Dynamic Resampling for Preference-based Evolutionary Multi-Objective Optimization of Stochastic Systems

Florian Siegmund[a], Amos H. C. Ng[a], and Kalyanmoy Deb[b]

[a] *School of Engineering, University of Skövde,*
Högskolevägen, 54128 Skövde, Sweden
{florian.siegmund, amos.ng}@his.se
[b] *Department of Electrical and Computer Engineering,*
Michigan State University,
428 S. Shaw Lane, East Lansing, MI 48824, USA
kdeb@egr.msu.edu

COIN Report Number 2015020

Abstract

In Preference-based Evolutionary Multi-objective Optimization, the goal is to find a diverse, but locally focused non-dominated front in a decision maker's area of interest, as close as possible to the true Pareto-front. The optimization can focus its efforts on the preferred area and achieve a better result. The modeled and simulated systems are often stochastic and a common method of handling the objective noise is Resampling. The given preference information allows for better resampling strategies to be defined, which can further improve the optimization result. In this paper, resampling strategies are proposed that base the sampling allocation on multiple factors, and thereby combine multiple resampling strategies proposed by the authors in Siegmund et al. (2013, 2015). These factors are, for example, the Pareto-rank of a solution and its distance to the decision maker's area of interest. In this paper we focus on Dynamic Resampling algorithms that do not consider the objective variance. The proposed hybrid Dynamic Resampling Strategies are evaluated on the Reference point-guided NSGA-II optimization algorithm (R-NSGA-II) with infeasible reference points on multi-objective benchmark functions and two stochastic production line problems.

Keywords:
Evolutionary multi-objective optimization, guided search, preference-based optimization, reference point, dynamic resampling, budget allocation, simulation-based optimization, decision support, stochastic systems, dynamic, resampling.

1. Introduction

In Preference-based Evolutionary Multi-objective Optimization, the decision maker is looking for a diverse, but locally focused non-dominated front in a preferred area of the objective space, as close as possible to the true Pareto-front. As solutions found outside of the area of interest are considered less important or even irrelevant, the optimization can focus its efforts on the preferred area and find the solutions that the decision maker is looking for more quickly, i.e. with fewer simulation runs. This is particularly important if the available time for optimization is limited, such as for many real-world applications (Lee & Chew, 2005). Focusing the search effort sets time resources free which, if not needed elsewhere, can be used to achieve a better result. Multi-objective evolutionary algorithms that can perform a guided search with preference information are, for example, the R-NSGA-II algorithm (Deb et al., 2006), Visual Steering (Stump et al., 2009), and interactive EMO based on progressively approximated value functions (Deb et al., 2010).

In Simulation-based Optimization, the modeled and simulated systems are often stochastic. To obtain an as exact as possible simulation of the system behavior, the stochastic characteristics are often built into the simulation models. When running the stochastic simulation, this expresses itself in deviating result values. Therefore, if the simulation is run multiple times for a selected parameter setting, the result value is slightly different for each simulation run. In the literature, this phenomenon of stochastic evaluation functions is sometimes called Noise, respectively Noisy Optimization (Bartz-Beielstein et al., 2007; Branke & Schmidt, 2004; Tan & Goh, 2008; Goh & Tan, 2009).

If an evolutionary optimization algorithm is run without countermeasure on an optimization problem with a noisy evaluation function, the performance will degrade, compared to the case if the true mean objective values were known. The algorithm will have incorrect knowledge about the solutions' quality and two cases of misjudgment will occur. The algorithm will see bad solutions as good and select them for the next generation while good solutions might be assessed as inferior and could be discarded. The performance can therefore be improved by increasing the knowledge of the algorithm regarding the solution quality.

Resampling is a way to reduce the uncertainty of the knowledge the algorithm has about the solutions. Resampling algorithms evaluate solutions several times to obtain an approximation of the expected objective values. This allows EMO algorithms to make better selection decisions, but it comes with a cost. As the modeled systems are usually complex, they require long simulation times, which limits the number of available solution evaluations. The additional solution evaluations needed to increase objective value knowledge are therefore not available for exploration of the objective space (Aizawa & Wah, 1993, 1994). This exploration vs. exploitation trade-off can be optimized, since the required knowledge about objective values varies between solutions. For example, in a dense, converged population, it is important to know the objective values well, whereas an algorithm which is about to explore the objective space is not harmed much by noisy objective values. Therefore, a resampling strategy which samples the solution carefully, according to the resampling need, can help an EMO algorithm achieve better results than a static resampling allocation.

Such a strategy is called Dynamic Resampling, DR. The resampling need varies between solutions. Solutions with a high objective variance need more evaluations to achieve a certain accuracy level. Dynamic Resampling algorithms for single-objective optimization problems that apply this strategy have been proposed by, for example, by Di Pietro et al. (2004); Di Pietro (2007) and Chen et al. (2008). Whereas the Optimal Computing Budget Allocation OCBA in Chen et al. (2008) requires a high implementation effort, the Standard Error Dynamic Resampling SEDR in Di Pietro et al. (2004) allows an easier application to an optimization algorithm. Other Multi-objective Dynamic Resampling algorithms have been proposed by Syberfeldt et al. (2010) and Lee et al. (2008, 2010); Chen & Lee (2010). MOPSA-EA (Syberfeldt et al., 2010) has limited applicability to only steady-state EMO algorithms and the MOCBA and EA approach in Chen & Lee (2010) requires an EA with high elitism and high implementation effort. DR algorithms which are integrated with an EA were proposed in Eskandari et al. (2007), Fieldsend & Everson (2005); Fieldsend (2015); Fieldsend & Everson (2015) and Park & Ryu (2011). The EA integration limits their applicability. They also do not consider the limited sampling budget as in our situation. In Siegmund et al. (2013) and Siegmund et al. (2015), we proposed easy to implement addon multi-objective DR algorithms which can be integrated with many preference-based EMO algorithms. Those DR algorithms were designed to add an increased sampling budget to solutions close to a preferred area (i.e., in this article close to a R-NSGA-II reference point), both keeping simplicity and general applicability in mind.

The paper is structured as follows. Section 2 introduces the R-NSGA-II algorithm, while Section 3 provides background information to the Dynamic Resampling method. Section 4 presents a classification of Dynamic Resampling algorithms and comments on the challenges of applying them in the scenario with a limited simulation budget. In Sections 5 and 6, different resampling algorithms for general EMO and for preference-based EMO are explained. Section 7 introduces the stochastic production line models, while Section 8 presents and defines new performance measures for preference-based EMO, used in this study. In Section 9, numerical experiments on benchmark functions and simulation optimization problems are performed. Furthermore, the test environment is explained and the experiment results are analyzed. Conclusions are drawn and possible future work is pointed out in Section 10.

2. Reference point-based NSGA-II

The resampling algorithms described in this paper are tested regarding how well they can support the Reference point-Guided NSGA-II algorithm (R-NSGA-II) (Deb et al., 2006), as an example for a guided Evolutionary Multi-objective Optimization (EMO) Algorithm. It is particularly suitable for evaluation in this paper, since it employs fitness functions which are used as resampling criteria in resampling algorithms. Therefore, the resampling algorithms can support R-NSGA-II particularly well.

R-NSGA-II is based on the Non-dominated Sorting Genetic Algorithm II (Deb et al., 2002) which is a widely-used and representative multi-objective evolutionary algorithm. NSGA-II sorts the solutions in population and offspring into different non-dominated fronts. Those selected comprise all solutions in those best fronts that fit completely into the next population. From the best front that only partially fits, a subset of solutions must be selected into the next population which have the largest distances to their neighbors. This selection mechanism is called Crowding Distance and guarantees that the result population will be diverse. The parental selection for offspring generation follows this fitness hierarchy. After selection is completed, offspring solutions are generated by tournament parental selection, crossover, and mutation. The offspring are evaluated and the selection step is performed again. The R-NSGA-II algorithm replaces the crowding distance operator by the distance to user-specified reference point(s). Solutions that are closer to a reference point receive a higher selection priority. The reference points are defined by the user in areas that are more interesting and where he or she wants more alternative solutions to be found in a limited area of the objective space. However, in order to not lose all diversity, a clustering mechanism with a minimum objective vector distance ϵ is used. Solutions that are too close to each other are considered with a lower priority. The R-NSGA-II selection step is depicted in Figure 1. The same fitness hierarchy, dominance first, then reference point distance and ϵ-clustering, is used for parental selection. All substeps of the selection process are documented in Figure 2.

The reference points can be created, adapted or deleted interactively during the optimization run, taking effect each time a new generation is started. In case all reference points have been removed, R-NSGA-II will continue the optimization as an instance of the regular NSGA-II algorithm. Since R-NSGA-II uses non-domination sorting it has a tendency to prioritize population diversity before convergence to the reference points. Therefore, extensions have been proposed which limit the influence of the Pareto-dominance and allow the algorithm to focus faster on the reference points (Siegmund et al., 2012b).

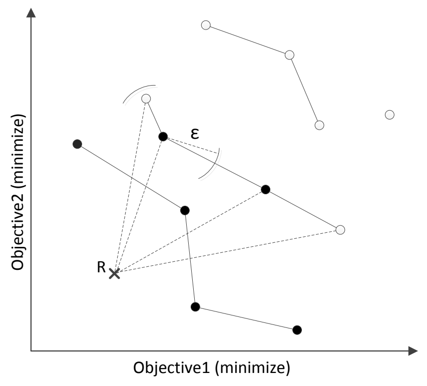
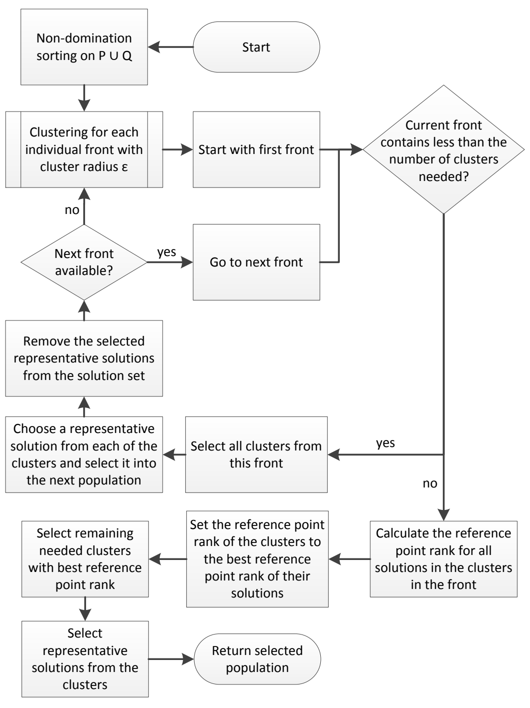


Figure 1: R-NSGA-II selection step. In this example, the set of population and offspring consists of 12 solutions grouped in 4 fronts. In order to select 6 solutions into the next population, 2 solutions have to be selected out of the second front by distance and ϵ-clustering.



Start
Non-domination sorting on P ∪ Q
Clustering for each individual front with cluster radius ε
Start with first front
Current front contains less than the number of clusters needed?
no
Next front available?
yes
Go to next front
Remove the selected representative solutions from the solution set
Choose a representative solution from each of the clusters and select it into the next population
Select all clusters from this front
yes
no
Select remaining needed clusters with best reference point rank
Set the reference point rank of the clusters to the best reference point rank of their solutions
Calculate the reference point rank for all solutions in the clusters in the front
Select representative solutions from the clusters
Return selected population

Figure 2: R-NSGA-II algorithm - Selection step flowchart showing all substeps of the selection process.

3. Dynamic Resampling

In this section, background information is given regarding Resampling as a noise handling method in Evolutionary Multi-objective Optimization and the preference-based multi-objective optimization algorithm R-NSGA-II (Deb et al., 2006), which are the basis for the proposed algorithms in this paper.

3.1. Noise Compensation by Resampling

To be able to assess the quality of a solution according to a stochastic evaluation function, statistical measures, such as sample mean and sample standard deviation, can be used. By executing the simulation model multiple times, a more accurate value of the solution quality can be obtained. This process is called Resampling. We denote the sample mean value of objective function F_i for solution s as follows: $\mu_n(F_i(s)) = \frac{1}{n}\sum_{j=1}^{n} F_i^j(s),\ \ i = 1 \dots H$, where $F_i^j(s)$ is the j-th sample of s, and the sample variance of objective function i: $\sigma_n^2(F_i(s)) = \frac{1}{n-1}\sum_{j=1}^{n}(F_i^j(s) - \mu_n(F_i(s)))^2$. The performance degradation evolutionary algorithms experience, caused by the stochastic evaluation functions, can be compensated partly through resampling. However, a price has to be paid for the knowledge gain through resampling and the resulting performance gain of the optimization. In a scenario with limited simulation time, the total number of evaluations is limited. If resampling is performed, the total number of solutions that can be evaluated is reduced. Thereby, the search space cannot be explored as much as when each solution is evaluated only once. A trade-off situation arises between sampling each solution several times and sampling different solutions in the search space.

In many cases, resampling is implemented as a Static Resampling scheme. This means that all solutions are sampled a fixed number of times. This static method uses an equal amount of the total, available sampling budget for all solutions. The need for resampling is, however, often not homogeneously distributed throughout the search space. Solutions with a smaller variance in their objective values will require fewer samples than solutions with a higher objective variance. Solutions closer to the Pareto-front or preferred areas in the objective space require more samples. In order to sample each solution the required number of times, a dynamic resampling technique is required, which is described in the following section.

3.2. Dynamic Resampling

Dynamic Resampling allocates a different sampling budget to each solution, based on the evaluation characteristics of the solution. The general goal of resampling a stochastic objective function is to reduce the sample standard deviation of the mean of an objective value $\sigma_n(\mu_n(F_i(s)))$ which increases the knowledge about the objective value. A required level of knowledge about the solutions can be specified. For each solution, a different number of samples is required, to reach the required knowledge level. Resampling can be performed dynamically in the way that each solution is allocated exactly the required number of samples, up to an upper bound. In comparison with Static Resampling, a part of the sampling budget can be saved and used to evaluate more solutions and run more generations of an evolutionary algorithm.

With only a limited number of samples available, the standard deviation of the mean can be estimated by the sample standard deviation of the mean, which is usually called standard error of the mean. It is calculated as in Equation 1.

$$\mathrm{se}_i^n(\mu_n(F_i(s))) = \frac{\sigma_n(F_i(s))}{\sqrt{n}}. \tag{1}$$

By increasing the number of samples n of $F_i(s)$, the standard deviation of the mean and its estimate $\mathrm{se}_n(\mu_n(F_i(s)))$ is reduced. For the standard error of the mean, however, this is not guaranteed, since the standard deviation estimate $\sigma_n(F_i(s))$ can be corrected to a higher value by drawing new samples of it. However, the probability of reducing the sample mean by sampling s increases asymptotically, as the number of samples is increased.

3.3. Sequential Dynamic Resampling

An intuitive dynamic resampling procedure would be to reduce the standard error until it drops below a certain user-defined threshold $\mathrm{se}_n(\mu_n(F(s))) < \mathrm{se}_{th}$. The required sampling budget for the reduction can be calculated as $n > \left(\frac{\sigma_n(F_i(s))}{\mathrm{se}_{th}}\right)^2$. However, since the sample mean changes as new samples are added, this one-shot sampling allocation might not be optimal. The number of fitness samples drawn might be too small to reach the error threshold, in case the sample mean has been shown to be larger than the initial estimate. On the other hand, a one-shot strategy might add too many samples, if the initial estimate of the sample mean is too big. Therefore, dynamic resampling is often done sequentially. Through this sequential approach, the number of required samples can be determined more accurately. For Sequential Dynamic Resampling, often the shorter term Sequential Sampling is used. Dynamic resampling algorithms can therefore be classified in one-shot sampling strategies and sequential sampling strategies.

