7 | -3 | -8 | 5 | -8 | 6 | -9 | 5 | 6 | -3 | -1 | 0 | -12 | -8 | -29 |
| NSGA-II$^{UX}$ | 7 | 7 | 0 | 2 | -3 | 5 | -4 | 4 | -10 | 4 | 6 | -2 | -2 | 3 | -13 | -7 | -3 |
| NSGA-II$^{IX}$ | 4 | 3 | -2 | 0 | -1 | 11 | -1 | 9 | -6 | 12 | 5 | -5 | 0 | -1 | -11 | -8 | 9 |
| $\mathbb{M}^{UX}_{1\text{ Cluster}}$ | 9 | 8 | 3 | 1 | 0 | 3 | 6 | 3 | -8 | 2 | 11 | 7 | 6 | 6 | -7 | -6 | 44 |
| $\mathbb{M}^{IX}_{1\text{ Cluster}}$ | -3 | -5 | -5 | -11 | -3 | 0 | -1 | -4 | -7 | -8 | -4 | -7 | -10 | -8 | -11 | -11 | -98 |
| $\mathbb{M}^{UX}_{\text{Par. Clust.}}$ | 10 | 8 | 4 | 1 | -6 | 1 | 0 | 2 | -7 | 1 | 11 | 6 | 6 | 5 | -9 | -8 | 25 |
| $\mathbb{M}^{IX}_{\text{Par. Clust.}}$ | -3 | -6 | -4 | -9 | -3 | 4 | -2 | 0 | -7 | 3 | -4 | -7 | -10 | -8 | -10 | -11 | -77 |
| $\mathbb{M}^{UX}_{\text{Obj. Clust.}}$ | 11 | 9 | 10 | 6 | 8 | 7 | 7 | 7 | 0 | 6 | 8 | 6 | 2 | 8 | -3 | 1 | 93 |
| $\mathbb{M}^{IX}_{\text{Obj. Clust.}}$ | -3 | -5 | -4 | -12 | -2 | 8 | -1 | -3 | -6 | 0 | -4 | -8 | -10 | -8 | -11 | -11 | -80 |
| $\mathbb{M}^{\text{Univariate}}_{1\text{ Cluster}}$ | -4 | -6 | -6 | -5 | -11 | 4 | -11 | 4 | -8 | 4 | 0 | -10 | -4 | -7 | -12 | -11 | -83 |
| $\mathbb{M}^{\text{Learning}}_{1\text{ Cluster}}$ | 4 | 3 | 2 | 5 | -7 | 7 | -6 | 7 | -6 | 8 | 10 | 0 | 1 | 2 | -8 | -7 | 15 |
| $\mathbb{M}^{\text{Univariate}}_{\text{Par. Clust.}}$ | 2 | 1 | 2 | 0 | -6 | 10 | -6 | 10 | -2 | 10 | 4 | -1 | 0 | -1 | -6 | -7 | 10 |
| $\mathbb{M}^{\text{Learning}}_{\text{Par. Clust.}}$ | 1 | 0 | -3 | 1 | -6 | 8 | -5 | 8 | -8 | 8 | 7 | -2 | 1 | 0 | -8 | -6 | -4 |
| naive $\mathbb{MIDEA}$ | 12 | 12 | 13 | 11 | 7 | 11 | 9 | 10 | 3 | 11 | 12 | 8 | 6 | 8 | 0 | 5 | **138** |
| $\mathbb{M}^{\text{Learning}}_{\text{Obj. Clust.}}$ | 7 | 8 | 7 | 8 | 6 | 11 | 8 | 11 | -1 | 11 | 11 | 7 | 7 | 6 | -5 | 0 | 102 |

