Hybrid Dynamic Resampling Algorithms for Evolutionary Multi-objective Optimization of Invariant-Noise Problems

Florian Siegmund[1], Amos H.C. Ng[1], and Kalyanmoy Deb[2]

[1] School of Engineering Science, University of Skövde, Sweden
{florian.siegmund,amos.ng}@his.se
[2] Department of Electrical and Computer Engineering,
Michigan State University, East Lansing, USA
kdeb@egr.msu.edu

COIN Report Number 2015023

Abstract. In Simulation-based Evolutionary Multi-objective Optimization (EMO) the available time for optimization usually is limited. Since many real-world optimization problems are stochastic models, the optimization algorithm has to employ a noise compensation technique for the objective values. This article analyzes Dynamic Resampling algorithms for handling the objective noise. Dynamic Resampling improves the objective value accuracy by spending more time to evaluate the solutions multiple times, which tightens the optimization time limit even more. This circumstance can be used to design Dynamic Resampling algorithms with a better sampling allocation strategy that uses the time limit. In our previous work, we investigated Time-based Hybrid Resampling algorithms for Preference-based EMO. In this article, we extend our studies to general EMO which aims to find a converged and diverse set of alternative solutions along the whole Pareto-front of the problem. We focus on problems with an invariant noise level, i.e. a flat noise landscape.

Keywords: evolutionary multi-objective optimization, simulation-based optimization, noise, dynamic resampling, budget allocation, hybrid

1 Introduction

Simulation-based optimization of real-world optimization problems is usually run with a limited time budget [13]. The goal of Evolutionary Multi-objective Optimization is to provide the decision maker with a well-converged and diverse set of alternative solutions close to the true Pareto-front. Covering the whole Pareto-front is usually hard to achieve within the available time, dependent on the optimization problem. Besides limited optimization time, Simulation-based Optimization entails the challenge of handling noisy objective functions [2, 4,

22, 11]. To obtain an as realistic as possible simulation of the system behavior, stochastic system 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 certain system configuration, the result value is slightly different for each simulation run.

If an evolutionary optimization algorithm is run without noise handling on a stochastic simulation optimization problem, the performance will degrade in comparison with the case if the true mean objective values would be known. The algorithm will have wrong knowledge about the solutions' quality. Two cases of misjudgment will occur. The algorithm will perceive bad solutions as good and select them into the next generation (Type II error). Good solutions might be assessed as inferior and might be discarded (Type I error). The performance can therefore be improved by increasing the knowledge of the algorithm about the solution quality.

Resampling is a way to reduce the uncertainty about objective values of 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. The additional samples used for Resampling, are not available for the optimization. Therefore, a resampling strategy which samples the solutions carefully according to their resampling need, can help an EMO algorithm to achieve better results than a static resampling allocation. Such a strategy is called Dynamic Resampling, DR. In this paper, we run Dynamic Resampling strategies for Evolutionary Multi-objective Optimization, which were proposed for Preference-based EMO in our previous work [17–19], and which can be used on any EMO algorithm. They are evaluated together with the EMO algorithms Non-domination Sorting Genetic Algorithm-II (NSGA-II) [7], and the Hypervolume Estimation algorithm (HypE) [1].

The resampling need varies between solutions and can be calculated in many different ways [19]. One approach is to assign more samples to solutions close to the Pareto-front. Since in real-world problems, the Pareto-front is not known, this can be achieved approximatively by assigning more samples to solutions as the optimization time progresses. If the Ideal Point can be estimated, fitness-based Dynamic Resampling algorithms can be used, which can assign higher samples to solutions with good objective values [8, 16]. Another approximation strategy is to assign more samples to solutions that dominate more other solutions, or are dominated by fewer other solutions, respectively. This is done by, for example, Confidence-based Dynamic Resampling and the MOPSA-EA algorithm [21], or Rank-based Dynamic Resampling [17–19]. EMO algorithms have a selection operator which determines if solutions will be part of the next population or be discarded. This approach is used by the MOPSA-EA algorithm [21] which compares pairs of solutions using Confidence-based DR (CDR) or the MOCBA and EA approach [5] (Multi-objective Optimal Computing Budget Allocation), which compares sets of solutions. CDR and MOCBA show promising results [5, 16], but they have the disadvantage of limited applicability to only

special types of Evolutionary Algorithms. CDR is limited to steady state EAs and MOCBA requires an EA with high elitism. Other examples for DR algorithms which have limited applicability, due to direct integration with a certain EA, were proposed [10] and [15]. The mentioned algorithms, except for MOPSA-EA/CDR, do not consider the limited sampling budget as in our situation. In this paper, we show how Dynamic Resampling and EMO performance can be improved by considering a fixed time limit for optimization.