Sequential sampling adds a fixed number of samples at a time. After an initial estimate of the sample mean and calculation of the required samples, the sufficiency of the knowledge about the solution is checked. If needed, another fixed number of samples is drawn and the number of required samples is recalculated. This is repeated as long as no additional sample needs to be added. The basic pattern of a sequential sampling algorithm is described in Algorithm 1. Through this sequential approach, the number of required samples can be determined more accurately than with a one-shot approach. It guarantees that the solution is sampled often enough, and can reduce the number of excess samples. Dynamic resampling algorithms can therefore be classified in one-shot sampling strategies and sequential sampling strategies.

Algorithm 1 Basic sequential sampling algorithm pattern.

```
1: Draw $b_{min}$ initial samples of the fitness of solution $s$, $F(s)$.
2: Calculate mean of the available fitness samples for each of
   the $H$ objectives: $\mu_n(F_i(s)) = \frac{1}{n}\sum_{j=1}^{n} F_i^j(s)$, $i = 1, \dots, H$.
3: Calculate objective sample standard deviation with available fitness samples:
   $\sigma_n(F_i(s)) = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}(F_i^j(s) - \mu_n(F_i(s)))^2}$, $i = 1, \dots, H$
4: Evaluate termination condition based on $\mu_n(F_i(s))$ and
   $\sigma_n(F_i(s))$, $i = 1, \dots, H$.
5: Stop if termination condition is satisfied or if the maximum
   number of samples is reached; Otherwise sample the fitness
   of $s$ another $k$ times and go to step 2.
```

4. Resampling Algorithms

In this chapter, several resampling algorithms are described, which are used in this study to support the R-NSGA-II algorithm for stochastic simulation optimization problems. We denote: Sampling budget for solution s: b_s, minimum and maximum number of samples for an individual solution: b_{min} and b_{max}, maximum overall number of simulation runs B, current overall number of simulation runs B_t, and acceleration parameter for the increase of the sampling budget: $a > 0$. Increasing a decreases the acceleration of the sampling budget, decreasing a increases the acceleration. The calculated normalized sampling need x_s is discretized as in Equation 2 which guarantees that b_{max} is already assigned for $x_s < 1$:

$$b_s = \min\{b_{max}, \lfloor x_s(b_{max} - b_{min} + 1)\rfloor + b_{min}\}. \tag{2}$$

4.1. Resampling algorithm classification

In this section, we present a classification of resampling algorithms based on Siegmund et al. (2013). Three classification categories have been described in the previous section. First, sampling algorithms that base their sampling allocation on solution properties such as mean and variance and sampling algorithms that do not use this information such as Static Resampling or resampling based on elapsed optimization time. Second, Dynamic Resampling algorithms that consider the objective value mean and variance and solutions that only consider the mean objective values. And third, Dynamic Resampling algorithms that calculate the sampling budget as one-shot and Dynamic Resampling algorithms that calculate the sampling budget sequentially. More classification categories can be defined. A list of resampling categories is presented in Table 1.

Table 1: Classification of resampling strategies.

Solution independent	Solution dependent
Mean-based	Variance-based
One-shot sampling	Sequential sampling
Single-objective	Multi-objective
Algorithm independent	Algorithm specific
Individual resampling	Comparative resampling
Single criterion	Hybrid

Another way to determine the number of samples is to aggregate the objective values into a single value. For example, the R-NSGA-II algorithm provides such a scalar aggregation value on which the sampling budget allocation can be based; the distance to the reference point. A Distance-based Resampling strategy could allocate few samples at the beginning of the optimization runtime, when the found solutions are far away from the reference point. Towards the end of the runtime, more objective value samples can be drawn, which helps to distinguish the many solutions that are close to the Pareto-front. In contrast, a truly multi-objective resampling strategy would be to assign the maximum number of samples to solutions with Pareto-rank 1 and the least number of samples to solutions with the maximum Pareto-rank.

If we use Distance-based Resampling on R-NSGA-II, we classify the use of this resampling technique as an algorithm specific resampling strategy. On the other hand, if Static Resampling is applied on an algorithm, we classify its use as an independent resampling strategy application, since it is not taking any characteristics of the optimization algorithm into account. This classification category is not always applicable in a clear way. If, for example, the Pareto-rank-based Resampling algorithm is applied on the R-NSGA-II algorithm, then only one part of the algorithm selection procedure is supported. The secondary selection criterion, the distance to a reference point, is not supported.

The sampling strategies introduced in the previous section consider only one single solution at a time, when making sampling allocation decisions. The Pareto-rank-based sampling strategy, however, compares solutions when allocating samples. The former strategies can therefore be classified as Individual Resampling and the Pareto-rank-based strategy as Comparative Resampling.

Some resampling algorithms base their sampling allocation on one resampling criterion only, such as elapsed optimization time, objective variance, or Pareto-rank. Hybrid dynamic resampling strategies combine multiple criteria to determine the sampling allocation.

For each solution in the generation, the Dynamic Resampling algorithm is applied and one or more multiple samples are assigned to it. This is called a resampling pass. Several resampling passes are executed, one after another, until no sample is added during the resampling pass. This is to make sure that each solution in a generation is treated equally, on the same preconditions. Otherwise, for Dynamic Resampling algorithms where the sampling allocation of one solution is dependent on the sampling result of other solutions, the desired sampling distribution cannot be achieved. Here, it is important that before the sampling allocation for a solution is calculated a second time, all other solutions have the chance to be evaluated by the resampling algorithm. In this way, we provide the possibility that more accurate information is available to recalculate the sampling allocation. Another case where we find that resampling in passes is advisable is when the sampling allocation is based on the elapsed optimization time, since the time changes with every solution sample that is evaluated. Solutions that are evaluated first in a generation might be sampled less often than others. This can be a disadvantage, in the case of hybrid time-based resampling algorithms. If the allocation is also based on other resampling criteria, this overall sampling allocation is biased inappropriately regarding the order of solution evaluation. Another reason for executing the resampling in passes is to facilitate the ex post analysis of the algorithms' inner workings.

4.2. *Final samples*

After the optimization is finished and the final non-dominated result set has been identified, a post-optimization resampling process is needed. For each solution that is presented to the decision maker, we want to be confident about the objective values. Therefore, we guarantee a high number of samples b_f for each solution ($b_f \geq b_{max}$) in the final population after the EA selection process. Thereby, we can also guarantee that the non-dominated solutions in the final population have been identified correctly. The total number of extra samples added to the last population is $B_F = (b_f - 1)|P|$, where $|P|$ is the population size.

For optimization problems with simple and quick function evaluations, another type of evaluation is possible. Instead of returning only the non-dominated solutions of the last population to the decision maker, a continuously updated archive of non-dominated solutions found during the whole optimization runtime is returned. The accurate evaluation of the solutions with b_f samples can be done in post-processing or in the background, without consuming and removing evaluations from the optimization process. The archive of non-dominated solutions can increase considerably, especially in high-dimensional objective spaces, and therefore it is not feasible to sample all the found non-dominated solutions b_f times with evaluations from the total optimization budget B. The evaluation can be done outside of this budget. Due to the negligible evaluation times for solutions, the time consumed for these extra evaluations, done as post-processing or in the background, is insignificant. This type of evaluation is suitable for algorithm benchmarking on test functions only.

4.3. *Comments on handling final results*

When Dynamic Resampling is performed, each solution is evaluated a different number of times. Consequently, the total number of samples done during one generation of an EA B_G is different and, in most cases, it is not possible to know this number of samples in advance. When the total number of solution evaluations B is limited, it is therefore necessary to estimate the number of samples needed for the next generation. Each time a new generation is to be started, the algorithm needs to make a decision whether the resampling on the new generation can be performed in the ordinary way with enough samples available.

In this paper, B_F samples are added to the last population, and our resampling algorithm framework makes a decision whether the remaining samples are sufficient according to Equation 3, where $|Q|$ is the number of offspring solutions. We do not evaluate more samples than b_f of each solution, in order to make the algorithm results more comparable. Any remaining samples are discarded.

$$B - B_t \geq (b_f - 1)|P| + b_{max}|Q|. \tag{3}$$

In case no final samples are added to the population the resampling algorithm framework makes a decision whether the remaining samples are sufficient according to Equation 4, where $\overline{B_G}$ is the average number of samples used for the last 3 generations. An exception is Static Resampling, where the coefficient of $\overline{B_G}$ is ignored.

$$1.1 \cdot \overline{B_G} \geq B - B_t. \tag{4}$$

If Equation 4 is satisfied, the next generation is executed as usual. If the available number of samples, contrary to expectations, is not enough, the sampling is stopped. Due to the sequential sampling procedure, almost all solutions should have been sampled according to the original sampling allocation.

If Equation 4 is not satisfied, no new generation is evaluated. Instead, the available samples are distributed among the current population, one by one, according to the fitness hierarchy. In case of R-NSGA-II, this means that the samples are distributed by the order of reference point distance (R-NSGA-II reference point rank).

5. Generally applicable resampling algorithms

This section contains resampling strategies that can be used in all multi-objective optimization problems, regardless of whether preference information is given by a decision maker or not. They can support the goal of finding the whole Pareto-front, as well as the goal of exploring a limited, preferred area in the objective space.

5.1. Static Resampling

Static Resampling assigns the same constant number of samples to all solutions involved in an optimization run ($b_s = b_{min} = b_{max}$). This is popular among optimization practitioners, since it requires a relatively small implementation effort. The disadvantage is that accurate knowledge of the objective vectors is not needed for all solutions; not all solutions have objective values with high variability and, at the beginning of an optimization run, the benefit of accurate values usually does not justify their cost. If, for example, $B = 5{,}000$ and $b_s = 5$, there will only be 1,000 solutions available, with which to explore the search space. Whether the increased accuracy can make up for the lost exploratory power depends on the objective noise level. If the noise is high enough, there might be a $b_s > 1$, where Static Resampling can lead to an improvement of the optimization result, compared to $b_s = 1$. However, compared to more advanced resampling algorithms, Static Resampling is inferior, since many samples are wasted on less important solutions. An exception to this rule is the degenerated form of Static Resampling with $b_s = 1$ for low noise levels. It achieves competitive results in this case (Siegmund et al., 2015), since the optimization has more solutions available for exploring the search space, which is most important in a low-noise scenario.

5.2. Time-based Dynamic Resampling

Time-based Dynamic Resampling (Siegmund et al., 2013) is a generally applicable Dynamic Resampling algorithm which can be used on any optimization algorithm. It assumes that the need for accurate knowledge of objective values increases towards the end of the optimization run. This hypothesis was proven in Aizawa & Wah (1993, 1994) for single-objective Evolutionary Algorithms. Time-based Dynamic Resampling allocates a small sampling budget at the beginning and a high sampling budget towards the end of the optimization. The strategy of this resampling algorithm is to support the algorithm, when the solutions in the population are close to the Pareto-front, and to save sampling budget at the beginning of the optimization, when the solutions are still far away from the Pareto-front. The normalized time-based resampling need x_s^T is calculated as in Equation 5.

$$x_s^T = \left(\frac{B_t}{B}\right)^a. \tag{5}$$

Time-based Resampling is an optimization algorithm independent and solution independent Dynamic Resampling algorithm. It makes only one sampling decision for each solution (one-shot allocation), according to the elapsed time. However, if a solution survives into the next generation of an evolutionary algorithm, the decision is made again.

The total number of samples available for optimization is limited by the final samples B_F added to the last population. For time-based resampling, this means that the elapsed time needs to be adjusted so that 100% of the time is already reached when $B - B_F$ samples have been used. The modified allocation function is shown in Equation 6. This function is used in all the other resampling algorithms that use the elapsed optimization time as one of their resampling critera.

$$x_s^T = \min\left\{1, \left(\frac{B_t}{B - B_F}\right)\right\}^a. \tag{6}$$

5.3. Rank-based Dynamic Resampling

Rank-based Dynamic Resampling (Siegmund et al., 2013) is a dynamic resampling algorithm which can be used on any multi-objective optimization algorithm. It measures the level of non-dominance of a solution and assigns more samples to solutions with a lower Pareto-rank and fewer samples to solutions in the last fronts, in order to save evaluations. The normalized rank-based resampling need x_s^R is calculated as in Equation 7. We denote: S = solution set of current population and offspring, R_s = Pareto-rank of solution s in S, $R_{max} = \max_{s \in S} R_s$.

$$x_s^R = 1 - \left(\frac{R_s - 1}{R_{max} - 1}\right)^a. \tag{7}$$

Rank-based resampling shows good results on (R-)NSGA-II, which is expected, since the dominance relation is its main fitness function. In contrast to the previously described algorithms, Rank-based DR is a solution-dependent and mean-based sampling allocation. Since the dominance relations in the population of a multi-objective optimization algorithm changes after each added sample, Rank-based DR is performed sequentially. It is based on multiple objectives, but otherwise independent of the optimization algorithm. Also, it is a comparative resampling technique based on the single criterion of Pareto-rank. In Many-objective Optimization, Rank-based Dynamic Resampling is not effective, since most solutions in a well-converged solution set are non-dominated and are assigned the maximum number of samples.

5.3.1. *RankMaxN-based DR*

We propose a modification of Rank-based Resampling that allocates additional samples only to the first n fronts. This allows us to concentrate the additional samples on the first fronts, where they can be more beneficial. The allocation function of MaxN-Rank-based Resampling is given in Equation 8.

$$x_s^{Rn} = 1 - \left(\frac{\min\{n, R_s\} - 1}{\min\{n, R_{max}\} - 1}\right)^a. \tag{8}$$

5.3.2. *Rank-Time-based DR*

We propose a Hybrid Dynamic Resampling algorithm: Rank-based DR can be combined with Time-based DR to avoid allocating samples at the beginning of the optimization, where the optimization algorithm only has slight gains from knowing the accurate objective values. Similar to a logical conjunction, x_s^R and x_s^T can be combined to form the Rank-Time-based Resampling allocation x_s^{RT} for solution s, as in Equation 9. Another allocation would be the product of both allocation values: $x_s^{RT} = x_s^T x_s^R$.

$$x_s^{RT} = \min\{x_s^T, x_s^R\}. \tag{9}$$

6. Preference-based resampling algorithms

This section describes two resampling algorithms that use the preference information given by a decision maker in the form of a reference point R in the objective space for the R-NSGA-II algorithm (Deb et al., 2006).

6.1. *Progress-based Dynamic Resampling*

Progress-based Dynamic Resampling allocates samples to solutions depending on the progress of the population towards a reference point r. Its premise is that if the EMO population progress towards a reference point slows down, the population members are concentrated in a small area. In this situation, the algorithm will benefit from accurate knowledge about the objective vectors. The progress is defined as the average distance from the population members to r. Since the progress in Evolutionary Multi-objective Optimization can fluctuate, the average progress $\overline{P}$ from the last n populations is used. Progress-based Resampling is a solution independent, mean-based, sequential sampling allocation. It uses the average scalar distance value and is algorithm dependent, in the sense that a reference point is required. All solutions in the population are treated equally, based on one resampling criterion, the average population progress. We denote: P_{max} = the maximum progress threshold. For $\overline{P} > P_{max}$, b_{min} is allocated to all solutions in the population. If a progress measurement is negative, the absolute value of the progress will be used, multiplied by a penalty factor. This method also works for the case of a feasible reference point, where a population moving away from r is a desirable behavior. When convergence is reached, the absolute progress value becomes smaller and smaller, leading to higher sampling allocations. The normalized progress-based resampling need x_s^P is calculated as in Equation 10.

$$x_s^P = 1 - \left(\frac{\min\{\overline{P}, P_{max}\}}{P_{max}}\right)^a. \tag{10}$$

The disadvantage of this strategy becomes apparent in a situation of premature convergence. If a population-based optimization algorithm fastens in a local optimum, it has been shown that more objective uncertainty, and not less, can be helpful to escape the local optimum (Jin & Branke, 2005). Therefore, progress-based resampling without using the distance information to the reference point is often of little use. As soon as the optimization fastens in a local optimum far from R, too many samples are wasted. In the following section, we describe a hybrid progress-based resampling algorithm which solves this problem by considering the reference point distance and elapsed optimization runtime: Distance-based Dynamic Resampling, DDR.