**Fig. 17.** Number of times an improvement was found to be statistically significant in the $D_{\mathcal{P}_F \to \mathcal{S}}$ performance indicator, summed over all tested problems. The numbers in a single row indicate the summed number of significantly better or worse results compared to the algorithms in the different columns. Note: naive $\mathbb{MIDEA}$ could also have been abbreviated as $\mathbb{M}^{\text{Univariate}}_{\text{Obj. Clust.}}$

the more powerful diversity exploration and preservation in $\mathbb{MIDEA}$. As the dimensionality of the problem goes up, the parameter space (i.e. the search space) becomes larger. In the case of the binary combinatorial problems, the number of solutions in the objective space becomes larger as well. If clustering in the objective space is used in $\mathbb{MIDEA}$, better results are obtained on average as the dimensionality of the problem increases. In Fig. 20 the Pareto fronts over 50 runs for a selection of algorithms are plotted on one problem from each problem class and dimensionality. The better diversity preservation and proper distribution of the points along the front can be seen clearly for the problems of larger dimensionality. For the lower dimensionality problems, better diversity preservation can also be observed, which is most exemplified by the fact that $\mathbb{MIDEA}$ obtains non-dominated solutions at the outer ends of the front for the knapsack problem with $l = 100$.

| *Front Spread* **FS** | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Statistically Significant Improvement Matrix* | SPEA$^{UX}$ | SPEA$^{IX}$ | NSGA-II$^{UX}$ | NSGA-II$^{IX}$ | $\mathbb{M}_{1\text{ Cluster}}^{UX}$ | $\mathbb{M}_{1\text{ Cluster}}^{IX}$ | $\mathbb{M}_{Par.\text{ Clust.}}^{UX}$ | $\mathbb{M}_{Par.\text{ Clust.}}^{IX}$ | $\mathbb{M}_{Obj.\text{ Clust.}}^{UX}$ | $\mathbb{M}_{Obj.\text{ Clust.}}^{IX}$ | $\mathbb{M}_{1\text{ Cluster}}^{Univariate}$ | $\mathbb{M}_{1\text{ Cluster}}^{Learning}$ | $\mathbb{M}_{Par.\text{ Clust.}}^{Univariate}$ | $\mathbb{M}_{Par.\text{ Clust.}}^{Learning}$ | naive $\mathbb{MIDEA}$ | $\mathbb{M}_{Obj.\text{ Clust.}}^{Learning}$ | **Sum** |
| SPEA$^{UX}$ | 0 | -7 | 8 | 8 | 2 | 13 | 5 | 11 | -3 | 11 | 11 | 3 | 13 | 2 | -7 | -9 | 61 |
| SPEA$^{IX}$ | 7 | 0 | 16 | 14 | 8 | 15 | 10 | 13 | 0 | 13 | 11 | 5 | 13 | 3 | -6 | -7 | 115 |
| NSGA-II$^{UX}$ | -8 | -16 | 0 | 5 | -8 | 3 | -8 | 3 | -15 | 1 | 3 | -8 | 1 | -6 | -16 | -14 | -83 |
| NSGA-II$^{IX}$ | -8 | -14 | -5 | 0 | -8 | 1 | -8 | 1 | -16 | -1 | 1 | -9 | 0 | -6 | -16 | -14 | -102 |
| $\mathbb{M}_{1\text{ Cluster}}^{UX}$ | -2 | -8 | 8 | 8 | 0 | 12 | 3 | 11 | -12 | 5 | 9 | -2 | 10 | -4 | -16 | -12 | 10 |
| $\mathbb{M}_{1\text{ Cluster}}^{IX}$ | -13 | -15 | -3 | -1 | -12 | 0 | -12 | -2 | -13 | -6 | 0 | -11 | -4 | -7 | -15 | -14 | -128 |
| $\mathbb{M}_{Par.\text{ Clust.}}^{UX}$ | -5 | -10 | 8 | 8 | -3 | 12 | 0 | 12 | -11 | 8 | 9 | 0 | 8 | -3 | -15 | -11 | 7 |
| $\mathbb{M}_{Par.\text{ Clust.}}^{IX}$ | -11 | -13 | -3 | -1 | -11 | 2 | -12 | 0 | -11 | -6 | 0 | -9 | -5 | -7 | -15 | -13 | -115 |
| $\mathbb{M}_{Obj.\text{ Clust.}}^{UX}$ | 3 | 0 | 15 | 16 | 12 | 13 | 11 | 11 | 0 | 9 | 11 | 9 | 14 | 6 | -9 | -8 | 113 |
| $\mathbb{M}_{Obj.\text{ Clust.}}^{IX}$ | -11 | -13 | -1 | 1 | -5 | 6 | -8 | 6 | -9 | 0 | 2 | -8 | 2 | -8 | -16 | -12 | -74 |
| $\mathbb{M}_{1\text{ Cluster}}^{Univariate}$ | -11 | -11 | -3 | -1 | -9 | 0 | -9 | 0 | -11 | -2 | 0 | -11 | 1 | -13 | -15 | -16 | -111 |
| $\mathbb{M}_{1\text{ Cluster}}^{Learning}$ | -3 | -5 | 8 | 9 | 2 | 11 | 0 | 9 | -9 | 8 | 11 | 0 | 12 | -4 | -13 | -13 | 23 |
| $\mathbb{M}_{Par.\text{ Clust.}}^{Univariate}$ | -13 | -13 | -1 | 0 | -10 | 4 | -8 | 5 | -14 | -2 | -1 | -12 | 0 | -14 | -15 | -16 | -110 |
| $\mathbb{M}_{Par.\text{ Clust.}}^{Learning}$ | -2 | -3 | 6 | 6 | 4 | 7 | 3 | 7 | -6 | 8 | 13 | 4 | 14 | 0 | -8 | -11 | 42 |
| naive $\mathbb{MIDEA}$ | 7 | 6 | 16 | 16 | 16 | 15 | 15 | 15 | 9 | 16 | 15 | 13 | 15 | 8 | 0 | -2 | **180** |
| $\mathbb{M}_{Obj.\text{ Clust.}}^{Learning}$ | 9 | 7 | 14 | 14 | 12 | 14 | 11 | 13 | 8 | 12 | 16 | 13 | 16 | 11 | 2 | 0 | 172 |