The paper is structured as follows. Section 2 provides background information to Dynamic Resampling and an introduction to the NSGA-II and HypE Evolutionary Algorithms. Different Resampling Algorithms for time-constrained optimization are explained in Section 3. Section 4 describes a stochastic production line problem. In Section 5, numerical experiments of Dynamic Resampling algorithms on the two Evolutionary Algorithms are performed and evaluated with different performance metrics for multi-objective optimization. In Section 6, conclusions are drawn and possible future work is pointed out.

2 Background

In this section, background information is provided regarding Resampling as a noise handling method in Evolutionary Multi-objective Optimization. Also, an introduction to the multi-objective optimization algorithms NSGA-II [7] and HypE [1] is given, which are used to evaluate the Dynamic Resampling algorithms in this paper.

2.1 Noise compensation by Sequential Dynamic Resampling

To be able to assess the quality of a solution according to a stochastic evaluation function, statistical measures like 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 and 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.

In many cases, resampling is implemented as a Static Resampling scheme. This means that all solutions are sampled an equal, fixed number of times. The need for resampling, however, is often not homogeneously distributed throughout the search space. Solutions with a smaller variance in their objective values will require less 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, Dynamic Resampling techniques are used, which are described in the following section.

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 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 needed 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, Dynamic Resampling can save sampling budget that can be used to evaluate more solutions and to run more generations of an evolutionary algorithm instead [17].

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 usually is called standard error of the mean. It is calculated as follows [8]:

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

By increasing the number of samples n of $F_i(s)$ the standard deviation of the mean is reduced. However, for the standard error of the mean 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. Yet, the probability of reducing the sample mean by sampling s increases asymptotically as the number of samples is increased.

Sequential Dynamic Resampling An intuitive dynamic resampling procedure would be to reduce the standard error until it drops below a certain user-defined threshold $\mathrm{se}_n(s) := \mathrm{se}_n(\mu_n(F(s))) < \mathrm{se}_{th}$. The required sampling budget for the reduction can be calculated as in Equation 2.

$$n(s) > \left(\frac{\sigma_n(F_i(s))}{\mathrm{se}_{th}}\right)^2. \tag{2}$$

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 for reaching the error threshold, in case the sample mean has 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 was too big. Therefore Dynamic Resampling is often done sequentially. For Sequential Dynamic Resampling often the shorter term Sequential Sampling is used. A Sequential Dynamic Resampling algorithm which uses the $\mathrm{se}_n(s) < \mathrm{se}_{th}$ termination criterion is Standard Error Dynamic Resampling, SEDR proposed in [8].

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 it is checked

Algorithm 1 Basic sequential sampling algorithm pattern

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

if the knowledge about the solution is sufficient. 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 to sample the solution sufficiently often, and can reduce the number of excess samples.

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 $B_F \leq (b_f - 1)|P|$, where $|P|$ is the population size.

2.2 Evolutionary Multi-objective Optimization

In this section, two popular Evolutionary Multi-objective Optimization algorithms are presented. In this article, those algorithms are used for evaluation with Multi-objective Dynamic Resampling algorithms.

NSGA-II algorithm The NSGA-II algorithm was proposed in [7] and is based on the NSGA-II algorithm [7]. NSGA-II is a population-based evolutionary algorithm for optimization problems with multiple objectives. In the selection step NSGA-II sorts the population and offspring into multiple fronts and selects front by front into the next population as long as the fronts can be selected as a whole. If the next front can only be selected partially NSGA-II uses a clustering method called Crowding Distance to determine which solutions of the front shall be selected, in a way that increases population diversity. The parental selection for offspring generation follows this fitness hierarchy. The NSGA-II selection step

is depicted in Figure 1. The same fitness hierarchy, i.e., dominance first, then reference point distance, is used for parental selection.

HypE algorithm The Hypervolume Estimation Algorithm HypE was proposed in [1] and is based on the NSGA-II algorithm [7]. It is designed to optimize the hypervolume of the final population and can achieve higher hypervolume values than NSGA-II [1]. It was chosen for evaluation in this paper, since it is a popular Evolutionary Multi-objective Optimization algorithm its goal is to find a distribution of solution alternatives which maximizes the hypervolume measure. This distribution is different that the NSGA-II distribution, and we hope to see a performance difference even in optimization with noisy objective values.

HypE works according to the same principal structure as NSGA-II. For both parental selection and environmental selection, first a non-domination sorting is performed. In the environmental selection step, HypE selects those fronts into the next population that fit as a whole. If the next front can only be selected partially, instead of using a clustering method like NSGA-II to maintain population diversity, HypE uses a measure called Shared Hypervolume to determine which solutions of the front shall be selected. This measure indicates how much a solution contributes to the hypervolume of the whole population. The HypE selection step is depicted in Figure 1. The same fitness hierarchy, i.e., dominance first, then Shared Hypervolume, is used for parental selection.