Another disadvantage of Progress-based DR is that all solutions in the population are assigned the same budget. This means that solutions, population or offspring, which are dominated by many solutions or are distant to R and thereby less relevant, will be assigned an unnecessarily high number of samples. These solutions might be discarded in the next selection step of the evolutionary algorithm, which reduces the benefit of the assigned samples even more. This can be avoided by combining the progress measure with the distance to R as it is done in DDR in the following section. Another way to solve this problem is to use a second resampling criterion that indicates whether solutions have a high chance of being selected, better than with the distance measure. For R-NSGA-II, this is the Pareto-rank. A resampling algorithm combining progress and Pareto-rank is presented in Section 6.3. In this algorithm, Distance-Rank-based Dynamic Resampling (called DR2), the distance to r is used to control b_{max} for the whole population instead.

6.2. *Distance-based Dynamic Resampling (DDR)*

In this section, we describe a resampling strategy that is specifically adapted to the R-NSGA-II algorithm and uses the extra information that is available through the R-NSGA-II reference points. This hybrid Dynamic Resampling strategy Distance-based Dynamic Resampling DDR was proposed in Siegmund et al. (2013). It assigns more samples to solutions that are close to the reference points. The intention of allocating samples in this way is that solutions which are close to a reference point are likely to be close to the preferred area on the Pareto-front. This part of the population has converged and it is important to be able to reliably identify which solutions dominate other solutions.

According to the classification in Table 1, Distance-based Dynamic Resampling considers only the mean objective values and ignores the objective variances, performs sequential sampling, uses the reference point distance which is an aggregated form of the objective values, and is an algorithm specific resampling method that allocates the sampling budget for each solution individually. It is also a hybrid resampling algorithm, as we describe in the following. We denote: r_s is the reference point that is closest to solution s. In this study, we use an Achievement Scalarization Function (Siegmund et al., 2012a) as distance metric. $\delta_{ASF}(F(s), R_s)$ is the absolute reference point distance. For the sampling allocation, we use the normalized reference point distance $d_s = \min\{1, \frac{\delta_{ASF}(F(s),R_s)}{D}\}$, where D is the maximum reference point distance of the initial population of R-NSGA-II. An intuitive way to assign resamplings based on reference point distance is Equation 11.

$$x_s^{DDR} = 1 - d_s. \tag{11}$$

Reference points are usually not chosen on the Pareto-front. Therefore, a transformation function needs to be applied to Equation 11. In this paper, we describe the case of infeasible reference points. The definition of an infeasible reference point is that it is not on the Pareto-front and no solution can be found that dominates it (Miettinen, 1998). Handling infeasible reference points requires an approach that adapts the transformation function dynamically during the optimization runtime, since it is not known from the beginning whether the reference point is infeasible. We assume that in most cases the decision maker will define a reference point that is optimistic and therefore no solution can be found that attains this level of expectation. The case of pessimistic reference points, where solutions are found that dominate the reference point, is not considered in this article. We have proposed transformation functions for feasible reference points in Siegmund et al. (2013), and they are not our priority in this article. We refer to the feature of R-NSGA-II that allows the interactive adaption of a pessimistic reference point, during the optimization process that has been identified as a feasible one, and moves it forward to turn it into an infeasible point. Distance-based Dynamic Resampling for feasible reference points is discussed in future work.

6.2.1. Adaptive strategy for infeasible reference points

In this section, we propose a method of how to adapt the transformation function in the case of an infeasible reference point. We use this transformation function as long as no solution that dominates the reference point is found. This means that feasible reference points on the Pareto-front are handled in the same way as infeasible reference points. In the case of an infeasible reference point, the reference point distance of the sought-after, focused Pareto set cannot be reduced below a certain distance $\underline{\delta}$ by optimization. This scenario is displayed in Figure 3. Therefore, if this lower bound $\underline{\delta}$ of the reference point distance is large enough and a linear, or delayed, distance-based allocation scheme is used, the maximum sampling budget will never be reached during the optimization runtime. Too few samples will be assigned to the last populations that include the best found solutions. Therefore, in order to create a good distance-based sampling strategy for infeasible reference points, the budget allocation has to be accelerated. Then, it is not necessary to come close to the reference point to gain the highest number of samples. The full number of samples will already be allocated to solutions at a certain distance to the reference point. Since the lower bound of the reference point distance is not known, the sampling allocation has to be accelerated iteratively. The challenge is to adapt the transformation automatically, so that at the end of the optimization run the samples are distributed among all found solutions, as if the reference point was feasible and on the Pareto-front.

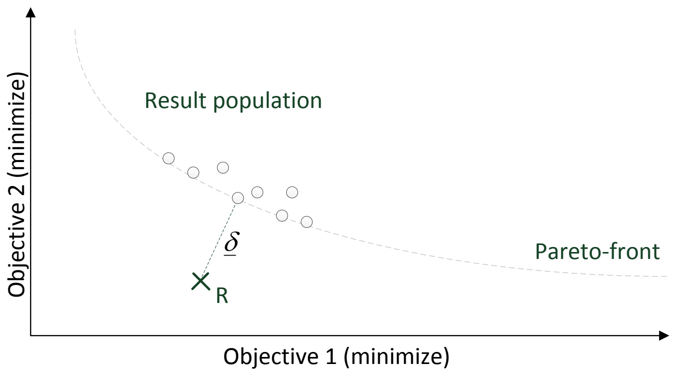


Figure 3: Bi-objective min/min-problem scenario with infeasible reference point R. The distance of the result population to R is limited by a lower bound $\underline{\delta}$.

To achieve this, the distance criterion is combined with two additional criteria, thus making DDR a hybrid Dynamic Resampling algorithm:

1. The progress of the population during the optimization runtime and 2., the elapsed time since the start of the optimization. In contrast to the algorithm independent strategies, this algorithm has access to distance information to reference points and can measure progress. The optimization progress will slow down as soon as the population approaches the Pareto-front, which is at some distance to the reference point. In this way, it can be detected that the algorithm is converging towards the preferred area and that more samples per solution are needed. However, the reference point distance and the optimization progress alone would not be sufficient, since the algorithm might temporarily slow down in local optima at an early stage and show only little progress. This can, for example, occur for the ZDT4 function (Zitzler et al., 2000) which features many local Pareto-fronts. In that case, the goal is to escape the local optimum, which does not require more exact knowledge about the solutions and no increased number of samples. That is why, in the case of infeasible reference points, the reference point distance, progress, and elapsed time together are taken into account for the sampling allocation.

The optimization progress is measured by the average Reference Point Distance of the population, on average for the last n populations. This metric was proposed in Siegmund et al. (2013). The average improvement in reference point distance of the last n populations is used as a progress indicator. The transformation function is designed as an ascending polynomial function. Initially, it will assign fewer than the maximum allowed samples to the solution that is closest to the reference point, in order to be prepared for the case of a feasible reference point. If the reference point is feasible, the solutions that dominate the reference point should get the highest number of samples instead. As long as we can assume infeasibility and if the optimization progress per generation drops, the slope of the transformation function will be increased in several steps. If the average generation progress $\overline{P}$ is less than 10% per generation, the transformation function will assign the maximum allowed number of samples only to the (hypothetical) solutions that are closest to the reference point. If the average progress is below 5%, the transformation function will be adapted in a way that guarantees at least 10% of the population full samples. Accordingly, a progress < 2.5% corresponds to 20% of the population with full samples and a progress < 1% to 40%.

As transformation function, $x_s^2 = (1 - d_s)^a$ is used, where a is a polynomial acceleration parameter. The calculation is done by taking a percentage of the population and measuring the maximum relative distance m of this subset S to the reference point, $m = max_{s \in S} d_s$. The transformation function is adapted through a factor c which makes sure that the maximum number of samples is already reached at distance m to the reference point, cf. Equation 12.

$$c(1 - m)^a := 1 \Rightarrow c = \frac{1}{(1 - m)^a}. \quad (12)$$

However, as mentioned above, if the average population progress is above 10%, then the sampling allocation will be reduced by using $c = 1 - m$, where m is the maximum distance of the best 10% of the population. If it will be detected that the reference point is feasible, this allows a smooth transition to the feasible case which uses a similar allocation function. The transformation functions are defined as in Equation 13.

$$\min\{1, c(1 - d_s)^a\}. \quad (13)$$

This allocation scheme is combined with a sample allocation based on the elapsed optimization runtime. The elapsed time is measured by the number of executed solution evaluations in relation to the maximum number of evaluations, as it is done by Time-based Resampling in Section 5.2. If B_f final samples are added at the end of the optimization run, the time measurement is adjusted as in Equation 6. The transformation function is adapted by reducing c to delay the incline of the sampling allocation. Before 50% of the time has passed ($x^T < 50\%$), the distance m will be reduced to $m_0 = 0$, i.e. $c = 1$. Until 65%, m will be reduced to $m_1 = 1/3m$ and, until 80%, it will be reduced to $m_2 = 2/3m$. The transformation functions for an example case with $\underline{\delta} = 0.37$ are shown in Figure 4.

Combining the intuitive distance-based allocation in Equation 11 with the adaptive transformation functions in Equation 13 gives us the resampling need x_s^{DDR} as in Equation 14.

$$x_s^{DDR} = \min\left\{1, \left(\frac{1 - d_s}{1 - \underline{\delta}}\right)^a\right\}. \quad (14)$$

As soon as a solution that dominates the reference point is found, the sampling allocation will switch to the allocation function for the case of feasible reference points, starting with the next generation of R-NSGA-II. We have proposed trans-

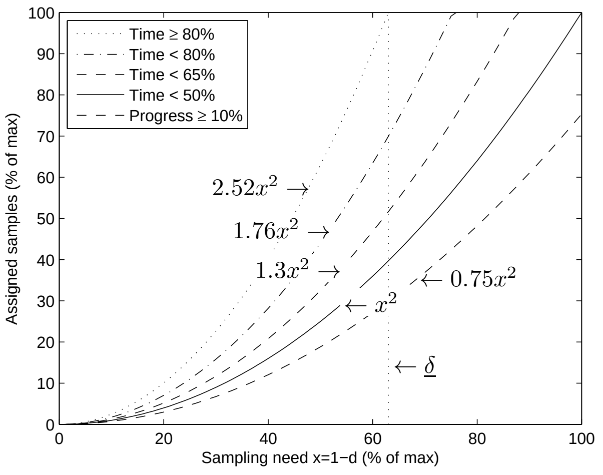


Figure 4: Transformation functions for an infeasible reference point. The figure shows values for coefficients c for a scenario with $\underline{\delta} = 0.37$. The coefficient for $x^T \geq 80\%$ is therefore $c = 1/(0.63)^2 \approx 2.52$. For $x^T \in [65\%, 80\%)$ and $m_2 = 2/3m$ it will be reduced to $c = 1/(0.75\overline{3})^2 \approx 1.76$. The sampling need x in percent based on the relative reference point distance d is translated into a relative number of samples that are allocated to a solution.

formation functions for feasible reference points in Siegmund et al. (2013) that can provide a seamless sampling allocation and smooth transition from the infeasible to the feasible case. Future work will cover a detailed evaluation of Distance-based Dynamic Resampling for feasible reference points.

6.3. Distance-Rank Dynamic Resampling (DR2)

As an attempt to combine multiple different resampling criteria in a resampling algorithm, in Siegmund et al. (2015) we proposed to combine the Rank-based Dynamic Resampling and the Distance-based Dynamic Resampling (DDR) strategies, described previously in Section 6. We call it Distance-Rank Dynamic Resampling (DR2). It uses four different factors to determine the resampling allocation for individual solutions: Pareto-rank, time, reference point distance, and progress. Whereas the elapsed optimization time can be combined with any other resampling criterion with little effort, as seen with Rank-Time-based Dynamic Resampling mentioned above, the Pareto-rank and the reference point distance are two truly different resampling criteria. Therefore, we emphasize the term Hybrid for the DR2 algorithm.

Equation 15 describes the sampling allocation of DR2. Similar to Rank-Time-based Resampling, the minimum of both the normalized sampling need of Distance-based Resampling and Rank-based Resampling is used to create a combined sampling allocation. However, the Distance-based sampling need is not calculated individually for each solution. Instead, DR2 identifies the solution s_m closest to r, $d_m = \min_k\{d_k\}$ and the normalized sampling need for x_m^{DDR} is used in the formula as fixed value for all solutions in the current generation. This particular way of combining rank and distance information was chosen for DR2, since it has shown the best optimization results in different situations.

$$x_s^{DR2} = \min\left\{x_m^{DDR}, x_s^R\right\}. \qquad (15)$$

7. Stochastic Production Line Models

The benchmark optimization problem used for evaluation is a stochastic simulation model consisting of 6 machines (Cycle time 1 min, σ = 1.5 min, CV = 1.5, lognormal distribution) and 5 buffers with sizes $\in [1, 50]$. The source distribution is lognormal with μ = 1 min and σ = 1.5 min, if not stated otherwise. The basic structure is depicted in Figure 5. The simulated warm-up time for the production line model is 3 days and the total simulated time of operation is 10 days. The conflicting objectives are to maximize the main production output, measured as Throughput (TH) (parts per hour), and to minimize the sum of the buffer sizes, TNB = Total number of buffers. This is a generalization of the lean buffering problem (Enginarlar et al., 2005) (finding the minimal number of buffers required to obtain a certain level of system throughput). In order to consider the maintenance aspect, the machines are simulated with an Availability of 90% and a MTTR of 5 min, leading to a MTBF of 45 min.

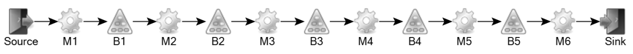


Figure 5: A simplistic production line configuration.

There are two variants of the production problem: a model where one of the machines has a highly variable cycle time and a model with an uncertain source node. We present the two variants in the following sections.

7.1. Noisy machine

In this stochastic production line model (abb. PL-NM), the cycle time of machine M4 has a higher standard deviation (σ = 10 min, CV = 10) than the other machines, causing a high standard deviation of the average throughput objective measured during the whole simulation time. Thus, Dynamic Resampling algorithms are required to compensate for the noisy Throughput values.

Automated machine processes are essentially deterministic, so that an automated production line has, in principle, very low variability. But in practice, a single, stochastic, manual workstation is enough to add very high variability to an automated production line (Papadopoulos et al., 2009). The high variability may be caused by a manual operator who is needed to start an automated process, e.g., after tool changes. Or more commonly, the variability may be caused by a manual quality inspection workstation in which the times for visually checking for any defects in the work-pieces vary significantly. For the model shown in Figure 5, if the processing times are all constant (= 1 min), the CV of the main production output (TH) is 0. Only machine M4 is changed to have a lognormal distribution. This is more suitable in order to model human work time distribution than a normal distribution, according to Dudley (1963). Mean and standard deviation of the processing time are set as 1 min and 25 min respectively (i.e. CV=25). Thereby, the CV of TH becomes 0.2089, i.e., around 20% noise, which is a sufficiently high noise level to test the Dynamic Resampling algorithms. We call this model PL-NM-20% in the following. Three other configurations of the problem with different noise levels are used for experimentation in this paper: PL-NM-5% with σ = 7 min for M4 which leads to a TH-CV of 0.0580. PL-NM-10% with σ = 15 min for M4 whith TH-CV of 0.1037. And the noisiest model PL-NM-30% with σ = 35 min for M4 and a TH-CV of 0.2958.