**Fig. 18.** Number of times an improvement was found to be statistically significant in the FS performance indicator, summed over all tested problems. The numbers in a single row indicate the summed number of significantly better or worse results compared to the algorithms in the different columns. Note: naive $\mathbb{MIDEA}$ could also have been abbreviated as $\mathbb{M}_{Obj.\text{ Clust.}}^{Univariate}$

*Influence of Mixtures by Clustering the Objective Space*

The fact that the use of mixtures by clustering the objective space allows for enhanced diversity exploration and preservation, can also be observed by the difference between the spread obtained by $\mathbb{MIDEA}$ with crossover operators using only a single cluster versus the case in which on average four clusters are used. A wider spread of solutions is found when clustering in the objective space is enabled. Furthermore, although clustering in the parameter space is a powerful approach to enhance the learning of probabilistic models, it does not immediately lead to better results in multi-objective optimization.

*Influence of the Problem Structure Exploitation Capabilities of EDAs*

On the $BT_1$ problem, modelling interactions in $\mathbb{MIDEA}$ clearly leads to better results than those obtained by the other MOEAs. Thus, exploiting interactions can be beneficial in multi-objective optimization. For the $BT_1$ problem with $l = 10$, if we allow for $5 \cdot 10^5$ evaluations, the $\mathbb{MIDEA}$ variant that learns Bayesian factorizations is even

