Evolutionary optimisation of noisy multi-objective problems using confidence-based dynamic resampling

Anna Syberfeldt, Amos Ng, Robert I. John, Philip Moore

PII: S0377-2217(09)00853-4
DOI: 10.1016/j.ejor.2009.11.003
Reference: EOR 9802

To appear in: *European Journal of Operational Research*

Received Date: 28 August 2008
Revised Date: 28 October 2009
Accepted Date: 3 November 2009

Please cite this article as: Syberfeldt, A., Ng, A., John, R.I., Moore, P., Evolutionary optimisation of noisy multi-objective problems using confidence-based dynamic resampling, *European Journal of Operational Research* (2009), doi: 10.1016/j.ejor.2009.11.003

Evolutionary optimisation of noisy multi-objective problems using confidence-based dynamic resampling

Anna Syberfeldt*[a], Amos Ng[a], Robert I. John[b], Philip Moore[c]

October 27, 2009

[a]Center for Intelligent Automation, University of Skövde, SE-54148 Skövde, Sweden
[b]Center for Computational Intelligence, De Montfort University, Leicester LE1 9BH, United Kingdom
[c]Center for Mechatronic Research, De Montfort University, Leicester LE1 9BH, United Kingdom

Abstract

Many real-world optimisation problems approached by evolutionary algorithms are subject to noise. When noise is present, the evolutionary selection process may become unstable and the convergence of the optimisation adversely affected. In this paper, we present a new technique that efficiently deals with noise in multi-objective optimisation. This technique aims at preventing the propagation of inferior solutions in the evolutionary selection due to noisy objective values. This is done by using an iterative resampling procedure that reduces the noise until the likelihood of selecting the correct solution reaches a given confidence level. To achieve an efficient utilisation of resources, the number of samples used per solution varies based on the amount of noise in the present area of the search space. The proposed algorithm is evaluated on the ZDT benchmark problems and two complex real-world problems of manufacturing optimisation. The first real-world problem concerns the optimisation of engine component manufacturing in aviation industry, while the second real-world problem concerns the optimisation of a camshaft machining line in automotive industry. The results from the optimisations indicate that the proposed technique is successful in reducing noise, and it competes successfully with other noise handling techniques.

Keywords: Evolutionary computations; Multi-objective optimisation; Noise; Simulation

1 Introduction

Real-world optimisation problems often contain non-linearities, combinatorial relationships, and uncertainty factors that are too complex to be effectively modelled analytically. For these problems, simulation-based optimisation is a powerful technique for determining optimal system parameters (April et al., 2004). Simulation-based optimisation is the process of finding the best parameter values for a system, in which the performance is evaluated on the basis of output from a simulated model of the system.

While traditional, analytical optimisation methods have been unable to cope with the challenges imposed by many simulation-based optimisation problems in an efficient way, such as multimodality,

*Corresponding author. Tel.: +46 701 408051; fax: +46 500 448598. E-mail address: anna.syberfeldt@his.se (A. Syberfeldt).

non-separability and high dimensionality, evolutionary algorithms have been shown to be applicable to this type of problem (Ong et al., 2004). Evolutionary algorithms (EAs) are also particularly suitable for solving problems that require simultaneous optimisation of more than one objective, which is often the case in real-world applications (Deb, 2004). The difficulty with multi-objective problems is that usually there is no single optimal solution with respect to all objectives, as improving the performance of one objective means decreasing the performance of another (Srinivas and Deb, 1995). Instead of a single optimum, there is a set of optimal trade-offs between the conflicting objectives, called Pareto-optimal solutions (or Pareto front, if plotted in criterion space) (Coello Coello et al., 2007). Different Pareto ranks can also be identified among solutions. Rank 1 includes the Pareto-optimal solutions in the complete population, and rank 2 the Pareto-optimal solutions identified when temporarily discarding all solutions of rank 1, and so on. Contrary to many other optimisation techniques, EAs can capture multiple trade-off solutions in one single optimisation run since they maintain a population of solutions.