7.2. Noisy source

In this stochastic production line model (abb. PL-NS), the input is unreliable. This means that the parts entering the system at the source do no follow a fixed input distribution, which makes it hard to design an optimized production line. Instead, a robust design is needed which performs as well with few parts entering the system as for the case of parts entering the systems frequently. The source follows a lognormal distribution with σ = 1.5 min, but the mean can vary between 15 parts per hour and 120 parts per hour, on a uniform distribution (This corresponds to a part entering the system between every 240 seconds and every 30 seconds). When a solution is simulated multiple times, this leads to an average TH-CV of 0.2037, which is approximately 20% noise. We call this model PL-NS-20%. We also perform experiments with three other configurations of different noise levels: PL-NS-5%, where the mean varies between 30 and 120 parts per hour and the TH-CV becomes 0.0503. PL-NS-10%, where the mean varies between 20 and 120 parts per hour and the TH-CV becomes 0.0977. And the noisiest model PL-NS-30%, where the mean varies between 12 and 120 parts per hour. This means that parts may enter the system as slowly as every 300 seconds and the TH-CV becomes 0.3103.

For the PL-NS models, the optimization has to be conservative and find robust solutions that anticipate the highly variable input source. Therefore, a type of multi-objective Robustness optimization approach is applied (Deb & Gupta, 2006), where Dynamic Resampling is used to estimate the mean according to different accuracy requirements. Objective variance, or a variance constraint, is not used explicitly, only the mean fitness is optimized. However, since the sampling distribution of the mean is not limited to a local neighborhood, but only to the global interval [30s, 240s], high variance values influence the mean value and are considered implicitly in this way.

8. Performance metrics

In this section, we propose several multi-objective performance metrics which allow us the measurement of optimization performance indicators, such as convergence of a population towards a reference point or diversity of the population. Two of the metrics can be used to measure both performance indicators, convergence and diversity, at the same time in a single aggregated value.

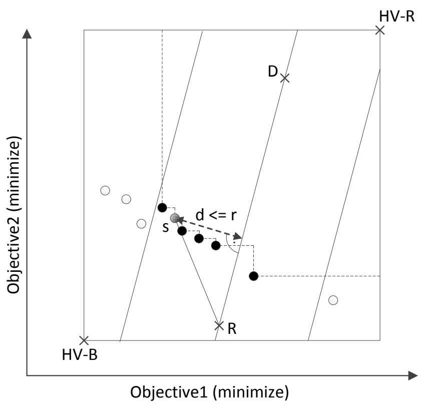


Figure 6: The Focused Hypervolume (F-HV) performance measure with cylinder filter of radius r. The objective vectors marked in black are considered for performance measurement. A solution s is only considered for hypervolume measurement, if it is contained in the cylinder, which means $d \leq r$. Some of solutions marked in black are dominated, but they are non-dominated after the cylinder filter operation.

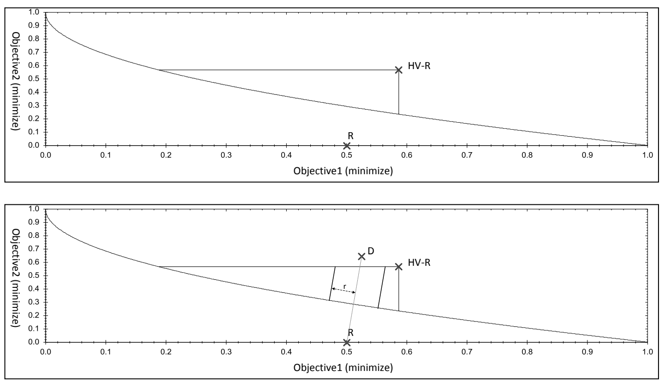


Figure 7: Hypervolume (HV) vs. Focused Hypervolume (F-HV) performance measures. In this example, the Pareto-front is flat in relation to the whole objective space [0,1]x[0,10]. If the HV-R point is put close to R the non-dominated points to the left side of R will have a higher influence on the HV value. Therefore, F-HV applies a cylinder filter with radius $r = c\frac{|P|}{2}\epsilon$, before the non-domination sorting is done. This ensures a more equal influence of the points on both sides of R on the Pareto-front.

8.1. Focused Hypervolume (F-HV)

To measure and compare the results of the different resampling algorithms together with R-NSGA-II, we propose the Focused Hypervolume performance metric for ϵ-dominance based EMOs (F-HV) (Siegmund et al., 2015). The F-HV allows the measurement of the convergence and diversity of a population limited to a preferred area in the objective space. The limits are defined on the basis of the intended diversity and the population size of the optimization algorithm. This allows us to measure the degree to which the optimization algorithm can achieve the intended diversity. For the R-NSGA-II algorithm, the intended diversity is controlled by the user parameter epsilon. F-HV is based on the Hypervolume metric (HV) (Zitzler & Thiele, 1998). In F-HV the population is filtered before it is assessed with the regular HV metric. The filter is a cylindrical subspace of the objective space retaining only solutions close to the reference point R dominating the HV-reference point $HV\text{-}R$. A graphical example is shown in Figure 6. The cylinder axis is defined by R and a second point D determines the direction, approximately orthogonal, to the potentially only partially known true Pareto-front. Often D is defined as $D := HV\text{-}R$.

8.1.1. Motivation

The F-HV metric allows us to limit a relevant area close to the reference point which aims to give equal influence to solutions in all directions on the Pareto-front. In the example shown in Figure 7, the solutions on the left side of R would have more influence on the HV value, if the F-HV cylinder filter was not in place. The cylinder radius is a parameter which allows us to customize the metric values, so that the highest values will be achieved, if the algorithm population is able to exactly maintain the intended diversity, which is controlled by ϵ in R-NSGA-II. As we are only measuring the algorithm performance on bi-objective optimization problems in this article, we set the cylinder radius as the product of the algorithm population size and ϵ, as described further below.

Another important reason for using the F-HV metric is to be able to measure HV metric values early in the optimization process. Due to the F-HV cylinder, $HV\text{-}R$ can be chosen further away from R and the larger HV area can capture solutions further away from R. A larger HV area however is not able to measure the performance of the algorithm to converge towards a preferred area. The cylinder filter ensures that the metric values remain relevant for preference-based optimization.

8.1.2. Detailed description

A more detailed description of the F-HV definition follows. For R-NSGA-II and a bi-objective problem, solutions within the distance $r = \frac{|P|}{2}\epsilon$ from the cylinder axis, with $|P|$ being the population size and ϵ the R-NSGA-II clustering parameter, are passed on to the standard HV, the rest is discarded. In this way, there is enough space within the cylinder for $|P|$ non-dominated solutions, each with the distance ϵ to its neighbors. The distance d of solution s to the cylinder axis is calculated by projecting $\vec{s}-\vec{R}$ onto the cylinder axis. The projection vector $\vec{p}$ is obtained as $\vec{p} = (\vec{s} - \vec{R}) \cdot (\vec{D} - \vec{R})/|(\vec{D} - \vec{R})|$, where the operator $\cdot$ denotes the scalar product. The distance of s towards the cylinder axis is then obtained as $d = |\vec{p} - (\vec{s} - \vec{R})|$ and has to be smaller than the cylinder radius, $d \leq r$, in order for solution s to be considered for hypervolume measurement.

The cylinder filter must be applied before the non-domination sorting is performed. Otherwise, dominated solutions are filtered out during the non-domination sorting, which would be non-dominated after the application of the cylinder filter. In our case of stochastic simulation with limited budget, we allow the cylinder radius to be larger: $r = c\frac{|P|}{2}\epsilon$, $c \geq 1$, since it is not possible for the optimization to converge with enough solutions into the cylinder within the available optimization time. F-HV requires four points to be specified: A reference point R, a direction point D, and a hypervolume reference point and base point $HV\text{-}R$ and $HV\text{-}B$ (Figure 6).

8.1.3. *Similar performance metrics*

We have proposed a performance metric similar to F-HV which is called R-Metric or R-HV (Deb et al., 2014). It is a performance metric that is able to compare sets of non-dominated solutions with different distances to R in the same hypervolume setting. The different non-dominated sets are made comparable by projecting the solutions on an axis defined by R and $HV\text{-}R$. Solution sets with a larger reference point distance will keep a larger distance on the axis. In other words, the order of reference point distance between solution sets is invariant to the projection operation. Also, an ϵ-filter similar to that of the F-HV metric is applied. A $|P|\epsilon$-wide box is applied as a filter, before the projection operation.

However, due to the limited optimization time in our experiments, the population often never fully converges towards the Pareto-front, or R. During the whole optimization runtime, the population has a wide spread and is moving. Therefore, the representative point used by the R-HV filter often changes, which leads to highly fluctuating metric values as the optimization progresses. Due to the noise and Dynamic Resampling, the objective values of already found solutions change as additional samples are added, which contribute to the fluctuation of metric values. As a result, we recommend the use of the R-HV metric for non-noisy scenarios on almost converged populations.

Another performance metric for preference-based EMO algorithms using hypervolume was created by Brockhoff et al. (2013). It uses a weighted hypervolume indicator. The hypervolume contribution of solutions close to R is emphasized by defining a density function with the help of a weight parameter.

8.2. *Focused IGD (F-IGD)*

We propose another performance measure for reference-point based multi-objective evolutionary algorithms. The Focused IGD performance measure F-IGD is based on the Inverse Generational Distance (Coello Coello & Reyes Sierra, 2004) and measures convergence and diversity of the population. It requires the true Pareto-front of the optimization problem to be known. Therefore, in this paper it is used to assess the performance on benchmark functions where the definition of the Pareto-front can be derived from the problem specification. Instead of using a reference set of objective vectors covering the whole Pareto-front, the Focused IGD metric uses a reference Pareto-set close to the reference point. This set is generated with the help of the Focused Hypervolume cylinder. All objective vectors of the non-dominated reference set which are contained in the cylinder are members of the focused reference set. A diagram showing a focused reference set of objective vectors is shown in Figure 8. An additional F-IGD parameter is the number of objective vectors in the reference set.

A similar performance metric which uses reference points to create a focused reference set of objective vectors was created by Mohammadi et al. (2013).

8.3. *Focused Reference Point Convergence (F-RC)*

The Reference Point Convergence RC is a performance metric that measures the convergence of a population towards a reference point in the objective space. It was proposed in Siegmund et al. (2013). In order to avoid outliers, only the $\alpha\%$ closest solutions to R are considered, which is set P_α, as in Equation 16.

$$RC\,(P) = \sum_{s \in P_\alpha} \delta_{ASF}\,(F(s), r)\,/\,|P_\alpha|. \tag{16}$$

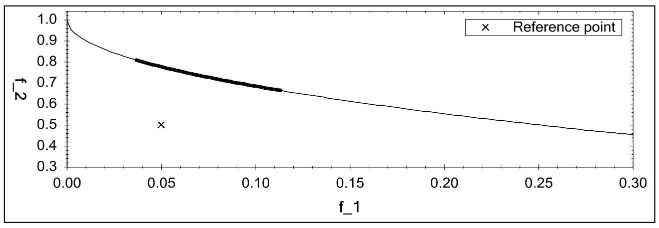


Figure 8: A Focused Inverse Generational Distance F-IGD reference Pareto-front of the ZDT1 and ZDT4 benchmark functions used in this paper. The focused reference Pareto-set marked in bold was generated by the application of the F-HV cylinder filter on the complete reference Pareto-set. In this example the cylinder axis was defined by R = (0.05, 0.5) and HV-R = (0.1, 1.5) with a cylinder radius of r = 0.05.

The Focused Reference Point Convergence F-RC makes use of the F-HV cylinder to create a focused solution set for performance assessment. Instead of using P_α which can be spread widely in the objective space, the focused population P_F is created, which is contained within the F-HV cylinder and locally limited in the objective space. In order to make the measurement insensitive to remaining outliers, the median distance of all points in P_F to the reference point r is measured and followed during the optimization runtime (Eq. 17).

$$F\text{-}RC\,(P) = \widetilde{\delta_{ASF}}\,(F(s), r), \quad s \in P_F. \tag{17}$$

Unlike F-HV and F-IGD, F-RC measures convergence towards r but is not able to measure the diversity of the focused population towards r. Therefore, another metric is required which only measures diversity of the focused population. It is described in the following section.

8.4. *Focused Diversity (F-Div)*

In order to measure the diversity of a focused population towards a reference point r, we propose a metric based on the NSGA-II Crowding Distance by Deb et al. (2002). The Crowding Distance is defined for mutually non-dominating sets of solutions. It is used by the NSGA-II algorithm to stimulate diversity of the population. During the selection phase, partial selection has to be done on one of the non-dominated fronts, where the solutions with the greatest distance to others are favored. Our goal is to continuously measure the diversity of a population-based multi-objective optimization algorithm during the optimization runtime. For this purpose, the diversity of not only the first front, but also of all other fronts in the population, has to be measured. This is because the diversity of the fronts $2 \ldots n$ influences the future development of the diversity of the first front. Another reason for measuring the diversity of all fronts in the population are the noisy objective vectors. Some solutions that appear to be part of the first front of the population would be identified as dominated solutions, if the true objective values were known.

Our goal is to calculate a diversity measure allowing us to compare the diversity of populations, therefore called Population Diversity PD. In order to make the value independent of the population size, the diversity measure δ based on the Crowding Distance is calculated for each solution within the population; all values are summed up and divided by the population size as in Equation 18.

$$PD\,(P) = \sum_{s \in P} \delta\,(s)\,/\,|P|. \tag{18}$$

In the context of preference-based optimization, it is crucial to only measure the diversity of the population in the preferred region of the objective space. For this purpose, we use the cylinder filter of the Focused Hypervolume, described above, to obtain the filtered population P_F and call the performance metric Focused Diversity F-Div (Eq. 19).

$$F - Div\,(P) = PD\,(P_F)\,. \tag{19}$$

The Crowding Distance measures the sum of the objective distances between neighboring solutions in the same non-dominated front. Extremal solutions without two neighboring solutions are assigned value infinity. In order to make the values comparable, for extremal solutions, PD assigns the objective distances towards the single neighbor N instead. If there is a front consisting of only one solution, then this solution is assigned value zero. The formal definition of $PD\,(s_k)$ is given in Equation 20, where N_j are neighboring solutions and we write $F_i(s_k) := \mu_n(F_i(s_k))$ to simplify the notation.

$$\delta\,(s_k) = \sum_{i=1}^{H} \begin{cases} |F_i(s_k) - F_i(s_N)| & \text{if } F_i(s_k) \text{ extremal} \\ |F_i(s_{N1}) - F_i(s_{N2})| & \text{otherwise.} \end{cases} \tag{20}$$

9. Numerical Experiments

In this section, the described resampling algorithms in combination with R-NSGA-II are evaluated on two benchmark functions. Stepwise, more and more advanced algorithms using multiple resampling criteria are compared in different configurations, showing superior results. To facilitate comparison, the experiments are grouped into experiments where no preference information is used for resampling and experiments where the resampling algorithm uses information about the distance to a reference point r defined for R-NSGA-II. For reasons of simplicity, only experiments with one reference point are run, even though R-NSGA-II is capable of guiding multiple subpopulations to different reference points in the objective space. Also, the reference point is chosen to be infeasible, in order to keep R-NSGA-II and the preference-based resampling algorithms as simple as possible. The combinations of R-NSGA-II with different resampling strategies are tested on three bi-objective benchmark functions. ZDT1, ZDT4 (Zitzler et al., 2000), and a variant of ZDT1 with a Pareto set which is not found on the boundaries of the input space and thereby more difficult to solve: ZDT1-NBPS-α, with $\alpha > 0$. The definition of ZDT1-NBPS-α is given in the next section. The ZDT1 function is used for evaluation due to its popularity in the literature. ZDT4 is more difficult to solve and features many local Pareto-fronts. The complexity of ZDT1-NBPS-α is in between the complexity of ZDT1 and ZDT4. The applicability of Dynamic Resampling algorithms requires a minimum time available for optimization. If convergence to the Pareto-front and a reference point is too fast, the sampling allocation is not able to save evaluations and distribute them in a beneficial way. The time until convergence is correlated to function complexity. Using a graded set of benchmark functions with different complexity therefore allows a more detailed analysis of the algorithm behavior.