The Shared Hypervolume is a better indicator of the hypervolume contribution of a solution marginal hypervolume used in SMS-EMOA [3]. In this paper, only bi-objective optimization problems are run. For a higher number of objectives, HypE uses a hypervolume estimation algorithm based on Importance Sampling for the Shared Hypervolume, hence the name. This makes HypE suitable for Many-objective Optimization.

3 Resampling Algorithms

This section presents the resampling algorithms that are evaluated in this paper.

3.1 Static Resampling

Static Resampling assigns the same fixed number of samples to all solutions involved in an optimization run. It 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 in the beginning of an optimization the benefit of accurate values often does not justify their cost.

3.2 Time-based Dynamic Resampling

Time-based DR [17] is a generally applicable dynamic resampling algorithm which can be used on any optimization algorithm. It assumes that the need

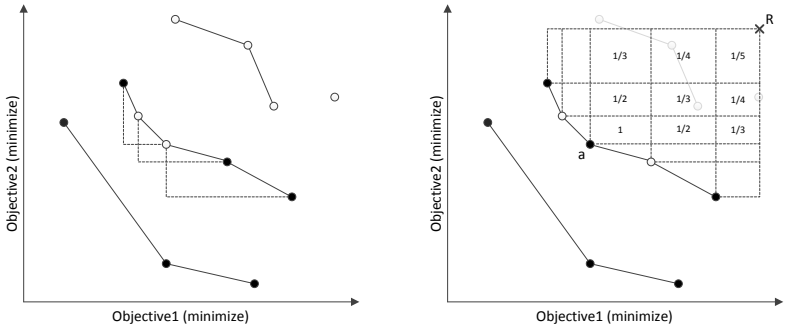

Fig. 1. Environmental selection step. Six out of twelve solutions are selected. The left figure shows the NSGA-II selection result. The complete first front and the extremal solutions from the second front are selected. The one remaining solution is selected due to its highest Crowding Distance value. — The right figure shows the HypE selection result. The complete first front and the solution set of three solutions with the highest overall hypervolume contribution are selected from the second front. The rectangles which are dominated by solution a are marked with a's hypervolume contribution.

for accurate knowledge of objective values increases towards the end of the optimization run. Time-based DR makes only one sampling decision for each solution (one-shot allocation) according to the elapsed time, the current overall number of simulation runs B_t. However, if a solution survives into the next generation of an evolutionary algorithm the decision is made again. The optimization is run until the total number of solution evaluations B is reached, leavning only B_F samples for the final population. The acceleration parameter $a > 0$ allows to speed up or slow down the sampling allocation.

$$x_s^T = \min\left\{1, \frac{B_t}{B - B_F}\right\}^a. \tag{3}$$

3.3 Time-Step Dynamic Resampling

We propose another time-based DR algorithm which uses an allocation function different from Time-based DR. Instead of gradually increasing the allocation budget, Time-Step Dynamic Resampling allocates only b_{min} samples in the beginning of the optimization runtime, like Static Resampling. At a certain time threshold B_{thr}, the allocation is increased to b_{max} within a short period of time. This allocation function cannot be created by changing the acceleration parameter a in Time-based DR. The sudden raise of the allocation function can be done with the step function, as in Equation 4.

$$x_s^{TS} = \begin{cases} 0 & \text{if } B_t < B_{thr}, \\ 1 & \text{otherwise.} \end{cases} \tag{4}$$

Another, smoother, way to rapidly increase the sampling allocation to b_{max} is the Sigmoid function or, more general, the Logistic Growth [23, 17]. The allocation function of Time-Logistic Dynamic Resampling is given in Equation 5, where $\gamma > 0$ is the growth rate, $B_{thr} \in [0, 1]$ the point of highest growth, and $\nu > 0$ determines if the maximum growth occurs close to the lower or the upper asymptote.

$$x_s^{TS} = \frac{1}{\left(1 + e^{-\gamma(B_t - B_{thr})/(B - B_F)}\right)^{\frac{1}{\nu}}}. \tag{5}$$

3.4 Rank-based Dynamic Resampling

Rank-based DR [17] 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 lower Pareto-rank. It can have a parameter allowing only the solutions with a Pareto-rank of n or less to be allocated additional samples (RankMaxN-based DR) [18]. Since after each added sample, the dominance relations in the population of an MOO algorithm changes Rank-based DR is performed sequentially.