In many practical applications of EAs, one has not only to cope with multiple optimisation objectives, but also with stochastic noise as a consequence of uncontrollable variations, caused, for example, by human operators or worn-out machines (Jin and Branke, 2005; Bui et al., 2005; Branke et al., 2007; Goh and Tan, 2007). Noise means that even if the initial conditions of the system and its input parameters are known, the output of the system cannot be predicted but will vary from time to time. These unpredictable variations in the simulation output are harmful to the optimisation process since the EA can be misdirected to propagate inferior solutions. To deal with noise in evolutionary optimisation, three basic approaches have been proposed: explicit averaging, implicit averaging, and selection modification (Jin and Branke, 2005). In explicit averaging, the same solution is simulated a number of times and the objective values are averaged. Simulating a solution n times reduces the noise by a factor of $\sqrt{n}$, but at the same time increases the computational effort by a factor of n (Jin and Branke, 2005). In implicit averaging, the sample size is adjusted to the population size; the larger the population the smaller the sample size (Fitzpatric and Grefenstette, 1988; Miller and Goldberg, 1996). The assumption of this approach is that there are many similar solutions in a large population, and that the influence of noise is compensated for as the algorithm revisits promising regions of the search space frequently. In selection modification, the ranking and/or selection procedure is modified to compensate for noise, such that a solution is only considered better than another solution if certain conditions are satisfied (Jin and Branke, 2005). For example, the probability of dominance can be considered as proposed by Hughes (2001), or the closeness of solutions can be considered as proposed by Babbar et al. (2003).

In this paper we present a new noise handling technique for multi-objective optimisation that efficiently deals with noise in simulations. This technique varies the number of samples used per solution based on the amount of noise in combination with a user-defined confidence level, controlling the trade-off between search space exploration and number of solution samplings. With the technique, resampling of solutions is performed iteratively until the noise is sufficiently reduced. The proposed technique is integrated in an algorithm called "Multi-Objective Parallel Surrogate-Assisted EA" (MOPSA-EA) described in Syberfeldt et al. (2008). MOPSA-EA is designed to reduce the large time consumption associated with many real-world problems approached by simulation. The algorithm

supports a high degree of parallelism by implementing the asynchronous master-slave parallelisation model in combination with a steady-state design. For improved efficiency, the algorithm also uses a simulation surrogate to screen candidate solutions and identify the most promising ones. A simulation surrogate is a computationally cheap approximation of a time-consuming simulation that can be used to estimate the objective values of solutions.

The rest of the paper is organised as follows. The next section discusses the effects of noise in EAs and outlines different techniques that have been suggested to cope with this problem in multi-objective optimisation. In Section 3, the new noise handling technique proposed in this paper is described. In evaluating the new technique, it is applied to five benchmark problems and two real-world problems of manufacturing optimisation, which are described in Section 4. The technique is implemented in MOPSA-EA, which is presented in Section 5. The results of the evaluation are presented in Section 6, along with a comparison of other noise handling techniques and an analysis of the results. Conclusions of the study and future work are presented in Section 7.

2 Noisy optimisation problems

A certain degree of randomness, so called noise, is an inherent property of most real-world systems. When a system is subject to noise, repeated evaluations of the same solution over time will result in different objective values. This effect is exemplified in Figure 1 (adopted from Büche et al., 2002), where the objective value returned from the evaluation function f has an error that is governed by a normal distribution and therefore varies from time to time.

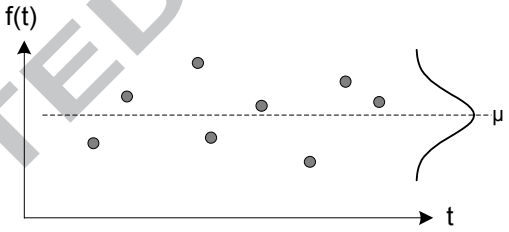

Figure 1: Varying fitness values due to noise.