The ZDT benchmark functions are deterministic in their original version. In order to create noisy problems, a zero-mean normal distribution is added onto both objective functions.

9.1. Problem settings

The used benchmark functions are deterministic in their original version. Therefore, zero-mean normal noise has been added to create noisy optimization problems. The ZDT1 objective functions are for ex. defined as $f_1(x) = x_1 + \mathcal{N}(0, \sigma_1)$ and $f_2(x) = g(x)\left(1 - \sqrt{x_1/g(x)}\right) + \mathcal{N}(0, \sigma_2)$, where $g(x) = 1 + 9\sum_{i=2}^{30} x_i/29$. The definition of ZDT1-NBPS-α is the same as for ZDT1, except for the $g(x)$ function as in Equation 21. In this article we use it as ZDT1-NBPS-0.5 and call it short ZDT1-H.

$$g(x) = 1 + 9\sum_{i=2}^{30} |x_i - \alpha|/29, \quad \alpha \in [0, 1]. \tag{21}$$

For the ZDT1 and ZDT4 functions, the two objectives have different scales. Therefore, the question arises whether the added noise should be normalized according to the objective scales. We consider the case of noise strength relative to the objective scale as realistic which can occur in real-world problems, and therefore this type of noise is evaluated in this article. For the ZDT1 function, the relative added noise (20%) is $(\mathcal{N}(0, 0.2), \mathcal{N}(0, 2))$ (considering the relevant objective ranges of $[0, 1]\times[0, 10]$), and for ZDT4 it is $(\mathcal{N}(0, 0.2), \mathcal{N}(0, 20))$ (relevant objective ranges $[0, 1] \times [0, 100]$). In the following, these problems are called ZDT1-20%, ZDT1-H-20%, and ZDT4-20%.

9.2. Algorithm parameters

The limited simulation budget is chosen as 5,000 solution replications for ZDT1 and ZDT1-H, and 10,000 replications for the ZDT4 problem. The production line problems are run with $B = 10,000$. This corresponds to a 1 day optimization runtime on a cluster with 100 computers/cores and a 15 minutes function evaluation time, which could be a realistic real-world optimization scenario. However, the benchmark functions and simplistic production line models run much faster, which allows fast experimentation. R-NSGA-II is run with a crossover rate $p_c = 0.8$, SBX crossover operator with $\eta_c = 2$, Mutation probability $p_m = 0.07$ and Polynomial Mutation operator with $\eta_m = 5$. The Epsilon clustering parameter is chosen as

$\epsilon = 0.001$. For ZDT1, ZDT1-H and ZDT4 the reference point $r = (0.05, 0.5)$ is used, which is close to the Ideal Point (0, 0). For the stochastic production line model variants, two different experiments are run, with $r = (10, 35)$ for the first experiment and $r = (30, 40)$ for the second experiment.

Since there is no perfect parameter configuration for Dynamic Resampling algorithms that works well on all optimization problems, we chose one configuration that seemed most intuitive to us and used it for all algorithms on all the experiments (for example, all algorithms that use a time-based allocation are configured with the same parameters for time-based resampling). Instead of performing an individual parameter tuning for one specific optimization problem, we use the same intuitive parameter configuration for all Dynamic Resampling algorithms and compare the results on five different optimization problems and problem variants with different noise levels. Thereby, we can identify which algorithms show a tendency to perform better in most cases. However, some algorithm parameters however have to be adapted according to the problem characteristics, but the same value is set for all algorithms. At the end of this article, we present a study about varying noise levels, where we adapt several parameters according to the noise level of the optimization problem: b_{max}, B, and b_f. For experiments with the ZDT4-20% function, we set the minimum budget to be allocated to $b_{min} = 1$ and the maximum budget to $b_{max} = 20$ for all Dynamic Resampling algorithms. For all other experiments with 20% noise level, we set $b_{max} = 15$ for all Dynamic Resampling algorithms. Static Resampling is run in configurations with b_s between 1 and 5. Time-based Resampling uses a linear allocation, $a = 1$. Rank-based Resampling and Rank-Time-based Resampling are run as RankMax5-based Dynamic Resampling and use linear allocation ($a = 1$) for both the rank-based and time-based criteria. Rank-Time-based Resampling uses the minimum of the Pareto-rank-based allocation and the time-based allocation: $x_s^{RT} = \min\{x_s^T, x_s^R\}$. Progress-based Dynamic Resampling is done with the average progress $\overline{P}$ of the last $n = 3$ populations. Distance-Progress-Time-based Dynamic Resampling DDR uses delayed ($a = 2$) distance-based allocation. Distance-Rank-based Dynamic Resampling DR2 uses the same parameters as the underlying resampling algorithms.

9.3. Evaluation, replication, and interpolation

In order to obtain a reliable performance measurement, B_F final samples are added to the result population. Furthermore, to be able to measure the performance over time, for some experiments, the accurate objective values for each evaluated solution are used. For the benchmark problems, either the added noise landscape can be removed and the solution is evaluated deterministically, or the solution can be sampled a great number of times to obtain accurate objective values, which was done in this study. Calculating 2500 samples on a benchmark problem solution reduces the uncertainty of the objective values by a factor of 50. For the production line models, the simulation time is around 1 second, which is tremendously longer than for the benchmark problems. Therefore, only 100 objective value samples are run for each solution, reducing the uncertainty by a factor of 10.

Table 2: F-HV configuration. Cylinder radius $r = c\,|P|/2\epsilon = 0.025c$.

	R	HV-R	HV-B	D	r
ZDT1	0.05, 0.5	0.1, 1.5	0, 0.5	0.06, 1.5	0.05
ZDT1-H	0.05, 0.5	0.2, 2	0.05, 1	0.2, 2	0.05
ZDT4	0.05, 0.5	0.1, 50	0, 0	0.1, 50	0.05
PL-NM	30, 40	55, 35	30, 40	55, 35	0.05
PL-NS	10, 35	20, 30	10, 35	20, 30	0.025

All experiments performed in this study are replicated 10 times and median performance metric values are calculated. To be able to see the performance development over time, a performance metric is evaluated after every generation of the optimization algorithm. This is shown in Figure 9. However, due to the resampling, each generation uses a different number of solution evaluations. Since it is assumed that solution evaluations are equally long and that the runtime of the optimization algorithm and resampling algorithms is negligible, compared to the evaluation time, the number of solution evaluations corresponds to the optimization runtime. Therefore, an interpolation is required, which calculates the performance metric values at equidistant evaluation number intervals, where the mean performance measure values for all experiment replications can be calculated. Not only does the number of solution evaluations per generation (measurement points) differ between experiment replications, but also between different experiments with different resampling algorithms of the same optimization problem, which shall be compared.

In the following, we specify the parameters of the metrics used to measure algorithm performance.

9.3.1. Performance measurement

The parameters of the F-HV metric is given in Table 2. The same cylinder radii are used for the calculation of the F-IGD, F-RC, and F-Div metrics. The R-NSGA-II experiments use an ϵ value of 0.001, which results in a cylinder radius of $r = |P|/2\epsilon = 25 \cdot 0.001 = 0.025$. For some experiments, however, an expanded cylinder ($c = 2$) is used with radius $r = 2\,|P|/2\epsilon = 0.05$. This is because the objective noise of some experiments causes a wide spread within the algorithm population and only a few objective vectors are found within the F-HV cylinder, especially in the early phase of the optimization process. In addition, the limited budget and the noise do not allow the algorithm to attain the optimal solution set along the true Pareto-front with distance ϵ in between the objective vectors. In order to be able to include more objective vectors in the measurements during the whole optimization process, the cylinder radius is doubled for some experiments.

F-IGD is used to assess the performance on the benchmark functions ZDT1, ZDT1-H, and ZDT4. All three functions have the same Pareto-front. Therefore, the reference Pareto-front used is identical for all three functions. If the normalized cylinder radius r is chosen as $r = 0.025$ then ZDT1, ZDT1-H, and

ZDT4 have f_1 values in [0.0401, 0.0842]. If r is chosen as $r = 0.05$ the three functions have f_1 values in [0.0351, 0.1155].

9.4. Results

In this section, the resampling algorithms from Sections 5 and 6 are evaluated and compared.The noise level of the used optimization problems is 20% of the relevant objective space dimensions. This means that the objective standard deviations for the ZDT1 problem are $(\sigma_1, \sigma_2) = (0.2, 2)$. The same noise level has been used in Syberfeldt et al. (2010).

The results are presented in figures measuring performance metric values continously during the optimization runtime and result tables which contain the metric values for the final non-dominated set or population. For purposes of clarity, the graphical results are split into figures comparing the results of the experiments with generally applicable resampling algorithms, described in Section 5, and figures comparing the results of the preference-based Dynamic Resampling algorithms described in Section 6.

9.4.1. Benchmark functions

In Figure 9, the results of the different resampling algorithms, together with R-NSGA-II, are evaluated on the ZDT1-5% problem with reference point (0.05, 0.5) and 5,000 function evaluations. The results show that Static Resampling with 1 sample is both better than Time-based and Rank-based Dynamic Resampling. Static2-Resampling is worse than Static1-Resampling, which shows that Dynamic Resampling is required to achieve a performance gain over the Static1 strategy. This is achieved by the hybrid strategy Rank-Time-based Resampling which outperforms all others. Table 3 shows the final performance metric values for the four different metrics for the ZDT1-20% problem.

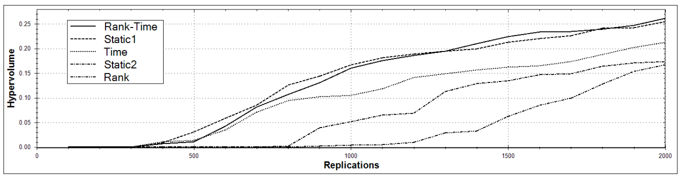


Figure 9: Focused Hypervolume chart showing the R-NSGA-II progress development over time on ZDT1-5% for resampling methods that do not rely on preference information. Reference point (0.05, 0.5).

Next, the resampling algorithms from Section 6 are evaluated and compared: Distance-Progress-Time-based Dynamic Resampling DDR, and Distance-Rank(-Progress-Time)-based Dynamic Resampling DR2. The results for Rank-Time-based Resampling are included for comparison purposes.

In Figure 10, the results of the different resampling algorithms, together with R-NSGA-II, are evaluated on the ZDT1-5% problem with reference point (0.05, 0.5) and 5,000 function evaluations. The results show that DDR is slightly better than Rank-Time-based Resampling. However, DR2 performs slightly worse than Rank-Time-based Resampling. As a reason, we can see that the ZDT1-5% problem is not sufficiently complex (short convergence time) and does not allow DR2 to develop its full potential.

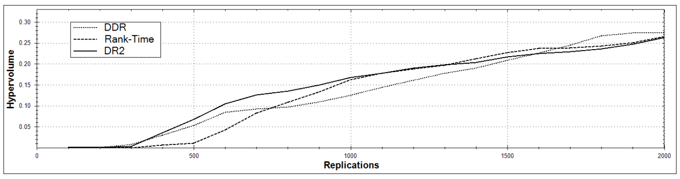


Figure 10: Focused Hypervolume chart showing the R-NSGA-II progress development over time on ZDT1-5% for resampling methods that use preference information. Reference point (0.05, 0.5). For comparison with the non-preference methods, the curve for Rank-Time-based resampling from Figure 9 is included.

Table 3: ZDT1-20% benchmark function, reference point (0.05, 0.5): Performance measurement results of Dynamic Resampling algorithms on R-NSGA-II that do not consider preference information, followed by algorithms that use a reference point. The measurement is performed on the last population where $b_f = 25$ final samples have been executed.

	F-HV	F-IGD	F-RC	F-Div
Static1	0.3393	0.0411	0.1359	0.0150
Static2	0.3092	**0.0403**	0.1112	0.0146
Static3	0.3469	0.0408	0.1172	0.0191
Static4	0.2503	0.0576	0.1463	0.0264
Static5	0.1944	0.0647	0.1747	0.0397
Time 1-15	**0.3744**	0.1068	0.1332	0.0520
Rank 1-15	0.2567	0.0566	0.1383	0.0279
R5 1-15	0.2707	0.0539	0.1375	0.0277
R5T 1-15	0.2927	0.0416	**0.1093**	**0.0125**
P 1-15	0.2289	0.0478	0.1327	0.0306
PT 1-15	0.2535	0.0434	0.1301	0.0285
DDR 1-15	0.3295	**0.0394**	0.1232	0.0314
DR2 1-15	**0.3780**	0.0538	**0.1023**	**0.0120**

In Figure 11, the results of the different generally applicable Dynamic Resampling algorithms, together with R-NSGA-II, are evaluated on the ZDT1-H-5% problem with reference point (0.05, 0.5) and 10,000 function evaluations. The results show that Static Resampling with 1 sample is both better than Time-based and Rank-based Dynamic Resampling. Static2-Resampling is worse than Static1-Resampling, which shows that Dynamic Resampling is required to achieve a performance gain over the Static1 strategy. This is achieved by the hybrid strategy Rank-Time-based Resampling, which outperforms all others. Table 4 shows the final performance metric values for the four different metrics for the ZDT1-H-20% problem.

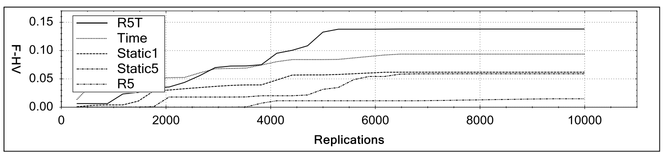


Figure 11: Focused Hypervolume chart showing the R-NSGA-II progress development over time on ZDT1-H-5% for resampling methods that do not rely on preference information. Reference point (0.05, 0.5).

Table 4: ZDT1-H-20% benchmark function, reference point (0.05, 0.5): Performance measurement results of Dynamic Resampling algorithms on R-NSGA-II that do not consider preference information, followed by algorithms that use a reference point. The measurement is performed on the last population where $b_f = 25$ final samples have been executed.

	F-HV	F-IGD	F-RC	F-Div
Static1	0.2101	0.1407	0.2292	0.0701
Static2	0.2619	0.1288	0.2208	0.0388
Static3	0.2549	0.1188	0.1940	**0.0214**
Static4	0.1728	0.1385	0.2218	0.0364
Static5	0.0734	0.1439	0.2305	0.0384
Time 1-15	**0.3527**	0.1252	0.1888	0.0397
Rank 1-15	0.2553	0.1275	0.2045	0.0477
R5 1-15	0.2634	0.1278	0.2037	0.0456
R5T 1-15	0.2921	**0.1097**	**0.1847**	0.0406
P 1-15	0.1819	0.1265	0.2068	0.0537
PT 1-15	0.2573	0.1259	**0.2016**	0.0498
DDR 1-15	0.3701	**0.1125**	0.2218	**0.0250**
DR2 1-15	**0.3826**	0.1219	0.2349	0.0390

In Figure 12, the results of the different preference-based Dynamic Resampling algorithms, together with R-NSGA-II, are evaluated on the ZDT1-H-5% problem with reference point (0.05, 0.5) and 10,000 function evaluations. Since the ZDT1-H-5% problem is more difficult, it allows for a more clear evaluation. Here, it can be seen very clearly that DR2 outperforms Rank-Time-based Resampling, and thereby all other resampling algorithms evaluated in Figure 11. DDR, however, is shown not to be very powerful on this problem. Yet, combined with the Pareto-rank criterion as DR2, it is superior to all others.