The normalized rank-based resampling need of RankMaxN-based Dynamic Resampling, x_s^{Rn}, is calculated as in Equation 6, where $a > 0$ is the acceleration parameter for the increase of the time-based and rank-based sampling need. n is the maximum considered Pareto-rank. Only solutions with Pareto-rank n or less are considered for allocating additional samples. Furthermore, S is the solution set of current population and offspring, R_s is the Pareto-rank of solution s in S, and $R_{max} = \max_{s \in S} R_s$.

$$x_s^{Rn} = 1 - \left(\frac{\min\{n, R_s\} - 1}{\min\{n, R_{max}\} - 1}\right)^a. \tag{6}$$

Rank-Time-based DR Rank-based DR can be used in a hybrid version as Rank-Time-based DR [18]. Thereby, it can use the information given by the limited time budget B. A Rank-Time-based sampling allocation can be achieved by using the minimum or the product of the Rank-based x_s^R or Time-based x_s^T allocation functions (Equation 7).

$$x_s^{RT} = \min\left\{x_s^T, x_s^R\right\}, \quad x_s^{RT} = x_s^T x_s^R. \tag{7}$$

3.5 Domination Strength Dynamic Resampling

We proposed Domination Strength Dynamic Resampling in [17], and present an improved version here. Like for Rank-based DR, we are interested to know the accurate objective values of Pareto-optimal solutions, or solutions which are only dominated by a few other solutions. In Domination Strength DR, a higher sampling budget is added to those solutions, but only if they dominate many other solutions. The idea behind this is that we need to know the accurate objective

values for Pareto-solutions which are surrounded by dominated solutions, in order to make good selection decisions in the Evolutionary Algorithm. In order to reduce the sampling allocation for dominated solutions which themselves dominate many other solutions, we subtract the number of other solutions by which the solution is dominated. The allocation function is given in Equation 8, where $\mathrm{dom}(s, S)$ is the number of solutions in the population which are dominated by solution s, and $D_{max} = \max_{s \in P} \mathrm{dom}(s, P)$. $\mathrm{dom}(P, s)$ is the number of solutions in the population which dominate solution s, or in other words, to which solution s is inferior. The maximum count of dominating solutions is therefore called $Inf_{max} = \max_{s \in P} \mathrm{dom}(P, s)$.

Similar to RankMaxN-based DR we introduce an upper limit n for counting the number of dominated or dominating solutions. Equation 8 defines the allocation function of DSMaxN-based DR.

$$x_s^{DSn} = \max \left\{ 0, \frac{\min\{n, \mathrm{dom}(s, P)\}}{\min\{n, D_{max}\}} - \frac{\min\{n, \mathrm{dom}(P, s)\}}{\min\{n, Inf_{max}\}} \right\}^a . \tag{8}$$

DS-Time-based DR In the same way as for Rank-Time-based DR, for DS-based DR we propose to define a hybrid version which gradually increases the Domination-Strength-based allocation based on the elapsed optimization runtime. Using the information about the time limit, we are able to define hybrid Dynamic Resampling algorithms with better performance. A DS-Time-based sampling allocation can be achieved by using the minimum or the product of the Domination Strength-based x_s^{DS} or Time-based x_s^T allocation functions (Equation 9).

$$x_s^{DST} = \min\left\{x_s^T, x_s^{DS}\right\}, \quad x_s^{DST} = x_s^T x_s^{DS}. \tag{9}$$

4 Stochastic Production Line Model

The benchmark optimization problem used for evaluation is a stochastic simulation model described in [19], consisting of 6 machines (Cycle time 1 min, $\sigma = 1.5$ min, CV = 1.5, lognormal distribution) and 5 buffers with sizes $\in [1, 50]$. The source distribution is lognormal with $\mu = 1$ min and $\sigma = 1.5$ min, if not stated otherwise. The basic structure is depicted in Figure 2. The simulated warm-up time for the production line model is 3 days and 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 [9] (finding the minimal number of buffers required to obtain 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.

In this stochastic production line model the cycle time of machine M4 has a higher standard deviation ($\sigma = 25$ min, CV = 25) than the other machines,

Fig. 2. A simplistic production line configuration.

causing a high standard deviation of the average throughput objective measured during the whole simulation time. The CV of TH becomes 0.2089, i.e. around 20% noise. We call this model PL-NM-20% in the following. 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 [14]. The high variability may be caused by a manual operator which is needed to start an automated process, e.g. after tool changes. Or more commonly, the variability may be cause by a manual quality inspection workstation in which the times for visually checking if there are any defeats in the work-pieces vary significantly.

5 Numerical Experiments

In this section, the described resampling algorithms are evaluated in combination with the NSGA-II and HypE algorithms on benchmark functions and a stochastic production line simulation model.