| *Front Occupation* **FO** | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Statistically Significant Improvement Matrix* | SPEA$^{UX}$ | SPEA$^{1X}$ | NSGA-II$^{UX}$ | NSGA-II$^{1X}$ | $\mathbb{M}^{UX}_{1\text{ Cluster}}$ | $\mathbb{M}^{1X}_{1\text{ Cluster}}$ | $\mathbb{M}^{UX}_{\text{Par. Clust.}}$ | $\mathbb{M}^{1X}_{\text{Par. Clust.}}$ | $\mathbb{M}^{UX}_{\text{Obj. Clust.}}$ | $\mathbb{M}^{1X}_{\text{Obj. Clust.}}$ | $\mathbb{M}^{\text{Univariate}}_{1\text{ Cluster}}$ | $\mathbb{M}^{\text{Learning}}_{1\text{ Cluster}}$ | $\mathbb{M}^{\text{Univariate}}_{\text{Par. Clust.}}$ | $\mathbb{M}^{\text{Learning}}_{\text{Par. Clust.}}$ | naive $\mathbb{M}$IDEA | $\mathbb{M}^{\text{Learning}}_{\text{Obj. Clust.}}$ | **Sum** |
| SPEA$^{UX}$ | 0 | -3 | 10 | 12 | 11 | 16 | 11 | 16 | 7 | 16 | 14 | 5 | 7 | -3 | 1 | -4 | 116 |
| SPEA$^{1X}$ | 3 | 0 | 11 | 13 | 12 | 16 | 12 | 16 | 10 | 16 | 14 | 7 | 8 | -1 | 3 | 0 | 140 |
| NSGA-II$^{UX}$ | -10 | -11 | 0 | 2 | -1 | 8 | 1 | 6 | -3 | 8 | 11 | -3 | 2 | -8 | -4 | -6 | -8 |
| NSGA-II$^{1X}$ | -12 | -13 | -2 | 0 | -3 | 9 | -3 | 8 | -6 | 7 | 10 | -4 | 0 | -11 | -8 | -11 | -39 |
| $\mathbb{M}^{UX}_{1\text{ Cluster}}$ | -11 | -12 | 1 | 3 | 0 | 10 | 0 | 10 | -10 | 9 | 14 | 2 | 3 | -7 | -14 | -10 | -12 |
| $\mathbb{M}^{1X}_{1\text{ Cluster}}$ | -16 | -16 | -8 | -9 | -10 | 0 | -11 | -4 | -11 | -8 | -2 | -5 | -14 | -16 | -15 | -15 | -160 |
| $\mathbb{M}^{UX}_{\text{Par. Clust.}}$ | -11 | -12 | -1 | 3 | 0 | 11 | 0 | 10 | -10 | 8 | 14 | 2 | 3 | -5 | -14 | -12 | -14 |
| $\mathbb{M}^{1X}_{\text{Par. Clust.}}$ | -16 | -16 | -6 | -8 | -10 | 4 | -10 | 0 | -12 | -4 | -1 | -5 | -14 | -16 | -16 | -15 | -145 |
| $\mathbb{M}^{UX}_{\text{Obj. Clust.}}$ | -7 | -10 | 3 | 6 | 10 | 11 | 10 | 12 | 0 | 12 | 14 | 7 | 3 | -2 | -2 | -8 | 59 |
| $\mathbb{M}^{1X}_{\text{Obj. Clust.}}$ | -16 | -16 | -8 | -7 | -9 | 8 | -8 | 4 | -12 | 0 | -2 | -4 | -13 | -16 | -16 | -15 | -130 |
| $\mathbb{M}^{\text{Univariate}}_{1\text{ Cluster}}$ | -14 | -14 | -11 | -10 | -14 | 2 | -14 | 1 | -14 | 2 | 0 | -12 | -7 | -16 | -16 | -16 | -153 |
| $\mathbb{M}^{\text{Learning}}_{1\text{ Cluster}}$ | -5 | -7 | 3 | 4 | -2 | 5 | -2 | 5 | -7 | 4 | 12 | 0 | 2 | -3 | -6 | -8 | -5 |
| $\mathbb{M}^{\text{Univariate}}_{\text{Par. Clust.}}$ | -7 | -8 | -2 | 0 | -3 | 14 | -3 | 14 | -3 | 13 | 7 | -2 | 0 | -11 | -10 | -9 | -10 |
| $\mathbb{M}^{\text{Learning}}_{\text{Par. Clust.}}$ | 3 | 1 | 8 | 11 | 7 | 16 | 5 | 16 | 2 | 16 | 16 | 3 | 11 | 0 | 2 | -2 | 115 |
| naive $\mathbb{M}$IDEA | -1 | -3 | 4 | 8 | 14 | 15 | 14 | 16 | 2 | 16 | 16 | 6 | 10 | -2 | 0 | -3 | 112 |
| $\mathbb{M}^{\text{Learning}}_{\text{Obj. Clust.}}$ | 4 | 0 | 6 | 11 | 10 | 15 | 12 | 15 | 8 | 15 | 16 | 8 | 9 | 2 | 3 | 0 | **134** |