A noisy evaluation function is also illustrated in Figure 2 (adopted from Di Pietro et al., 2004). In this figure, the function to be optimised is shown without noise to the left and with added noise to the right (the probability of a function evaluation resulting in a particular value is represented by the shading; the darker the area the more likely the value occurs). While it is trivial to optimise the original function by an EA, the problem becomes significantly harder to solve when noise is present.

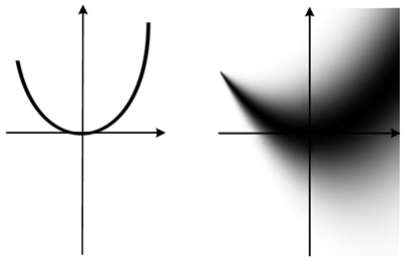

Figure 2: Function without noise (left) and with noise (right).

The problem of noise is that it is not possible to say for certain which of two solutions is the better one (Tan and Goh, 2008). This influences the evolutionary selection negatively in two aspects:

(i) an inferior solution may be erroneously believed to be superior and therefore survives and is given the opportunity to reproduce, or

(ii) a superior solution may be erroneously believed to be inferior and is therefore eliminated.

Noisy objective values like these are likely to cause a reduced convergence rate of the optimisation and a deterioration of the quality of the final sub-optimum (Beyer, 2000; Arnold and Beyer, 2002; Branke and Schmidt, 2003; Jin and Branke, 2005), since the evolutionary process, more or less, degenerates into a random search (Tan and Goh, 2008). A number of different techniques have been suggested to handle this problem, both for single-objective and multi-objective problems. Although many techniques for single-objective problems can be applied to multi-objective problems as well, this paper focuses on techniques specifically designed for multi-objective optimisation. There are four major such approaches; static resampling, modified Pareto ranking scheme, dominance-dependent lifetime, and fitness inheritance. A description of these approaches is provided in the remainder of this section. For an overview of noise handling techniques in single-objective optimisation the reader is referred to Jin and Branke (2005).

2.1 Static resampling

Static resampling, that is, sampling the objective values of all solutions a fixed number of times and using the average values, is the most commonly used method for handling noise. The reduction of variance in the estimated objective is proportional to the sample size; sampling an objective value n times reduces the standard deviation of the objective by a factor of $\sqrt{n}$, but at the same time increases the computational effort by a factor of n (Jin and Branke, 2005).

2.2 Modified Pareto ranking scheme

Two different approaches of modifying the original Pareto ranking scheme for handling noise have been suggested; one probability-based and one based on a clustering method.

Probability-based With the probability-based Pareto ranking scheme, the original Pareto ranking scheme is replaced by a probabilistic ranking process that takes noise into consideration (Hughes, 2001; Teich, 2001). In this ranking process, a solution s is assigned a rank representing the sum of probabilities that each of the solutions in the population dominates s (the lower the rank, the better the solution). In assigning ranks, the probability of making a wrong selection among two solutions is quantified. When considering only one objective of two solutions, estimated with value a and value b, respectively, the probability of making a wrong decision with respect to this objective is calculated according to Equation 1 (assuming a minimisation problem).

$$P(a > b) = \int_{-\infty}^{\infty} \left(pdf_a(a-x) \int_{x}^{-\infty} pdf_b(b-x)\, dx \right) \tag{1}$$