5.1 Problem settings

Two function characteristics are important for experimentation. The function complexity and the noise level. Therefore, we test the DR algorithms on the ZDT1-5% function, which has low complexity, and the ZDT4-5% function with high complexity. In order to test a high-noise function, we use the stochastic production line model PL-NM-20%. We configure it with CV=25 for machine 4, which gives an output noise of around 20%.

The used benchmark functions ZDT1 and ZDT4 [25] are deterministic in their original versions. Therefore, zero-mean normal noise has been added to create noisy optimization problems. For example, the ZDT4 objective functions are 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 $g(x) = 1 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 - 10\cos(4\pi x_i))$.

The two objectives of the ZDT functions 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 ZDT functions, the relative added noise 5% is $(\mathcal{N}(0, 0.05), \mathcal{N}(0, 5))$ (considering the relevant objective ranges of $[0, 1] \times [0, 100]$). In the following, the ZDT benchmark optimization problems are called ZDT1-5% and ZDT4-5%.

5.2 Algorithm parameters

The limited simulation budget is chosen as $B = 10,000$ replications for both the ZDT problems and the production line problem. NSGA-II and HypE are 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 $\eta_m = 5$.

We set the minimum budget to be allocated to $b_{min} = 1$ and the maximum budget to $b_{max} = 15$ for all Dynamic Resampling algorithms. Static Resampling is run in configurations with fixed number of samples per solution between 1 and 5. Time-based Resampling uses a linear allocation, $a = 1$. The allocation threshold of Time-Step DR is set to $B_{thr} = 0.8$. The allocation function of Time-Step Logistic DR is configured as $x_s^{TS} = \frac{1}{(1+e^{-40(B_t-0.6)/(B-B_F)})^{0.5}}$. 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: $x_s^{RT} = \min\{x_s^T, x_s^R\}$. The same parameters are chosen for Domination Strength Dynamic Resampling and DS-Time-based Dynamic Resampling.

5.3 Performance Measurement

In this paper, the Hypervolume measure HV [26] is used for measuring convergence and diversity of the optimization results.For the HV metric, a Hypervolume reference point HV-R and a Hypervolume base point HV-B for normalization must be specified. For ZDT1-5%, we use $HV\text{-}R$=(1,1) and $HV\text{-}B$=(0,0). For ZDT4-5%, we use $HV\text{-}R$=(1,20) and $HV\text{-}B$=(0,0). The HV performance of the PL-NM-20% model is measured with $HV\text{-}R$=(100,25) and $HV\text{-}B$=(10,45).

In order to get a second measurement value, the Inverse Generational Distance Metric IGD [6] is used to measure convergence and diversity for problems where a reference Pareto-front is available. In order to measure convergence separately, the Generational Distance GD [24] is used, for optimization problems with known true Pareto-front. For measuring the diversity of the population separately we use a new metric which is described in the following section.

Population Diversity (PD) In order to measure the diversity of a population, we use a metric based on the NSGA-II Crowding Distance [7], which we proposed in [19]. It measures the diversity of not only the first front, but also of all other fronts in the population. This is because the diversity of the fronts $2 \dots n$ 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 10.

$$PD\,(P) = \sum_{s \in P} \delta\,(s) \,/\, |P|. \tag{10}$$

The formal definition of $\delta\,(s_k)$ is given in Equation 11, where N_j are neighboring solutions and we write $F_i(s_k) := \mu_n(F_i(s_k))$ to simplify the notation.

$$\delta\,(s_k) = \sum_{i=1}^{H} \begin{cases} |F_i(s_k) - F_i(s_N)| & \text{if } F_i(s_k) \text{ extremal,} \\ |F_i(s_{N1}) - F_i(s_{N2})| & \text{otherwise.} \end{cases} \tag{11}$$

5.4 Results

The measured optimization performance results are shown in Table 1 for the ZDT1-5% benchmark problem, in Table 2 for the ZDT4-5% benchmark problem, and in Table 3 for the stochastic production line problem PL-NM-20%.

Table 1. Performance measurement results on the ZDT1-5% benchmark function for NSGA-II and HypE with different Dynamic Resampling algorithms. The measurement is performed on the last population where $b_f = 25$ final samples have been executed.