**Fig. 19.** Number of times an improvement was found to be statistically significant in the FO performance indicator, summed over all tested problems. The numbers in a single row indicate the summed number of significantly better or worse results compared to the algorithms in the different columns. Note: naive $\mathbb{M}$IDEA could also have been abbreviated as $\mathbb{M}^{\text{Univariate}}_{\text{Obj. Clust.}}$

capable of finding near optimal solutions whereas the other MOEAs were observed not to be able to produce comparable results. Furthermore, if we compare the results of the $\mathbb{M}$IDEA without clustering and with learning interactions with the $\mathbb{M}$IDEA without clustering and also without learning interactions (i.e. $\mathbb{M}^{\text{Learning}}_{1\text{ Cluster}}$ vs. $\mathbb{M}^{\text{Univariate}}_{1\text{ Cluster}}$), exploiting interactions often leads to better results and thus enhances the quality of the multi-objective search process. However, the same can be said for clustering the objective space in general. Moreover, the much cheaper operation of clustering the objective space can lead to significant improvements, regardless of the type of recombination used inside each cluster. Concordantly, the naive $\mathbb{M}$IDEA in which objective clustering is used obtains good results overall. In fact, summarized over all problems, the naive $\mathbb{M}$IDEA is arguably the best algorithm that we have tested. Moreover, the naive $\mathbb{M}$IDEA runs quickly, even for problems with many variables. Hence, learning dependencies between a problems' variables does not necessarily lead to advanced information about the trade-off in objective space that is the most important in multi-objective optimization problems. Clustering the objective space on the other hand does seem to help directly.

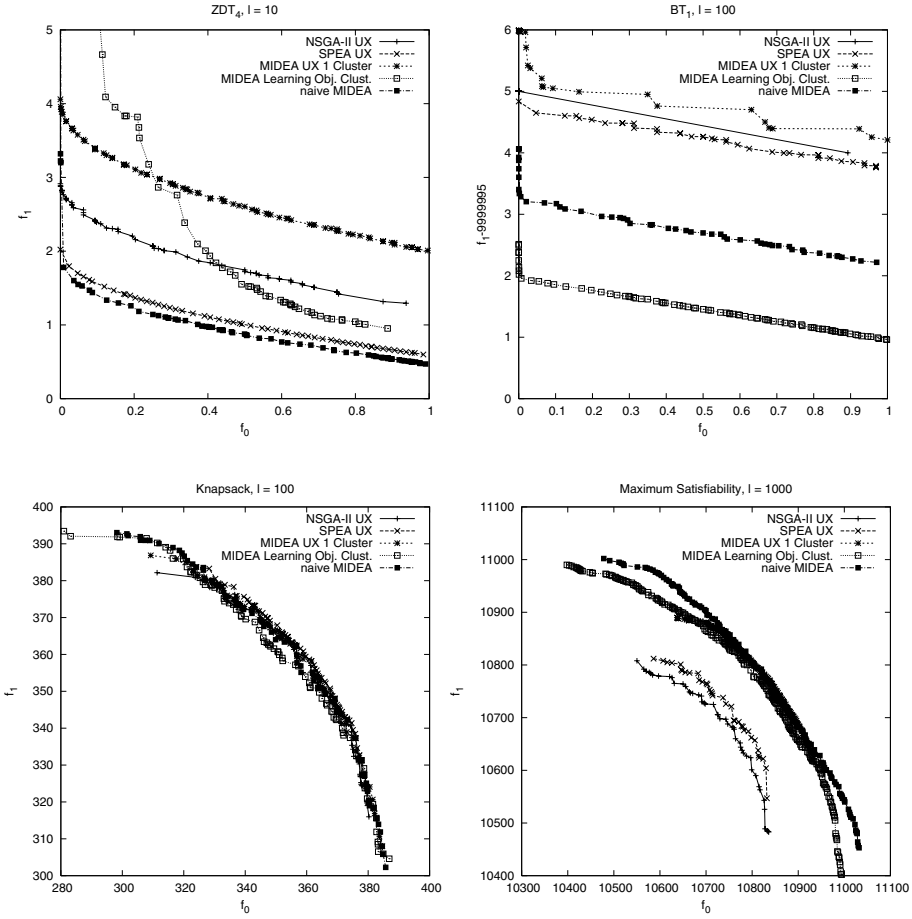


**Fig. 20.** Pareto fronts over 50 runs on a few of the tested problems. For clarity only a selection of all tested algorithms is shown. Note: naive $\mathbb{M}$IDE$\mathbb{A}$ could also have been written as MIDEA Univariate Obj. Clust.