In this Equation 1, x is the point being integrated over, and pdf is a probability distribution function defined by the mean and the standard deviation of the objective value. The formula estimates the probability of b being in any position left of x, when a is located at point x. In a multi-objective minimisation problem of M objectives, the probability of making a wrong decision when choosing between two solutions A and B is calculated according to Equation 2

$$P(A > B) = \prod_{i=0}^{M} P(A_i > B_i), \tag{2}$$

and the probability of A and B being mutually non-dominating is calculated according to Equation 3

$$P(A \equiv B) = 1 - P(A < B) - P(A > B). \tag{3}$$

Based on these formulae, the probabilistic rank R of solution p with index i is then calculated according to Equation 4

$$R_i = \sum_{j=1}^{N} P\left(p_j > p_i\right) + \frac{1}{2} \sum_{j=1}^{N} P\left(p_j \equiv p_i\right) - 0.5. \tag{4}$$

With probabilistic ranks, most solutions will be assigned unique ranks as a consequence of the ranks being real values. A potential problem with this is that crowding distance among solutions (that is, the density of solutions in a particular area) is not be considered, resulting in a poor diversity of the population. Another drawback of the probability-based technique is the computational expense from performing integral operations for each objective in each solution (cf. Equation 1).

A variant of the probability-based Pareto ranking scheme is the technique of multi-objective computing budget allocation (Lee et al., 2008). In this variant, the solutions are first sorted in descending order according to the R values and then the n solutions with the best ranks are included in the Pareto front (where n is user-defined). Two types of error probabilities are calculated; (*i*) the probability that at least one solution in the non-Pareto set (that is, solutions that do not belong to rank 1) is non-dominated, and (*ii*) the probability that at least one solution in rank 1 is dominated. As long as these probabilities are above a user-defined threshold, resampling of solutions continues. When both errors have decreased below the given threshold, the algorithm proceeds to generate the next generation of the population.

Cluster-based The cluster-based Pareto ranking scheme is also based on a modified ranking procedure (Babbar et al., 2003). In this approach, a cluster-based Pareto front is formed by solutions of rank 1 and solutions that lie in the neighbourhood of rank 1. Two solutions A and B are considered to be neighbours if their difference in mean values in the i:th objective is less then $K\sqrt{\frac{\sigma_{iA}+\sigma_{iB}}{2}}$ (where i is user-defined, K is a user-defined neighbourhood restriction factor, and σ_{ij} is the standard deviation in the i:th objective of solution j). At the same time, the difference in any objective $m \neq i$ must be less then $\sqrt{\frac{\sigma_{mA}+\sigma_{mB}}{2}}$. Initially, the value of K is large and a large number of dominated solutions are included in the cluster-based Pareto front. During each generation of the EA, all solutions in the population are resampled n times (where n is user-defined). With a decreased standard deviation resulting from the resamplings, the K value is decreased during the optimisation. A smaller K value makes it harder for solutions to be included in the front, and thereby the front becomes increasingly reliable.

A potential drawback of the cluster-based Pareto ranking scheme is that the diversity of the population might be impeded with the modified ranking scheme. The clustering effect arising from including solutions close to solutions of rank 1 in the front results in many similar solutions. A consequence of many solutions being similar to each other is that the diversity of the population becomes poor and hence the convergence of the search is negatively affected.

2.3 Domination-dependent lifetime

In the technique of dominance-dependent lifetime, each solution is assigned a maximal lifetime based on the number of solutions it dominates (Büche et al., 2002). A solution dominating a large number of solutions is assigned a short lifetime, and vice verse. The purpose of this strategy is to reduce the impact of solutions that appear to be good, but whose fitness value is misleading due to noise. To enable good solutions to proceed in the evolutionary process, non-dominating solutions whose lifetimes have expired are resampled and added to the population with the new objective values.

Contrary to the other techniques, the technique of dominance-dependent lifetime does not explicitly resample solutions, but instead evaluates non-dominating solutions whose lifetime has expired anew and adds them to the population with their new objective values. Basically, this means that a noisy sample of a solution is simply replaced by another noisy sample. It can be argued that the noise is actually not reduced and the original problem that the algorithm may be mislead by the noise is likely to remain.