	HV	IGD	GD	PD	HV	IGD	GD	PD
	NSGA-II				HypE			
Static1	0.7493	0.3416	**0.0558**	0.0438	0.4030	0.0780	0.0182	0.0570
Static2	0.7141	0.3365	0.1239	0.0233	0.4658	0.1282	0.0078	0.0693
Static3	0.6225	0.3687	0.1191	0.0183	0.5328	0.0489	0.0090	0.0641
Static4	0.6159	0.3294	0.0678	0.0206	0.5306	**0.0369**	0.0104	0.0723
Static5	0.5218	0.3719	0.2711	0.0463	0.5977	0.0560	0.0082	0.0446
Time 1-10	0.7178	0.2928	0.2678	0.0538	0.6278	0.0446	0.0076	0.0538
TS 1-10	0.6854	**0.2755**	0.2279	0.0573	0.4372	0.1108	0.0163	0.0492
TSL 1-10	0.6617	0.3085	0.2162	0.0364	0.5682	0.0513	0.0085	0.0458
Rank 1-10	0.4387	0.4042	0.3887	0.0507	0.5249	0.0644	0.0095	0.0376
RT 1-10	0.7496	0.2861	0.2848	0.0569	0.5927	0.0548	**0.0071**	0.0382
R5 1-10	0.6677	0.3307	0.2737	0.0553	0.5927	0.0834	0.0129	0.0603
R5T 1-10	**0.9100**	0.3135	0.1048	0.0228	0.6738	0.0502	0.0133	0.0610
DS 1-10	0.5108	0.3576	0.3225	0.0345	0.4840	0.1602	0.0109	0.0354
DST 1-10	0.7576	0.2937	0.2507	**0.0593**	0.5437	0.1070	0.0134	0.0416
DS5 1-10	0.7231	0.3305	0.1818	0.0532	0.4585	0.2718	0.0084	0.0247
DS5T 1-10	0.8618	0.3151	0.1973	0.0445	**0.6924**	0.0517	0.0152	**0.0866**

For the ZDT1-5% function, it can be seen that considering the elapsed optimization time in sampling allocation increases the performance of both NSGA-II

and HypE, compared with DR algorithms that do not consider time. The R5T and DS5T DR algorithms, combining dominance and time, show the best performance. The highest diversity could be achieved with the Domination Strength allocation and the HypE algorithm.

Similar observations can be made on more complex ZDT4-5% problem. However, not as high diversity values could be achieved. This is due to the fact that the algorithms are not able to explore the whole Pareto-front within the $B = 10,000$ function evaluations. A final population close to the Ideal Point is found. Rank-based and Domination Strength-based DR both show good results, with an advantage for Domination Strength-based DR.

Table 2. Performance measurement results on the ZDT4-5% benchmark function for NSGA-II and HypE with different Dynamic Resampling algorithms. The measurement is performed on the last population where $b_f = 25$ final samples have been executed.

	HV	IGD	GD	PD	HV	IGD	GD	PD
	NSGA-II				HypE			
Static1	0.5594	0.2825	0.1048	0.0321	0.4304	0.3635	0.0607	0.0177
Static2	0.3014	0.4077	0.1301	0.0398	0.4288	0.4160	0.1598	0.0468
Static3	0.5434	0.3535	0.1075	0.0223	0.2198	0.4255	0.1897	0.0367
Static4	0.0614	0.3596	0.1006	0.0169	0.2433	0.1592	0.0895	0.0485
Static5	0.0000	0.4360	0.1244	0.0105	0.0000	0.2451	0.2649	**0.0518**
Time 1-10	0.0720	**0.1573**	0.0903	0.0549	0.2016	0.2458	0.1245	0.0282
TS 1-10	0.6331	0.2406	0.0816	0.0422	0.6981	0.3978	**0.0391**	0.0087
TSL 1-10	0.2359	0.2248	0.0964	0.0373	0.2963	0.3956	0.1373	0.0436
Rank 1-10	0.0559	0.1879	0.1677	0.0467	0.0000	0.3628	0.1512	0.0270
RT 1-10	0.4020	0.3741	0.1158	0.0249	0.0000	0.4069	0.0958	0.0095
R5 1-10	0.1043	0.4240	0.1209	0.0187	0.0000	0.4259	0.1332	0.0430
R5T 1-10	0.5297	0.4057	0.0961	0.0196	0.3863	**0.1284**	0.0634	0.0275
DS 1-10	0.4754	0.3450	0.1495	0.0332	0.0567	0.3775	0.1512	0.0225
DST 1-10	0.0022	0.3929	0.1294	0.0236	0.4546	0.3876	0.1157	0.0221
DS5 1-10	0.2885	0.4135	**0.0625**	**0.0080**	0.0644	0.4140	0.1367	0.0263
DS5T 1-10	**0.6530**	0.3802	0.0648	0.0164	**0.6983**	0.4007	0.0498	0.0153

On the stochastic production line model PL-NM-20%, NSGA-II and HypE show a similar performance. Time-based DR algorithms show the best results, whereof Time-based Logistic DR, with an increase of samples around $B_t = 0.6$ shows the best results. The reason for the superiority of time-based Dynamic Resampling is shown in the #Sol. column in Table 3. Time-based DR algorithms save function evaluations in the beginning of the optimization runtime, using less function evaluations for resampling. These evaluations can be used to explore the objective space and Pareto-front instead. It can be observed that

Domination-Strength-based DR algorithms allocate less samples and have more unique solution evaluations available for objective space exploration.