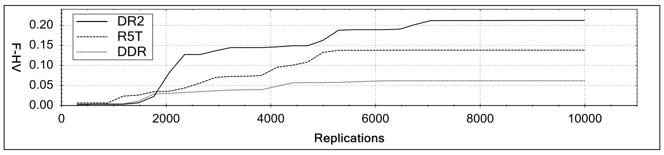


Figure 12: Focused Hypervolume chart showing the R-NSGA-II progress development over time on ZDT1-H-5% for resampling methods that use preference information. Reference point (0.05, 0.5). For comparison with the non-preference methods, the curve for Rank-Time-based resampling from Figure 11 is included.

In Figure 13, the results of the different generally applicable Dynamic Resampling algorithms, together with R-NSGA-II, are evaluated on the ZDT4-5% problem with reference point (0.05, 0.5) and 5,000 function evaluations. The results confirm the findings in Figure 9, however, the differences between the various resampling algorithms become more clear, since the ZDT4-5% problem is more difficult (many local Pareto-fronts) and needs more time to converge to the reference point. Table 5 shows the final performance metric values for the four different metrics for the ZDT4-20% problem.

In Figure 14, the results of the different preference-based Dynamic Resampling algorithms, together with R-NSGA-II, are evaluated on the ZDT4-5% problem with reference point

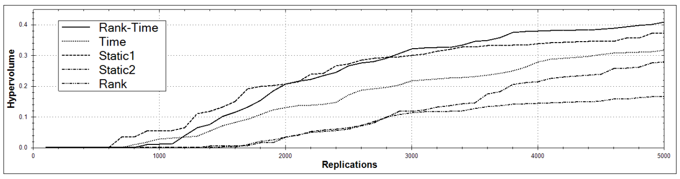


Figure 13: Focused Hypervolume chart showing the R-NSGA-II progress development over time on ZDT4-5% for resampling methods that do not rely on preference information. Reference point (0.05, 0.5).

Table 5: ZDT4-20% benchmark function, reference point (0.05, 0.5): Performance measurement results of Dynamic Resampling algorithms on R-NSGA-II that do not consider preference information, followed by algorithms that use a reference point. The measurement is performed on the last population where $b_f = 25$ final samples have been executed.

	F-HV	F-IGD	F-RC	F-Div
Static1	0.2260	0.1226	0.1620	0.0429
Static2	0.4553	0.1172	**0.1233**	**0.0031**
Static3	0.0342	0.1705	0.2377	0.0155
Static4	0.0000	0.2147	0.2739	0.0395
Static5	0.0000	0.2408	0.2947	0.0476
Time 1-15	0.1931	0.2067	0.2211	0.0211
Rank 1-15	0.1557	0.1683	0.2009	0.0379
R5 1-15	0.1694	0.1622	0.1956	0.0393
R5T 1-15	**0.5005**	**0.0770**	0.1733	0.0358
P 1-15	0.2117	0.1357	0.2176	0.0354
PT 1-15	0.2370	0.1178	0.1998	0.0336
DDR 1-15	0.3696	**0.0831**	**0.1224**	**0.0211**
DR2 1-15	**0.5285**	0.0935	0.1453	0.0283

(0.05, 0.5) and 5,000 function evaluations. Since the ZDT4-5% problem is even more difficult, it allows for a more clear evaluation. Here, it can be seen very clearly that DR2 outperforms Rank-Time-based Resampling, and thereby all other resampling algorithms evaluated in Figure 13. DDR, however, is shown not to be very powerful, also on this problem. Yet, combined with the Pareto-rank criterion as DR2, it is superior to all other Dynamic Resampling algorithms on R-NSGA-II.

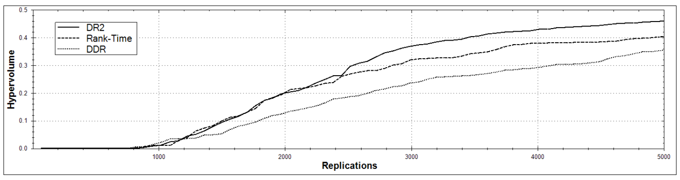


Figure 14: Focused Hypervolume chart showing the R-NSGA-II progress development over time on ZDT4-5% for resampling methods that use preference information. Reference point (0.05, 0.5). For comparison purposes, the curve for Rank-Time-based resampling from Figure 13 is included.

9.4.2. Results of the PL-NM model

In the following, the results of the stochastic production line models are presented and analyzed with different types of figures and corresponding data.

Figure 15 shows the attainment surface of the different resampling algorithms, together with R-NSGA-II, on the PL-NM-20% problem with reference point (TNB, TH)=(30,40) and 10,000 function evaluations. The results show that DDR performs slightly better than DR2. Together, they show a better performance than all other algorithms.

In Figure 16, the F-HV measurement results over time on PL-NM-20% of the resampling algorithms, together with R-NSGA-II, are shown. The results indicate that DDR and DR2 outperform the other Dynamic Resampling algorithms. Towards the end of the runtime, a drop in the F-HV measurement values can be observed. This is due to the final samples that are added to the last population. Table 6 shows the measurement values for the final population of the different metrics.

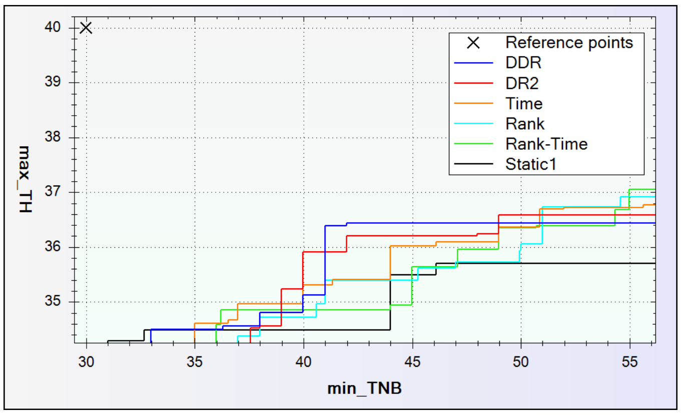


Figure 15: PL-NM-20% model results shown as median attainment surface. 10,000 evaluations and Reference point (TNB, TH)=(30,40).

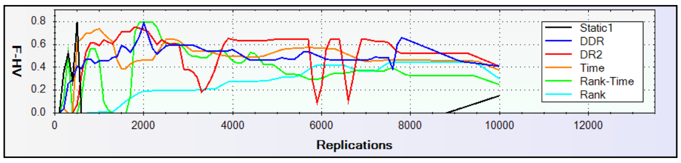


Figure 16: PL-NM-20% model normalized optimization results of 10,000 replications shown as median Focused Hypervolume. The error bars show the minimum and maximum values of the experiment replications. The data is based on accurate objective vectors and is interpolated according to intervals of 100 replications. Reference point (TNB, TH)=(30,40).

Figure 17 presents the F-RC measurement results over time on PL-NM-20% of the resampling algorithms, together with R-NSGA-II. Time-based Dynamic Resampling outperforms the Static Resampling with 1 evaluation for each solution. Towards the end of the runtime, a rise in the F-RC measurement values can be observed for Static1 Resampling. This is due to the final samples that are added to the last population which move the uncertain objective vectors of the Static1 experiment further away from the reference point.

Figure 18 shows the average sampling allocation for the different resampling algorithms over time on the PL-NM-20% simulation model. Rank-based Dynamic Resampling allocates a high number of samples during the whole optimization time. The allocation of Time-based Resampling is rising continuously. At the DDR allocation curve, it can be seen that the allocation

Table 6: Stochastic production line model PL-NM-20%, reference point (TNB, TH)=(30,40): Performance measurement results of Dynamic Resampling algorithms on R-NSGA-II that do not consider preference information followed by algorithms that use a reference point. The measurement is performed on the last population where $b_f = 25$ final samples have been executed. Average runtime.

	F-HV	F-RC	F-Div
Static1	0.0611	0.1638	0.0382
Static2	0.0505	0.1526	0.0392
Static3	0.0565	0.1349	0.0389
Static4	0.0467	0.1312	0.0404
Static5	0.0431	0.1315	0.0462
Time 1-15	0.1381	0.1584	0.0349
Rank 1-15	0.0643	0.1383	0.0475
R5 1-15	0.0627	0.1309	0.0514
R5T 1-15	**0.0996**	**0.1135**	**0.0294**
P 1-15	0.0934	0.1567	0.0452
PT 1-15	0.1006	0.1373	**0.0419**
DDR 1-15	0.1610	**0.1223**	0.0422
DR2 1-15	**0.1618**	0.1284	0.0465

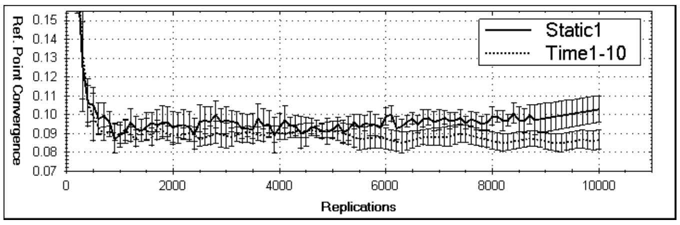


Figure 17: PL-NM-20% model normalized optimization results of 10,000 replications shown as median Focused Reference Point Convergence of 10 experiments each. The error bars show the minimum and maximum values of the experiment replications. The data is based on accurate objective vectors and is interpolated according to intervals of 100 replications. Reference point (TNB, TH)=(30,40).

can drop, if the population temporarily moves away from the reference point (at approx. 7,000 evaluations).

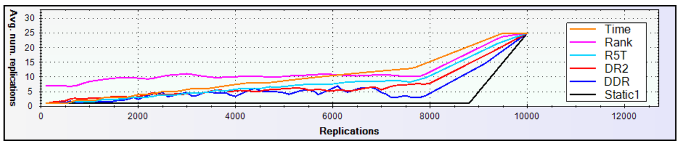


Figure 18: PL-NM-20% model average number of replications until 10,000 replications interpolated according to intervals of 100 replications. $b_{min} = 1$, $b_{max} = 15$, and $b_f = 25$.

On the other hand, Figures 19 to 23 show resampling allocations for single experiments. They allow for a detailed analysis and comparison of the algorithm behaviour and enable us to verify the behavior of the algorithms. The allocation of Time-based Dynamic Resampling in Figure 19 shows allocation steps which become shorter and shorter as the optimization progresses, since at a higher number of samples, fewer solutions can be evaluated until the next step is reached. It shall be mentioned that if the next step is reached during the evaluation of a generation, all solutions will be evaluated with the higher number of samples. This ensures a fair distribution of samples among the unordered list of solutions in the population.

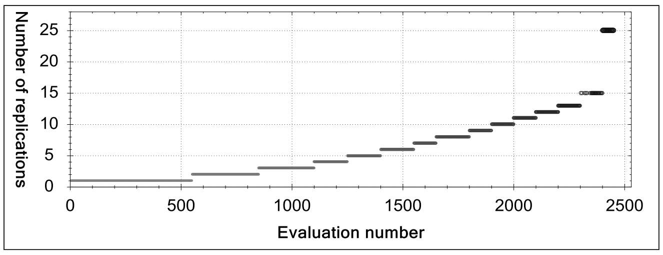


Figure 19: PL-NM-20% model number of replications per solution until 10,000 replications for one optimization run of Time-based Dynamic Resampling on R-NSGA-II. $b_{min} = 1$, $b_{max} = 15$, and $b_f = 25$. Reference point (TNB, TH)=(30,40).

It can be seen, that with an average of around 8 samples per solution the Rank-based experiment (Figure 20) only has around 1200 solutions available. This is only around half of the solutions as for the Time-based experiment in Figure 19. The experiment with the Rank-Time-based Resampling can avoid this drawback by adding less samples than Rank-based DR in the beginning, and providing a better allocation with less samples than Time-based DR in the end of the optimization run.

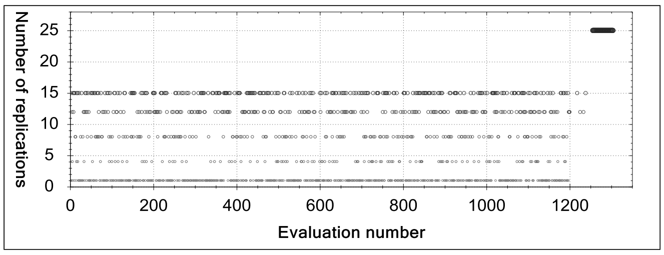


Figure 20: PL-NM-20% model number of replications per solution until 10,000 replications for one optimization run of RankMax5-based Dynamic Resampling on R-NSGA-II. $b_{min} = 1$, $b_{max} = 15$, and $b_f = 25$. Reference point (TNB, TH)=(30,40).

The Rank-Time-based allocation is shown in Figure 21. The minimum of the Pareto-rank-based and Time-based allocated samples is used: $x_s^{R5T} = \min\{x_s^T, x_s^{R5}\}$. This is the allocation used in the experiments for this article. An alternative allocation is to use the product of the Pareto-rank-based and Time-based for allocation: $x_s^{R5T} = x_s^T x_s^{R5}$, depicted in Figure 22. This allows to assign samples more differentiated earlier during the optimization runtime, which saves sampling budget and allows to run around 400 more unique solutions than in the case of Figure 21. However, initial experiments shows better performance metric results for the allocation in Figure 21 and therefore was not used in this article.

In Figure 23 the sampling allocation of an optimization run with Distance-based Dynamic Resampling is shown. The initial increase in allocated samples until 400 solution evaluations is caused by the reducing distance to the reference point. At around 1,000 evaluations the population progress was negative, leading to a more conservative allocation. But starting from around 1,100 evaluations, a large part of the population is assigned a high number of samples.

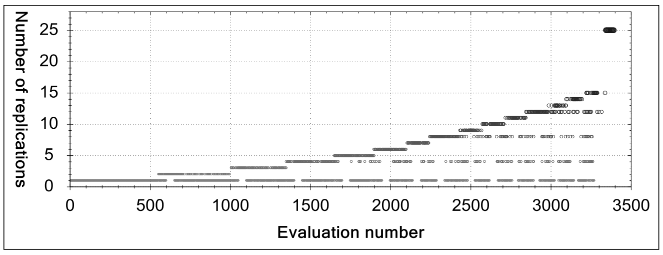


Figure 21: PL-NM-20% model number of replications per solution until 10,000 replications for one optimization run of RankMax5-Time-based Dynamic Resampling on R-NSGA-II. The minimum of the Pareto-rank-based and Time-based allocated samples is used: $x_s^{R5T} = \min\{x_s^T, x_s^{R5}\}$. $b_{min} = 1$, $b_{max} = 15$, and $b_f = 25$. Reference point (TNB, TH)=(30,40).

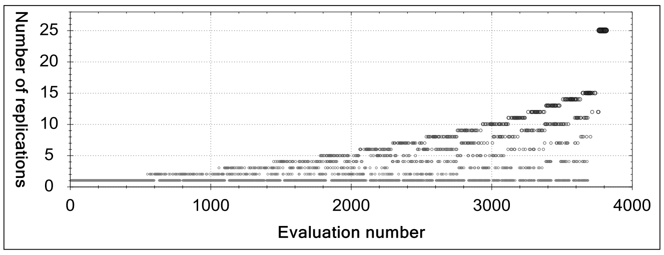


Figure 22: PL-NM-20% model number of replications per solution until 10,000 replications for one optimization run of RankMax5-Time-based Dynamic Resampling on R-NSGA-II as in Figure 21, but for the sampling allocation the product of the Pareto-rank-based and Time-based allocation is used: $x_s^{R5T} = x_s^T x_s^{R5}$. $b_{min} = 1$, $b_{max} = 15$, and $b_f = 25$. Reference point (TNB, TH)=(30,40).