2.4 Fitness inheritance

A technique to handle noise based on the concept of fitness inheritance has also been suggested (Bui et al., 2005). In this approach, a child inherits the mean objective value $\bar{\mu} = \frac{\mu_{parent1}+\mu_{parent2}}{2}$ and the mean standard deviation $\bar{\sigma} = \frac{\sigma_{parent1}+\sigma_{parent2}}{2}$ from its parents. The child is evaluated once, and if the objective values fall within the confidence interval $\left(\bar{\mu} - 3 * \bar{\sigma} \leq f \leq \bar{\mu} + 3 * \bar{\sigma}\right)$ the inherited fitness is accepted. Otherwise, the child is resampled a user-defined number of times and assigned the mean value and standard deviation of these evaluations.

The technique of fitness inheritance uses a strategy in which the greater the standard deviation of a solution's parents (that is, the larger their noise size), the greater the probability that the inherited

values are accepted without resampling. Intuitively, the inverse is more logical, that is, the larger the noise the greater the need to perform resampling to reduce the noise.

In the next section, a new noise handling technique for multi-objective problems is presented that aims at addressing the drawbacks of the existing noise handling techniques.

3 Confidence-based dynamic resampling

When the optimisation problem is subject to noise, this must be compensated for by performing multiple samplings (that is, simulations) of solutions, otherwise the evolutionary selection process may become unstable. Given s samplings of each solution and a total of n simulations, n/s unique solutions can be evaluated. With 500 simulations, for example, 500 different solutions can be evaluated if each of them is sampled once, and 50 if they are sampled ten times. A crucial aspect here is to find the best trade-off between the number of unique solutions evaluated and the number of samplings of each solution. The larger the number of unique solutions evaluated, the more the search space can be explored and the greater the probability of finding its optimum. However, at the same time, resamplings of solutions is necessary in order to prevent the search from being misdirected due to noise. The remainder of this section includes a description of a novel technique that efficiently addresses this trade-off problem. This technique varies the number of samples used per solution, based on the amount of noise in combination with a user-defined confidence level, controlling the trade-off between search space exploration and number of solution samplings. With this technique, resampling of solutions is performed iteratively until the noise is sufficiently reduced.

3.1 Basic procedure

The procedure of the proposed technique, referred to as confidence-based dynamic resampling (CDR), comprises five main steps which are presented below. A flowchart illustrating the overall procedure can be found on-line alongside the electronic version of the paper (Figure A).

Step 1: Initial sampling

Initially, the two solutions being compared are sampled n times each to form an initial estimate of the amount of noise[1]. In order to avoid spending expensive simulations on inferior solutions, the default value of n is two. In very noisy problems, however, it might be necessary to increase the value of n.

Step 2: Calculation of mean and sample standard deviation

Based on the collected samples, the mean and sample standard deviation of each objective is calculated for the two solutions.

[1] It can be noted that this step is passed over if the solutions for some reason already have been sampled n times.

Step 3: Selection of confidence level

The confidence level is defined by the user and represents the desired certainty of the true relation between two solutions. Three types of true relations are possible between two solutions A and B:

(i) A dominates B, or

(ii) B dominates A, or

(iii) A and B are mutually non-dominating.

The confidence level α is in the interval [0,1] and specifies that in at least α of the cases the selection between two solutions should be correct (that is, the selection should result in the same solution as if the noise would have been completely reduced). It is natural to think that the user would always want 100% of the selections to be correct, that is, set α to 1. However, to reach a high α, a large number of solution samplings are necessary and not as many unique solutions can be evaluated. With a low α, on the other hand, many unique solutions can be evaluated, but there is a risk that the noise will not be sufficiently reduced.

A specific confidence level is defined by the user for each Pareto rank, and generally a higher rank implies a higher confidence level since high precision is usually more important for solutions in, or nearby, the Pareto front. Usually, for rank 1, an α value of about 0.75 presents an acceptable trade-off between search space exploration and noise reduction (for each of the succeeding ranks, the value can the be decreased by 0.05). The confidence level to use in a comparison of two solutions is derived from the solution of highest rank, which means that a non-dominating sort must first be performed to establish the ranks of the solutions. Although this sort may not return the true ranks – the goal of the procedure itself is to find out this ranking – it gives an indication of the relations between solutions in the population.