Using more advanced factorizations to further exploit a problem's structure in the form of dependencies between a problem's variables can lead to the generation of more solutions on a less preferred front. Although such an approximation set is a result that can be found more efficiently by estimating involved probability distributions instead of using classical recombination operators, such a result is intuitively less desirable. More research is required to investigate the issue of exploiting dependencies between a problem's variables in an EDA for multi-objective optimization further. On the one hand it would be interesting to attempt to overcome this problem and ensure that the added complexity of the inductive capabilities of estimating probability distributions results in a more effective exploration towards the Pareto optimal front. On the other hand it would be interesting to investigate what type of

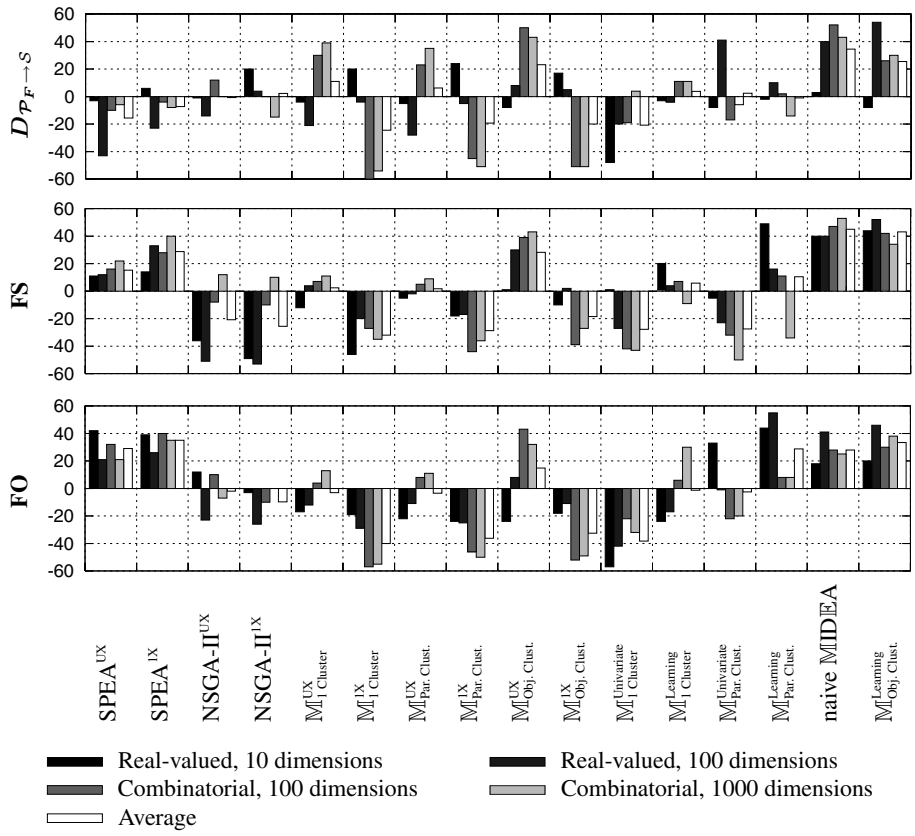


**Fig. 21.** A summary of the results of the statistical hypothesis tests performed for each pair of algorithms. For each algorithm, the sum of the outcome of the statistical hypothesis tests is shown for the real-valued problems and the combinatorial problems for each dimensionality separately. Furthermore, the average of these values is also shown, which serves as a global indicator of the performance of an algorithm relative to the other tested algorithms. Note: naive 𝕄IDEA could also have been abbreviated as $\mathbb{M}^{Univariate}_{Obj.\ Clust.}$

(real-world) multi-objective optimization problems can be solved more efficiently using 𝕄IDEA instances because of difficulties such as non-linear dependencies between the problem variables.