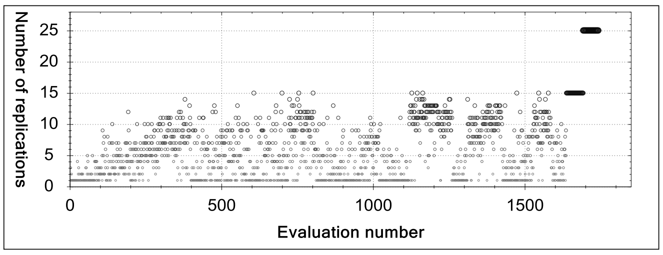


Figure 23: PL-NM-20% model number of replications per solution until 10,000 replications for one optimization run of Distance-based Dynamic Resampling on R-NSGA-II. $b_{min} = 1$, $b_{max} = 15$, and $b_f = 25$. Reference point (TNB, TH)=(30,40).

Figures 24, 25, and 26 show the objective space of the PL-NM-20% model after 10,000 solution evaluations with all solutions found during the optimization runtime. The bigger circles mark the position in the objective space of the solutions in the result population, after the final b_f samples have been added. A fallback can be observed for those solutions. This effect is strongest for the Static1 Dynamic Resampling algorithm (Figure 24). The Static5 algorithm has only 2,000 evaluations available for the optimization since each solution is evaluated five times. Therefore, it can only attain a small area in the objective space with the final population (Figure 25). The Time-based

Resampling algorithm in Figure 26 performs best, since it can achieve a trade-off between exploration of the objective space and exploitation of better knowledge of the objective vectors achieved through resampling.

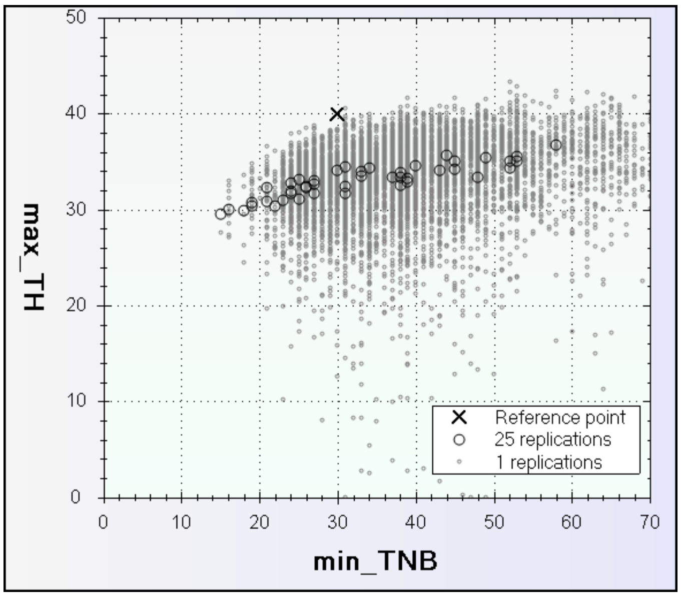


Figure 24: PL-NM-20% model R-NSGA-II designs in objective space. No resampling algorithm was used. Reference point (TNB, TH)=(30,40).

9.4.3. *Results of the PL-NS model*

Figure 27 shows the attainment surface of the different resampling algorithms, together with R-NSGA-II, on the PL-NS-20% problem with reference point (TNB, TH)=(10,35) and 10,000 function evaluations. The results show that DDR and DR2 perform equally well and better than all other algorithms. Time-based Resampling and Rank-Time-based Resampling also show a promising performance.

In Figure 28, the F-HV results on PL-NS-20% of the Dynamic Resampling algorithms, together with R-NSGA-II, are shown. The results indicate that DR2 shows the best F-HV performance. The second best result is achieved by Rank-Time-based Dynamic Resampling. Towards the end of the runtime, a drop in the F-HV measurement values can be observed. This is due to the final samples that are added to the last population. Table 7 shows the measurement values for the final population of the different metrics.

Figure 29 illustrates the sampling allocation for the different resampling algorithms over time on the PL-NS-20% simulation model. Rank-based Dynamic Resampling allocates a high number of samples during the whole optimization time. The allocation of Time-based Resampling is rising continuously.

Table 8 shows a comparison of the F-HV metric values for the final population of all optimization problems run in this paper.

9.5. *Varying noise strength*

In this section, we perform a study on how the algorithm performance changes under different levels of noise. This al-

Table 7: Stochastic production line model PL-NS-20%, Source interval: [30s,240s], reference point (TNB, TH)=(10,35): Performance measurement results of Dynamic Resampling algorithms on R-NSGA-II that do not consider preference information followed by algorithms that use a reference point. The measurement is performed on the last population where $b_f = 25$ final samples have been executed. Average runtime.

	F-HV	F-RC	F-Div
Static1	0.3034	0.1701	0.0433
Static2	0.4347	0.1726	0.0201
Static3	0.4058	0.1713	0.0184
Static4	0.3640	0.1532	0.0206
Static5	0.2709	0.1721	0.0236
Time 1-15	0.4885	0.1423	**0.0068**
Rank 1-15	0.4552	0.1468	0.0206
R5 1-15	0.4597	0.1401	0.0206
R5T 1-15	**0.5680**	**0.1391**	0.0151
P 1-15	0.4047	0.1558	**0.0153**
PT 1-15	0.4237	0.1406	0.0174
DDR 1-15	0.4779	0.1542	0.0443
DR2 1-15	**0.6693**	**0.1347**	0.0194

Table 8: F-HV results for the final non-dominated result front of Dynamic Resampling algorithms on R-NSGA-II. The problem noise level was 20% and 25 final samples were run on each member of the non-dominated front.

F-HV	ZDT1	ZDT1-H	ZDT4	PL-NM	PL-NS
Static1	0.3393	0.2101	0.2260	0.0611	0.3034
Static2	0.3092	0.2619	0.4553	0.0505	0.4347
Static3	0.3469	0.2549	0.0342	0.0565	0.4058
Time 1-15 / 20	0.3744	**0.3527**	0.1931	0.1381	0.4885
Rank 1-15 / 20	0.2097	0.2357	0.2513	0.0643	0.4552
R5 1-15 / 20	0.2256	0.2386	0.2891	0.0627	0.4597
R5T 1-15 / 20	**0.2927**	0.2912	**0.5005**	**0.0996**	**0.5680**
P 1-15 / 20	0.2371	0.2509	0.1381	0.0934	0.4047
PT 1-15 / 20	0.2703	0.2703	0.1857	0.1006	0.4237
DDR 1-15 / 20	0.3295	0.3701	0.3696	0.1610	0.4779
DR2 1-15 / 20	**0.3780**	**0.3826**	**0.5285**	**0.1618**	**0.6693**

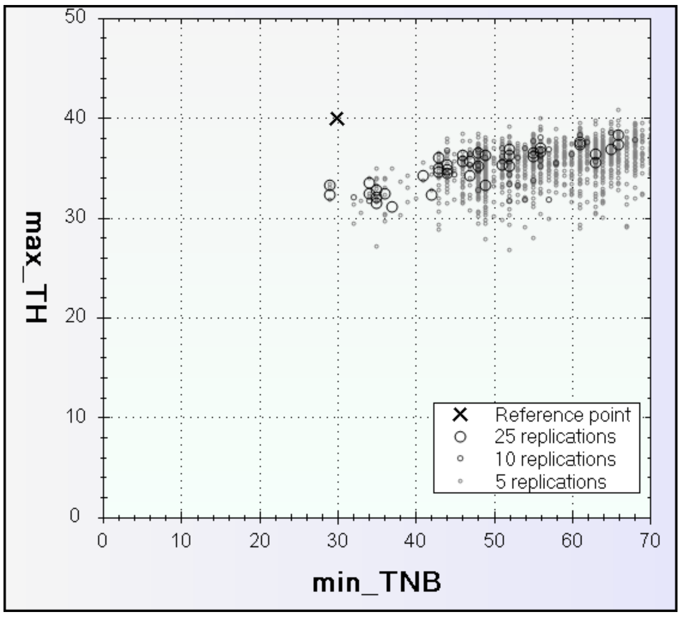


Figure 25: PL-NM-20% model Dynamic Resampling R-NSGA-II designs in objective space. Static Resampling with 5 samples was used. Reference point (TNB, TH)=(30,40).

lows us to present different parameter configurations that perform well under the different noise circumstances. In particular, the relationship between the objective noise level of the problem and the total simulation budget and maximum allowed number of samples is illustrated. Another important problem characteristic to consider is the problem complexity (time until convergence) which requires adaption of the delayed allocation parameters.

Four different noise levels are defined: 5%, 10%, 20%, and 30%, and three different optimization problems of increasing complexity are used: ZDT1, ZDT1-H, and ZDT4. Also, the results are verified on the production line simulation models. The results of the benchmark functions are shown in Figures 30, 31, and 32. The performance on the industrial models is documented in Figures 33 and 34, and the F-HV measurement values of the final population of all experiments are listed in Table 9.

As a result, it can be seen that the Rank-Time-based Resampling, the Distance-based Resampling DDR, and the Distance-Rank-based Resampling perform best. For the ZDT1 function with its low complexity and all experiments with the lowest noise level 5%, this conclusion cannot be drawn, but it becomes more clear as the noise level rises.

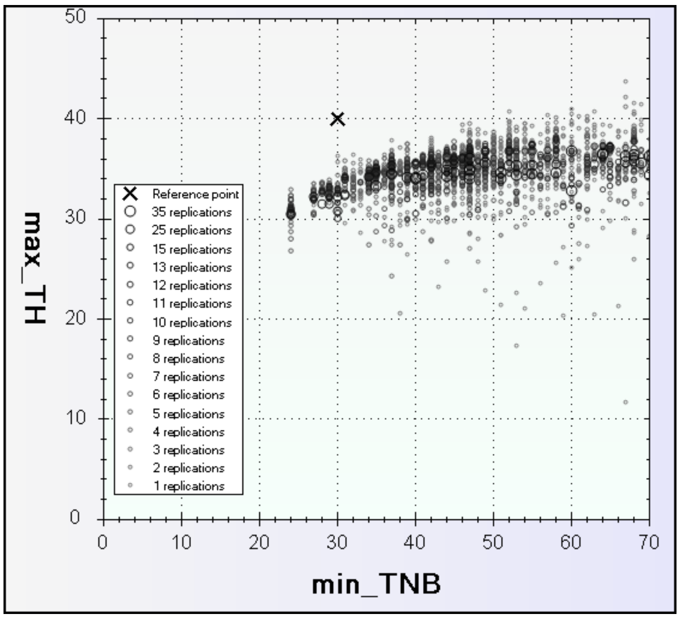


Figure 26: PL-NM-20% model R-NSGA-II designs in objective space. Time-based Dynamic Resampling with maximum 15 samples was used. Reference point (TNB, TH)=(30,40).

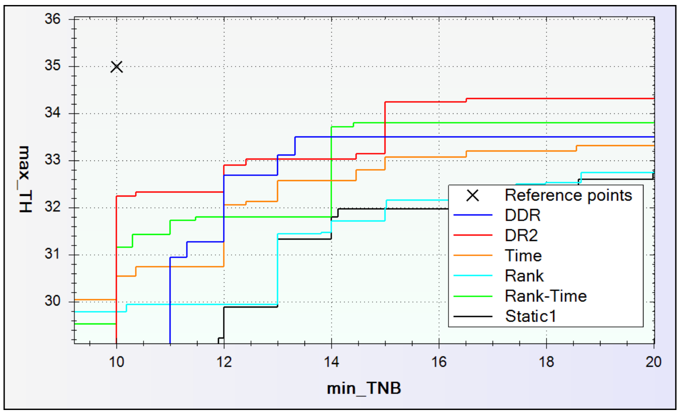


Figure 27: PL-NS-20% model results shown as median attainment surface. Reference point (TNB, TH)=(10,35).

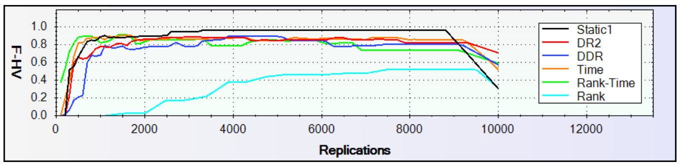


Figure 28: PL-NS-20% model normalized optimization results of 10k replications shown as median Focused Hypervolume. The error bars show the minimum and maximum values of the experiment replications. The data is based on accurate objective vectors and is interpolated according to intervals of 100 replications. Reference point (TNB, TH)=(10,35).

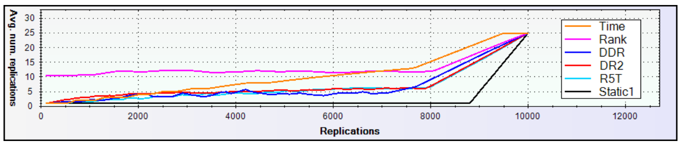


Figure 29: PL-NS-20% model average number of replications until 10,000 replications interpolated according to intervals of 100 replications. $b_{min} = 1$, $b_{max} = 15$, and $b_f = 25$.

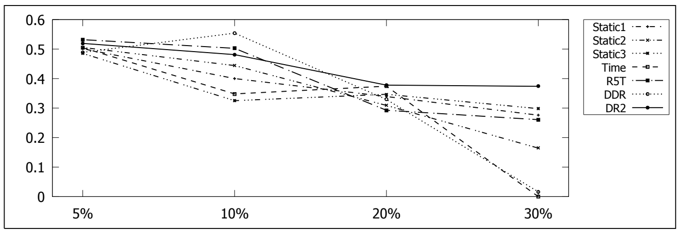


Figure 30: F-HV performance results for different variants of the ZDT1 benchmark function with increasing noise level.

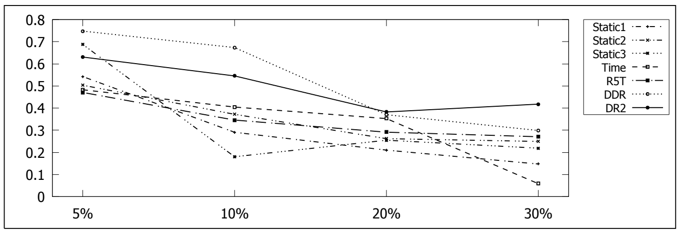


Figure 31: F-HV performance results for different variants of the ZDT1-H benchmark function with increasing noise level.

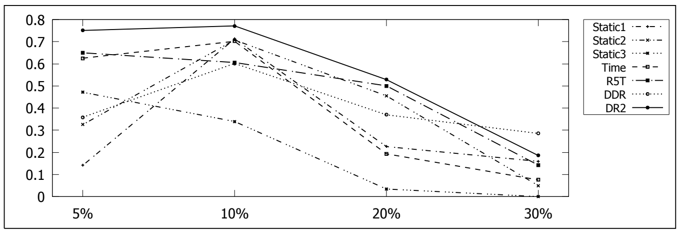


Figure 32: F-HV performance results for different variants of the ZDT4 benchmark function with increasing noise level.

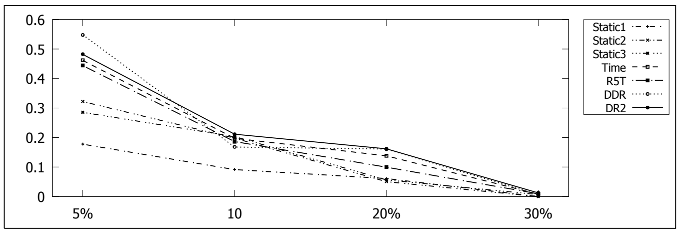


Figure 33: F-HV performance results for different variants of the PL-NM model with increasing noise level.