Step 4: Confidence test

In a noisy context, the true relation between two solutions can only be determined by taking the mean of all possible samples of the solutions. In practice, it is not possible to collect the complete set of samples, only a limited number of samplings can be performed. Instead, the probability that the computed relation is the same when given the collected (observed) samples as given all samples has to be established (that is, the probability of making a correct selection from the solutions). The method of Welch confidence interval is used to do this. Welch confidence interval (WCI) can be used to compare whether or not there is a significant difference between two samples of unknown and possibly unequal variances with respect to a given confidence level. There are two assumptions behind the method of WCI. The first assumption is that the two samples being compared must be independent (that is, the samples must be collected independently of each other). The second assumption is that the samples are drawn from normal distributed populations. Many measurements can be approximated, to varying degrees, by the normal distribution. Also, even though the data are non-normal, the method can still be used to approximate whether or not there is a significant difference between two solutions.

In CDR, WCI values are calculated for each objective i according to Equation 5 (Law and Kelton, 2000)

$$\mu_{iA}(N_A) - \mu_{iB}(N_B) \pm t_{\widehat{f},1-\frac{\alpha}{2M}} \sqrt{\frac{s_{iA}^2}{N_A} + \frac{s_{iB}^2}{N_B}}, \tag{5}$$

where A and B are the two solutions being compared, μ_i is the mean of objective i, N_p is the number of samples of solution p, t is a Student t-distribution with estimated degree of freedom $\widehat{f}$

$$\widehat{f} = \frac{\left[\frac{s_A^2}{N_A} + \frac{s_B^2}{N_B}\right]}{\frac{\left[\frac{s_A^2}{N_A}\right]^2}{N_A - 1} + \frac{\left[\frac{s_B^2}{N_B}\right]^2}{N_B - 1}} \tag{6}$$

and probability $1 - \frac{\alpha}{2M}$ (α=confidence level, M=number of objectives), and s_i^2 is the variance of objective i. In multi-objective problems, the confidence level α has to be divided by $2M$, and not by 1 as in a single-objective problem, due to the Bonferroni Inequality (Law and Kelton, 2000). This means that for a problem of two objectives, to obtain a 0.95 confidence, for example, each objective has to be compared with a confidence level of 0.975.

If the WCI resulting from Equation 5 does not include 0, there is a significant difference between the two solutions in the i:th objective. If none of the WCIs for the M objectives include 0, it means that the relation between the two solutions (see Step 3) can be established with respect to the given confidence level. The dominating solution is then returned, or the one with largest crowding distance, if the solutions are mutually non-dominating, and the procedure is terminated. Otherwise (that is, if any of the objective's intervals covers 0) the difference between the solutions is not significant and further noise reduction is necessary in order to determine their internal relation.

Step 5: Noise reduction by resampling

Ultimately, the relation between the solutions should be established using as few resamplings as possible to save simulation resources. Therefore, the strategy adopted in this step is to resample only one of the solutions at a time. The solution having the largest sample standard deviation in the objective with the largest interval including 0 (that is, with the largest potential to eliminate the undesired insignificance) is the one resampled. After a new sampling of this solution, the procedure is repeated from step 2 and a new check is made if further resampling is necessary.

To prevent the resampling of two solutions that are close to each other in objective space from continuing forever, the number of samplings of a solution is limited. Similar to the specification of confidence level, the maximum number of samplings is defined by the user for each rank. A larger number of samplings (about 5-10) is usually allowed for solutions in higher ranks where a higher precision is needed. A limited number of samplings means that if a solution in this step has already reached its allowed number, it cannot be further sampled. The other solution is then resampled instead, unless it has also reached its maximum number. In such a case the dominating solution (or the one with largest crowding distance, if the solutions are mutually non-dominating) given the obtained samples is returned and the procedure is terminated. It should be noted that if too few resamplings are allowed,

there is a risk that the desired confidence level will not be reached.