Table 3. Performance measurement results on the production line model PL-NM-20% for NSGA-II and HypE with different Dynamic Resampling algorithms. The measurement is performed on the last population where $b_f = 25$ final samples have been executed. The #Sol. column shows how many unique solutions have been simulated.

	HV	PD	#Sol.	HV	PD	#Sol.
	NSGA-II			HypE		
Static1	0.4899	0.0501	8800	0.4728	0.0505	8800
Static2	0.4860	0.0512	4400	0.4888	0.0353	4400
Static3	0.4667	0.0581	2900	0.4638	0.0485	2900
Static4	0.4614	0.0606	2200	0.4661	**0.0599**	2200
Static5	0.4390	0.0650	1750	0.4735	0.0373	1750
Time 1-10	0.4910	0.0552	2400	0.4602	0.0423	2400
TS 1-10	0.4688	0.0554	6800	0.4891	0.0330	6800
TSL 1-10	**0.5079**	0.0563	3850	**0.5152**	0.0334	3850
Rank 1-10	0.4355	0.0530	1150	0.4062	0.0359	1100
RT 1-10	0.4739	**0.0718**	2500	0.4674	0.0337	2550
R5 1-10	0.4891	0.0480	1900	0.4490	0.0367	1950
R5T 1-10	0.4995	0.0368	3650	0.4851	0.0479	3600
DS 1-10	0.4149	0.0642	2650	0.5110	0.0451	2400
DST 1-10	0.4883	0.0613	4800	0.4744	0.0460	4950
DS5 1-10	0.4824	0.0557	2050	0.4860	0.0445	2100
DS5T 1-10	0.5067	0.0594	4450	0.4817	0.0417	4600

6 Conclusions and future Work

This paper has provided a number of existing and possible methodologies of handling noise in multi-objective optimization algorithms. Different possibilities from Static to several Dynamic Resampling strategies have been discussed. Two stochastic, numerical two-objective problems and an uncertain production line optimization problem have been chosen to compare different sampling methodologies. As a conclusion, we find that in EMO with limited time budget, Hybrid Time-based Dynamic Resampling algorithms are superior to DR algorithms which do not consider the elapsed optimization runtime. The proposed Domination Strength Dynamic Resampling variants show comparable results to Pareto-Rank-based Dynamic Resampling algorithms.

Future work In this paper, we focused on invariant-noise problems. In a future study, we will evaluate Dynamic Resampling for general EMO with limited budget on optimization problems with variable output noise. This study will compare Multi-objective Standard Error Dynamic Resampling [20], which allocates samples until a certain objective accuracy is reached, with other Multi-objective Dynamic Resampling algorithms, like the MOCBA algorithm [5] or Confidence-based Dynamic Resampling [21], on optimization problems with different noise landscapes.

Acknowledgments This study was partially funded by the Knowledge Foundation, Sweden, through the BlixtSim and IDSS projects. The authors gratefully acknowledge their provision of research funding.

References

1. Bader, J. and Zitzler, E.: HypE: An algorithm for fast hypervolume-based many-objective optimization. Computer Engineering and Networks Laboratory (TIK), ETH Zurich (2008)
2. Bartz-Beielstein, T., Blum, D., and Branke, J.: Particle swarm optimization and sequential sampling in noisy environments. Metaheuristics – Progress in Complex Systems Optimization, 261–273 (2007)
3. Beume, N. and Naujoks, B. and Emmerich, M.: SMS-EMOA: Multiobjective Selection Based on Dominated Hypervolume. European Journal of Operational Research, vol. 181, no. 3, pp. 1653–1669 , ISSN 1049-3301 (2007)
4. Branke, J., and Schmidt, C.: Sequential Sampling in Noisy Environments. Proceedings of the Parallel Problem Solving from Nature VIII conference - LNCS, vol. 3242, 202–211 (2004)
5. Chen, C.-H. and Lee, L. H.: Stochastic Simulation Optimization - An Optimal Computing Budget Allocation. World Scientific Publishing Company, ISBN 978-981-4282-64-2 (2010)
6. Coello Coello, C. A. and Reyes Sierra, M.: A Study of the Parallelization of a Coevolutionary Multi-Objective Evolutionary Algorithm. Proceedings of the 3rd Mexican International Conference on Artificial Intelligence, 2004, Mexico City, pp. 688–697, ISBN 3-540-21459-3 (2004)
7. Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, 182–197 (2002)
8. Di Pietro, A., While, L., and Barone, L.: Applying evolutionary algorithms to problems with noisy, time-consuming fitness functions. Proceedings of the Congress on Evolutionary Computation 2004, vol. 2, 1254–1261 (2004)
9. Enginarlar, E. and Li, J. and Meerkov, S. M.: How lean can lean buffers be?. IIE Transactions, vol. 37, no. 5, pp. 333–342 (2005)
10. Fieldsend, J. E. and Everson, R. M.: The Rolling Tide Evolutionary Algorithm: A Multiobjective Optimizer for Noisy Optimization Problems. IEEE Transactions on Evolutionary Computation, vol. 19, no. 1, pp. 103–117, ISSN 1089-778X (2015)
11. Goh, C. K. and Tan, K. C.: Evolutionary Multi-objective Optimization in Uncertain Environments: Issues and Algorithms. Springer-Verlag, ISBN 978-3-540-95976-2 (2009)