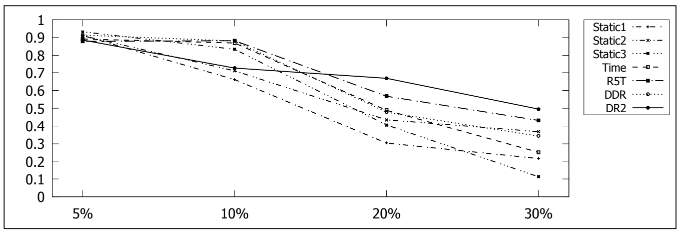


Figure 34: F-HV performance results for different variants of the PL-NS model with increasing noise level.

Table 9: F-HV results of Dynamic Resampling algorithms on R-NSGA-II on benchmark problems at different noise levels. The measurement is done on the non-dominated front of the last generation, without archive, after applying final samples. For the optimization runs on ZDT4 we increased b_{max} by 5 samples.

F-HV	ZDT1, B=5k	ZDT1-H, B=5k	ZDT4, B=10k	PL-NM, B=10k	PL-NS, B=10k
			Noise level 5%, b_f=15		
Static1	0.5013	0.5425	0.1410	0.1776	0.9111
Static2	0.5056	0.5034	0.3251	0.3220	0.9099
Static3	0.4854	0.6877	0.4715	0.2854	**0.9322**
Static4	0.4627	0.5414	0.5546	0.4724	0.9225
Static5	0.4488	0.6620	0.6278	0.4222	0.9166
Static6	0.4115	0.3703	0.4717	0.4716	0.8801
Time 1-5 / 10	0.5048	0.4830	0.6254	0.4615	0.8913
R5T 1-5 / 10	**0.5321**	0.4707	0.6498	0.4439	0.8779
DDR 1-5 / 10	0.4891	**0.7480**	0.3586	**0.5484**	0.9135
DR2 1-5 / 10	0.5194	0.6307	**0.7508**	0.4820	0.8856
			Noise level 10%, b_f=20		
Static1	0.4003	0.2908	0.7125	0.0914	0.6619
Static2	0.4443	0.3719	0.7078	0.1981	0.7116
Static3	0.3253	0.1802	0.3393	0.2033	0.8332
Time 1-10 / 15	0.3482	0.4047	0.7018	0.1975	0.8688
R5T 1-10 / 15	0.5027	0.3459	0.6059	0.1864	**0.8819**
DDR 1-10 / 15	**0.5541**	**0.6736**	0.6017	0.1679	0.8799
DR2 1-10 / 15	0.4814	0.5454	**0.7714**	**0.2114**	0.7272
			Noise level 20%, b_f=25		
Static1	0.3393	0.2101	0.2260	0.0611	0.3034
Static2	0.3092	0.2619	0.4553	0.0505	0.4347
Static3	0.3469	0.2549	0.0342	0.0565	0.4058
Time 1-15 / 20	0.3744	0.3527	0.1931	0.1381	0.4885
R5T 1-15 / 20	0.2927	0.2912	0.5005	0.0996	0.5680
DDR 1-15 / 20	0.3295	0.3701	0.3696	0.1610	0.4779
DR2 1-15 / 20	**0.3780**	**0.3826**	**0.5285**	**0.1618**	**0.6693**
			Noise level 30%, b_f=35		
Static1	0.2761	0.1473	0.1591	0.0000	0.2162
Static2	0.1644	0.2494	0.0495	0.0000	0.3680
Static3	0.2983	0.2189	0.0000	0.0068	0.1132
Time 1-15 / 25	0.0000	0.0585	0.0755	0.0081	0.2516
R5T 1-15 / 25	0.2609	0.2705	0.1423	0.0095	0.4314
DDR 1-15 / 25	0.0163	0.2988	**0.2855**	0.0063	0.3429
DR2 1-15 / 25	**0.3739**	**0.4179**	0.1856	**0.0124**	**0.4955**

10. Conclusions and future work

We have proposed and evaluated Dynamic Resampling strategies that use different resampling criteria on the guided EMO algorithm R-NSGA-II under a limited simulation budget. The results on benchmark problems and industrial simulation models show that it is beneficial to add more samples towards the end of the optimization and that using the Pareto-rank information for resampling gives a performance advantage, if it is combined with time-based allocation.

Since hybrid Dynamic Resampling algorithms have shown to be a promising concept, we also proposed Dynamic Resampling Algorithms that use multiple resampling criteria, including the distance to a reference point specified by a decision maker. Examples are Distance-Progress-Time-based Dynamic Resampling (DDR) (Siegmund et al., 2013) which uses the distance and progress to a reference point, and the elapsed optimization runtime for sampling allocation, or Rank-Distance-based Dynamic Resampling (DR2) which combines the DDR allocation with the Pareto-rank-based allocation. The hybrid algorithms are compared with resampling algorithms that base their sampling allocation on a single criterion, such as Time-based Dynamic Resampling or Rank-based Dynamic Resampling. The results on benchmark functions and industrial models show that Hybrid Dynamic Resampling algorithms are superior to single-criterion algorithms and Static Resampling, given that the optimization problem is sufficiently complex, or has a high noise level.

We proposed a new performance metric for reference-point based EMO algorithms, the Focused Hypervolume metric (F-HV). We also proposed a set of other performance metrics, based on the F-HV metric, which allow a more differentiated assessment of the algorithm performance. The new metrics proved to be effective in measuring algorithm performance in an optimization scenario with preference information in the form of a user-specified reference point.

In our future work, we will propose and investigate hybrid dynamic resampling algorithms that consider the objective variance of the solutions. In particular, we will evaluate a multi-objective extension of the Standard Error Dynamic Resampling algorithm by Di Pietro et al. (2004); Di Pietro (2007) which we proposed in Siegmund et al. (2013) and combine it with other resampling criteria.

The Pareto-rank-based Dynamic Resampling algorithms in this article base their sampling allocation on a comparison of solutions. They can thereby have an advantage over resampling algorithms that treat each solution individually. Slightly modified, the comparison approach could support the evolutionary optimization algorithm in comparing solutions for selection decisions, which we call Selection Sampling. A study investigating the effect of Selection Sampling on preference-based EMO of stochastic systems will be performed.

For extended analysis of the distance-based sampling allocation algorithms, future work will cover the evaluation of the Dynamic resampling algorithms for optimization scenarios with feasible reference points.

Table A.10: F-HV results for the final non-dominated result front of Dynamic Resampling algorithms on R-NSGA-II, when a solution archive is used. The problem noise level was 20% and 25 final samples were run on each member of the non-dominated front.

	ZDT1	ZDT1-H	ZDT4	PL-NM	PL-NS
Static1	0.3269	0.2501	0.2819	0.1145	0.3942
Static2	0.3154	0.2931	0.4230	0.0790	0.4532
Static3	0.3708	0.2236	0.1080	0.0576	0.4758
Time 1-15 / 20	0.3278	0.3293	0.2503	**0.1535**	0.4734
Rank 1-15 / 20	0.2122	0.2699	0.2675	0.0799	0.4763
R5 1-15 / 20	0.2376	0.2809	0.3178	0.0933	0.4948
R5T 1-15 / 20	**0.3570**	**0.3457**	**0.5253**	0.1347	**0.6203**
P 1-15 / 20	0.2966	0.2367	0.2548	0.0836	0.4431
PT 1-15 / 20	0.3120	0.2589	0.2989	0.0989	0.4787
DDR 1-15 / 20	0.3700	0.3947	0.4131	0.1972	0.5482
DR2 1-15 / 20	**0.4543**	**0.4277**	**0.5603**	**0.2363**	**0.6978**

Acknowledgments

This study was partially funded by the Knowledge Foundation, Sweden, through the BlixtSim and IDSS projects. Their provision of research funding is gratefully acknowledged.

Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at doi:

References

Aizawa, A. N., & Wah, B. W. (1993). Dynamic control of genetic algorithms in a noisy environment. In *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp. 48–55).

Aizawa, A. N., & Wah, B. W. (1994). Scheduling of genetic algorithms in a noisy environment. *Evolutionary Computation*, *2*, 97–122.

Bartz-Beielstein, T., Blum, D., & Branke, J. (2007). Particle swarm optimization and sequential sampling in noisy environments. In *Metaheuristics – Progress in Complex Systems Optimization* (pp. 261–273). volume 39 of *Operations Research/Computer Science Interfaces Series*.

Branke, J., & Schmidt, C. (2004). Sequential sampling in noisy environments. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature, 2004, Birmingham* (pp. 202–211). volume 3242 of *Lecture Notes in Computer Science*.

Brockhoff, D., Bader, J., Thiele, L., & Zitzler, E. (2013). Directed multiobjective optimization based on the weighted hypervolume indicator. *Journal of Multi-Criteria Decision Analysis*, *20*, 291–317.

Chen, C.-H., He, D., Fu, M., & Lee, L. H. (2008). Efficient simulation budget allocation for selecting an optimal subset. *Journal on Computing*, *20*, 579–595.

Chen, C.-H., & Lee, L. H. (2010). *Stochastic Simulation Optimization - An Optimal Computing Budget Allocation*. World Scientific Publishing Company.

Coello Coello, C. A., & Reyes Sierra, M. (2004). A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In *Proceedings of the 3rd Mexican International Conference on Artificial Intelligence, 2004, Mexico City* (pp. 688–697).

Deb, K., & Gupta, H. (2006). Introducing robustness in multi-objective optimization. *Journal of Evolutionary Computation*, *14*, 463–494.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*, 182–197.

Deb, K., Siegmund, F., & Ng, A. H. C. (2014). R-HV : A metric for computing hyper-volume for reference point based EMOs. In *Proceedings of the 5th International Conference on Swarm, Evolutionary, and Memetic Computing, 2014, Bhubaneswar* (pp. 98–110).

Deb, K., Sinha, A., Korhonen, P., & Wallenius, J. (2010). An interactive evolutionary multiobjective optimization method based on progressively approximated value functions. *IEEE Transactions on Evolutionary Computation*, *14*, 723–739.

Deb, K., Sundar, J., Bhaskara Rao N., U., & Chaudhuri, S. (2006). Reference point based multi-objective optimization using evolutionary algorithms. *International Journal of Computational Intelligence Research*, *2*, 273–286.

Di Pietro, A. (2007). *Optimizing Evolutionary Strategies for Problems with Varying Noise Strength*. Ph.D. thesis University of Western Australia, Perth.

Di Pietro, A., While, L., & Barone, L. (2004). Applying evolutionary algorithms to problems with noisy, time-consuming fitness functions. *Proceedings of the Congress on Evolutionary Computation 2004, Portland*, *2*, 1254–1261.

Dudley, N. A. (1963). Work-time distributions. *International Journal of Production Research*, *2*, 137–144.

Enginarlar, E., Li, J., & Meerkov, S. M. (2005). How lean can lean buffers be? *IIE Transactions*, *37*, 333–342.

Eskandari, H., Geiger, C. D., & Bird, R. (2007). Handling uncertainty in evolutionary multiobjective optimization: SPGA. In *Proceedings of the Congress on Evolutionary Computation 2007, Singapore* (pp. 4130–4137).

Fieldsend, J. E. (2015). Elite accumulative sampling strategies for noisy multi-objective optimisation. In *Proceedings of the 8th International Conference on Evolutionary Multi-Criterion Optimization, 2015, Guimarães* (pp. 172–186). volume 9019 of *Lecture Notes in Computer Science*.

Fieldsend, J. E., & Everson, R. M. (2005). Multi-objective optimisation in the presence of uncertainty. In *Proceedings of the Congress on Evolutionary Computation 2005, Edinburgh* (pp. 476–483).

Fieldsend, J. E., & Everson, R. M. (2015). The rolling tide evolutionary algorithm: A multiobjective optimizer for noisy optimization problems. *IEEE Transactions on Evolutionary Computation*, *19*, 103–117.

Goh, C. K., & Tan, K. C. (2009). *Evolutionary Multi-objective Optimization in Uncertain Environments: Issues and Algorithms*. Springer-Verlag.

Jin, Y., & Branke, J. (2005). Evolutionary optimization in uncertain environments - a survey. *IEEE Transactions on Evolutionary Computation*, *9*, 303–317.

Lee, L. H., & Chew, E. P. (2005). Design sampling and replication assignment under fixed computing budget. *Journal of Systems Science and Systems Engineering*, *14*, 289–307.

Lee, L. H., Chew, E. P., Teng, S., & Chen, Y. (2008). Multi-objective simulation-based evolutionary algorithm for an aircraft spare parts allocation problem. *European Journal of Operational Research*, *189*, 476–491.

Lee, L. H., Chew, E. P., Teng, S., & Goldsman, D. (2010). Finding the non-dominated pareto set for multi-objective simulation models. *IIE Transactions*, *42*, 656–674.

Miettinen, K. (1998). *Nonlinear Multiobjective Optimization* volume 12 of *International Series in Operations Research & Management Science*. Springer US.

Mohammadi, A., Omidvar, M. N., & Li, X. (2013). A new performance metric for user-preference based multi-objective evolutionary algorithms. In *Proceedings of the Congress on Evolutionary Computation 2013, Cancún* (pp. 2825–2832).

Papadopoulos, C. T., O'Kelly, M. E. J., Vidalis, M. J., & Spinellis, D. (2009). *Analysis and Design of Discrete Part Production Lines* volume 31. Springer Optimization and its Application Series.

Park, T., & Ryu, K. R. (2011). Accumulative sampling for noisy evolutionary multi-objective optimization. In *Proceedings of the Conference on Genetic and Evolutionary Computation 2011, Dublin* (pp. 793–800).

Siegmund, F., Bernedixen, J., Pehrsson, L., Ng, A. H. C., & Deb, K. (2012a). Reference point-based evolutionary multi-objective optimization for industrial systems simulation. In *Proceedings of the Winter Simulation Conference 2012, Berlin*.

Siegmund, F., Ng, A. H. C., & Deb, K. (2012b). Finding a preferred diverse set of pareto-optimal solutions for a limited number of function calls. In *Proceedings of the Congress on Evolutionary Computation 2012, Brisbane* (pp. 2417–2424).

Siegmund, F., Ng, A. H. C., & Deb, K. (2013). A comparative study of dynamic resampling strategies for guided evolutionary multi-objective optimization. In *Proceedings of the Congress on Evolutionary Computation 2013, Cancún* (pp. 1826–1835).

Siegmund, F., Ng, A. H. C., & Deb, K. (2015). Hybrid dynamic resampling for guided evolutionary multi-objective optimization. In *Proceedings of the 8th International Conference on Evolutionary Multi-Criterion Optimization, 2015, Guimarães* (pp. 366–380). volume 9018 of *Lecture Notes in Computer Science*.

Stump, G., Simpson, T. W., Donndelinger, J. A., Lego, S., & Yukish, M. (2009). Visual steering commands for trade space exploration: User-guided sampling with example. *Journal of Computing and Information Science in Engineering*, *3*, 044501:1–10.

Syberfeldt, A., Ng, A. H. C., John, R. I., & Moore, P. (2010). Evolutionary optimisation of noisy multi-objective problems using confidence-based dynamic resampling. *European Journal of Operational Research*, *204*, 533–544.

Tan, K. C., & Goh, C. K. (2008). Handling uncertainties in evolutionary multi-objective optimization. In *Plenary/Invited Lectures of World Congress on Computational Intelligence 2008, Hong Kong* (pp. 262–292). volume 5050 of *Lecture Notes in Computer Science*.

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Journal of Evolutionary Computation*, *8*, 173–195.

Zitzler, E., & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms - A comparative case study. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature, 1998, Amsterdam* (pp. 292–301).