*The Influence of $\delta$*

In our benchmarks, we have picked a specific value for $\delta$. However, the $\delta$ parameter is a unique parameter that determines the balance between non-domination selection pressure and diversity preservation selection pressure. Although we acknowledge the influence of this parameter, we find it outside the scope of this chapter for an

in-depth discussion. We refer the interested reader to existing literature regarding the influence of $\delta$ [5].

### 4.5 Practitioner's Summary

Our experimental results indicate that clustering the objective space leads to superior MOEAs. For EDAs this implies that constructing mixture probability distributions in $\mathbb{MIDEA}$s based on geometric aspects of the objective space is a good approach. This makes the naive $\mathbb{MIDEA}$ instance based on mixture probability distributions truly an effective and easy-to-use new tool for multi-objective optimization. Furthermore, NSGA-II is overall the most competitive. However, there is an added value to the use of $\mathbb{MIDEA}$ in that it is able to obtain and maintain a larger and more diverse Pareto front by parallel front exploration and diversity preserving selection. The experiments underline these results as the front spread (Figs. 11 and 12), front occupation (Figs. 13 and 14) and the global Pareto fronts in Fig. 20 indicate a better performance. This increased performance is also statistically significant, as can be seen in figures 18 and 19. The use of clustering to obtain mixture probability distributions clearly leads to a significant increase of performance in the preservation and exploration of diversity.

Overall, the naive $\mathbb{MIDEA}$ is a very good MOEA that could be applied to real-world problems. We suggest setting $\delta \in [1; 1\frac{1}{2}]$ and to first use simple factorizations such as the univariate factorization. If more time and function evaluations are available, more complex factorizations can be used as well. An implementation of the naive $\mathbb{MIDEA}$ in C is available for download from the website of the first author.

## 5 Conclusions

In this paper we have presented the naive $\mathbb{MIDEA}$ for multi-objective optimization. The naive $\mathbb{MIDEA}$ clusters the selected solutions in the objective space, after which it estimates a univariate factorization in each cluster separately. New solutions are then drawn from the so-obtained mixture probability distribution. The naive $\mathbb{MIDEA}$ is a specific instance of the algorithmic framework $\mathbb{MIDEA}$ which is a general form of an EDA for multi-objective optimization in which a probabilistic model is learned. For the specific task of multi-objective optimization, the use of mixture distributions obtained by clustering the objective space has been observed to stimulate the desirable parallel exploration along the Pareto front. The naive $\mathbb{MIDEA}$ has only little computational overhead since clustering in the objective space can be done very fast as can the estimation of a univariate factorization. Furthermore, although no further exploitation of dependencies between a problem's variables is used in the naive $\mathbb{MIDEA}$, the results obtained for the naive $\mathbb{MIDEA}$ are already superior to results obtained with algorithms in which clustering the objective space is not used. Concluding, the naive $\mathbb{MIDEA}$ has been found to be a fast, easy-to-use and effective tool for multi-objective optimization.

## References


1. P. A. N. Bosman and D. Thierens. Advancing continuous IDEAs with mixture distributions and factorization selection metrics. In M. Pelikan and K. Sastry, editors, *Proceedings of the Optimization by Building and Using Probabilistic Models OBUPM Workshop at the GECCO-2001 Genetic and Evolutionary Computation Conference*, pp. 208–212. Morgan Kaufmann Publishers, 2001.
2. P. A. N. Bosman and D. Thierens. Exploiting gradient information in continuous iterated density estimation evolutionary algorithms. In B. Kröse, M. de Rijke, G. Schreiber, and M. van Someren, editors, *Proceedings of the 13th Belgium-Netherlands Artificial Intelligence Conference BNAIC'01*, pp. 69–76, 2001.
3. P. A. N. Bosman and D. Thierens. Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms. *International Journal of Approximate Reasoning*, 31:259–289, 2002.
4. P. A. N. Bosman and D. Thierens. Permutation optimization by iterated estimation of random keys marginal product factorizations. In J. J. Merelo, P. Adamidis, H.-G. Beyer, J.-J. Fernández-Villicañas, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VII*, pp. 331–340, Berlin, 2002. Springer-Verlag.
5. P. A. N. Bosman and D. Thierens. The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7:174–188, 2003.
6. P.A.N. Bosman and D. Thierens. A thorough documentation of obtained results on real-valued continuous and combinatorial multi-objective optimization problems using diversity preserving mixture-based iterated density estimation evolutionary algorithms. Technical report UU-CS-2002–52, Institute of Information and Computing Sciences, Utrecht University, Utrecht, 2002.
7. W. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.
8. C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.
9. C. A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems. An International Journal*, 1(3):269–308, 1999.
10. A. P. Dawid and S. L. Lauritzen. Hyper Markov laws in the statistical analysis of decomposable graphical models. *Annals of Statistics*, 21:1272–1317, 1993.
11. K. Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7(3):205–230, 1999.
12. K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VI*, pp. 849–858. Springer, 2000.
13. K. Deb, A. Pratap, and T. Meyarivan. Constrained test problems for multi-objective evolutionary optimization. In E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pp. 284–298, Berlin, 2001. Springer-Verlag.
14. A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistic Society*, Series B 39:1–38, 1977.
15. M. Ehrgott and X. Gandibleux. An annotated bibliography of multi-objective combinatorial optimization. Technical Report 62/2000, Fachbereich Mathematik, Universität Kaiserslautern, Kaiserslautern, 2000.