Since the ranks of solutions are calculated in every iteration of the procedure, the maximum number of samplings of solutions may change between iterations if their ranks change (as may also the confidence level to use when comparing them). In this way, the resampling strategy becomes dynamic and adjusts automatically to changes in the population.

3.2 Optimised implementation

In step 3, a non-dominating sort is performed to derive the ranks of the solutions. This is a relatively expensive operation with a complexity of $O(MN^2)$ (where M is the number of objectives and N is the number of solutions in the population)[2]. To improve efficiency, step 3 can be optimised to avoid the non-dominating sort whenever possible. This is done by first checking if there is a significant difference between the two solutions in all objectives with respect to the confidence level of rank 1 (that is, no WCI includes 0). If this is the case, there will be a significant difference between them with respect to all other confidence levels also, and their ranks do not need to be established, hence a non-dominating sort can be avoided.

Another potential optimisation of the implementation of the noise handling technique is to perform less frequent resamplings in the beginning of the search during the rough exploration, and increase the resampling frequency when the optimisation begins to converge. For example, the resampling frequency can be increased when the population has not changed in the last i iterations, or when the difference in objective values of solutions is smaller than a parameter p.

3.3 Discussion

The benefits of a dynamic resampling strategy have previously been discussed by Di Pietro et al. (2004). Di Pietro et al. suggest two dynamic strategies to reduce noise. In the first strategy, called standard error dynamic resampling, all solutions are resampled until the standard error of the mean for each solution is below a user-defined threshold (the same threshold is used for all solutions). In the second strategy, called m-level resampling, different thresholds are used for different noise intervals. Although Di Pietro et al. were able to demonstrate the potential of a dynamic procedure, they also found that properly applying a dynamic resampling strategy requires a more sophisticated approach than simply forcing the standard error below some threshold. The CDR technique is a step in this direction; instead of using standard error thresholds, statistical tests based on the concept of Welch confidence interval are being used. Furthermore, the resampling procedure used in the CDR technique is more efficient since noise is not reduced for *all* solutions in the population, but only for those participating in the evolutionary selection. The motivation of this approach is that noise is not harmful to every element of an EA, but only to those evolutionary processes that involve comparative selection. In other evolutionary operations, such as mating or mutation, noise is irrelevant.

The following section presents a description of the evaluation of the CDR technique on a number of optimisation problems.

[2] It can be noted that Jensen (2003) has suggested an improved version of the non-dominated sorting that reduces the complexity to $O(N \log^M N)$.

4 Optimisation problems

The evaluation has used five benchmark problems and two complex real-world problems from the manufacturing domain. The first real-world problem concerned the optimisation of a manufacturing cell for the production of components for aircraft- and gas turbine engines at Volvo Aero, while the second real-world problem concerned the optimisation of a camshaft machining line at Volvo Cars Engine.

4.1 Benchmark problems

A set of guidelines for the systematic development of benchmark problems for multi-objective optimisation was first proposed in Deb (1999). Based on these guidelines, Zitzler et al. suggest six benchmark functions that have been extensively used in the literature for the analysis and comparison of multi-objective EAs: ZDT1, ZDT2, ZDT3, ZDT4, ZDT5, and ZDT6 (Zitzler et al., 2000). These problems have properties that are known to cause difficulties in converging to the true Pareto-optimal front and reflect characteristics of real-world problems, such as multimodality and high dimensionality. This has motivated the use of these functions in assessing the performance of the CDR technique. However, function ZDT5 has been omitted since it defines a Boolean function over binary strings, and binary encodings are not considered in this study. Artificial noise was added to the functions generated from a zero mean Gaussian distribution whose standard deviation (set to 0.2) represents the amount of noise. Functions ZDT1-ZDT3 were used with 30 input parameters, while ZDT4 and ZDT6 were used with 10 input parameters.