12. Jin, Y., and Branke, J.: Evolutionary optimization in uncertain environments - a survey. IEEE Transactions on Evolutionary Computation, vol. 9, no. 3, pp. 303–317, ISSN 1089-778X (2005)
13. Lee, L. H. and Chew, E. P.: Design sampling and replication assignment under fixed computing budget. Journal of Systems Science and Systems Engineering, Systems Engineering Society of China, vol. 14, no. 3, pp. 289–307, ISSN 1004–3756 (2005)
14. Papadopoulos, C. T. and O'Kelly, M. E. J. and Vidalis, M. J. and Spinellis, D.: Analysis and Design of Discrete Part Production Lines. Springer Optimization and its Application Series, vol. 31, ISBN 978-1-4419-2797-2 (2009)
15. Park, T. and Ryu, K. R.: Accumulative Sampling for Noisy Evolutionary Multi-objective Optimization. Proceedings of the Conference on Genetic and Evolutionary Computation, Dublin, Ireland, pp. 793–800, ISBN 978-1-4503-0557-0 (2011)
16. Siegmund, F.: Sequential Sampling in Noisy Multi-Objective Evolutionary Optimization. Master thesis, University of Skövde, Sweden and Karlsruhe Institute of Technology, Germany (2009). Available at http://urn.kb.se/resolve?urn=urn:nbn:se:his:diva-3390
17. Siegmund, F., Ng, A. H.C., and Deb, K.: A Comparative Study of Dynamic Resampling Strategies for Guided Evolutionary Multi-Objective Optimization. Proceedings of the IEEE Congress on Evolutionary Computation 2013, Cancún, Mexico, pp. 1826–1835, ISBN 978-1-4799-0454-9 (2013)
18. Siegmund, F., Ng, A. H.C., and Deb, K.: Hybrid Dynamic Resampling for Guided Evolutionary Multi-Objective Optimization. Proceedings of the 8th International Conference on Evolutionary Multi-Criterion Optimization, Guimarães, Portugal, pp. 366–380, ISBN 978-3-319-15934-8 (2015)
19. Siegmund, F., Ng, A. H.C., and Deb, K.: Dynamic Resampling for Preference-based Evolutionary Multi-Objective Optimization of Stochastic Systems. Submitted to European Journal of Operational Research (2016). Available at http://www.egr.msu.edu/ kdeb/papers/c2015020.pdf
20. Siegmund, F., Ng, A. H.C., and Deb, K.: Standard Error Dynamic Resampling for Preference-based Evolutionary Multi-objective Optimization. Submitted to Computers & Operations Research (2016). Available at http://www.egr.msu.edu/ kdeb/papers/c2015021.pdf
21. Syberfeldt, A. and Ng, A. H. C. and John, R. I. and Moore, P.: Evolutionary optimisation of noisy multi-objective problems using confidence-based dynamic resampling. European Journal of Operational Research, vol. 204, no. 3, 533–544, ISSN 0377-2217 (2010)
22. Tan, K. C. and Goh, C. K.: Handling Uncertainties in Evolutionary Multi-Objective Optimization. Proceedings of the World Congress on Computational Intelligence 2008, Hong Kong, China, pp. 262–292, ISBN 978-3-319-15934-8 (2008)
23. Tsoularis, A.: Analysis of Logistic Growth Models. Research Letters in the Information and Mathematical Sciences, vol. 2, 23–46, ISSN 0377-2217 (2001)
24. Van Veldhuizen, D. A.: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. PhD-thesis, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, USA (1999)
25. Zitzler, E., Deb, K., and Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation, vol. 8, no. 2, 173–195 (2000)
26. Zitzler, E., and Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. Proceedings of the Parallel Problem Solving from Nature V conference - LNCS, 292–301 (1998)