16. C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.
17. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, 1989.
18. G. Harik. Linkage learning via probabilistic modeling in the ECGA. IlliGAL Technical Report 99010, 1999.
19. G. Harik, F. Lobo, and D. E. Goldberg. The compact genetic algorithm. In *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pp. 523–528. IEEE Press, 1998.
20. J. A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, Inc., 1975.
21. M.G. Kendall and A. Stuart. *The Advanced Theory Of Statistics, Volume 2, Inference And Relationship*. Charles Griffin & Company Limited, 1967.
22. J. Knowles and D. Corne. On metrics for comparing non-dominated sets. In *Proceedings of the 2002 Congress on Evolutionary Computation CEC 2002*, pp. 666–674, Piscataway, New Jersey, 2002. IEEE Press.
23. P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation.* Kluwer Academic Publishers, 2001.
24. M. Laumanns, E. Zitzler, and L. Thiele. On the effects of archiving, elitism, and density based selection in evolutionary multi-objective optimization. In E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, editors, *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization - EMO 2001*, pp. 181–197. Springer-Verlag, 2001.
25. S. L. Lauritzen. *Graphical Models*. Clarendon Press, Oxford, 1996.
26. H. Mühlenbein and T. Mahnig. FDA - a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, 7:353–376, 1999.
27. H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. binary parameters. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN V*, pp. 178–187. Springer, 1998.
28. A. Ochoa, H. Mühlenbein, and M. Soto. A factorized distribution algorithm using single connected Bayesian networks. In M. Schoenauer et al., editor, *Parallel Problem Solving from Nature - PPSN VI*, pp. 787–796. Springer, 2000.
29. M. Pelikan and D. E. Goldberg. Escaping hierarchical traps with competent genetic algorithms. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the GECCO-2001 Genetic and Evolutionary Computation Conference*, pp. 511–518. Morgan Kaufmann, 2001.
30. M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20, 2002.
31. M. Pelikan, D. E. Goldberg, and K. Sastry. Bayesian optimization algorithm, decision graphs and Occam's razor. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the GECCO-2001 Genetic and Evolutionary Computation Conference*, pp. 519–526. Morgan Kaufmann, 2001.
32. R. Santana, A. Ochoa, and M. R. Soto. The mixture of trees factorized distribution algorithm. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the GECCO-2001 Genetic and Evolutionary Computation Conference*, pp. 543–550. Morgan Kaufmann, 2001.

33. M. Soto and A. Ochoa. A factorized distribution algorithm based on polytrees. In *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, pp. 232–237. IEEE Press, 2000.
34. 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, WPAFB, Ohio, 1999.
35. E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1999.
36. E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.
37. E. Zitzler, M. Laumanns, L. Thiele, C. M. Fonseca, and V. Grunert da Fonseca. Why quality assessment of multiobjective optimizers is difficult. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *Proceedings of the GECCO-2002 Genetic and Evolutionary Computation Conference*, pp. 666–674, San Francisco, California, 2002. Morgan Kaufmann.
38. 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.