4.2 Volvo Aero

At their factory located at the headquarters in Sweden, Volvo Aero has recently introduced a new manufacturing cell. The cell comprises five multi-task machines and five burring stations, and processes a wide range of different engine components. The cell is highly automated, but some operations, such as burring, are performed manually. Human operators being part of the process is a source of noise in system, as well as there are unpredictable machine breakdowns.

The inflow of the cell is controlled by using fixed inter-arrival times for components, which are to be optimised. The inter-arrival time not only specifies when a component should enter the system, but also determines the component's due time since an overall production strategy is to process no more than one component of a specific type in the cell at a time. This means that if, for example, the inter-arrival time for a component type is set at two hours, a new component of that type is introduced in the cell every two hours with a due time of two hours from the time it was introduced. For efficient production, the inter-arrival times should be specified in a way that maximises the utilisation of the cell (objective 1) and simultaneously minimises overdue components, that is tardiness (objective 2). For high utilisation, short inter-arrival times are needed in order to obtain a high cell load and thereby avoid machine starvation. However, avoiding overdue components requires generous due times; that is long inter-arrival times. This means that the two objectives of maximal utilisation and minimal tardiness conflict with each other.

For the optimisation, a discrete-event simulation model of the manufacturing cell was built using the SIMUL8 software package[3], and a scenario with eleven different component types was specified in the simulation. This means that the problem has eleven decision variables, all continuous. A single simulation run takes approximately 7 minutes including input and output processing. The company's optimisation time budget allowed for 400 simulations.

4.3 Volvo Cars Engine

Volvo Cars Engine manufactures passenger cars, and in this study a factory in Sweden responsible for producing petrol and diesel engine components was considered. The specific focus of the study was on the factory's camshaft machining line, responsible for producing 15 different camshaft variants. The machining line comprises a number of operation groups, each responsible for performing a specific operation on a camshaft being produced. For each operation group, there are a number of parallel machines responsible for actually performing these operations. There are 14 operating groups and 34 machines in total. Unlike an ordinary flow shop with parallel machines, each machine has its own processing time, physical capability and limitations, as well as variability in terms of failures and set-ups. All finished camshafts are taken to a storage area, from where they can later be distributed to other production areas. It is important to always maintain stock levels in the storage area above certain limits in order to ensure that production is not delayed elsewhere because of unexpected events such as machine breakdowns or quality defects (noise). Camshafts are produced in batches (that is, a collection of camshafts of the same kind), with each batch being prioritised for production in order to keep stock levels in the storage area at an acceptable level. Prioritisation is determined by a schedule, which also defines the individual path through machines and operations that the batches should take. Stock levels are checked at continuous time intervals, and a mean value of the shortage noticed at each measure point is calculated at the end of the scheduling period. Besides minimising product shortage (objective 1), it is also important that a schedule results in a throughput of the line that is as high as possible for maximum efficiency (objective 2). For high throughput, there should ideally be only one variant produced in the line at the same time to avoid set-up times for machines, which cause a large overhead. Furthermore, for a high throughput, variants with shorter processing times should be prioritised before those with longer processing times. However, to maintain the minimum stock levels, an even mix of the different variants being produced in the line is needed and variants should be prioritised so that shortage is avoided. In other words, the objectives of minimum shortage and maximum throughput conflict with each other.

In evaluating the new noise handling technique when applied to the problem at Volvo Cars Engine, as well as the previous problems described, it is integrated in a multi-objective surrogate-assisted EA which is described in the next section.

For the optimisation of the machining line, a discrete-event simulation was constructed using the QUEST simulation software[4]. The simulation model was primarily built for short-period scheduling, and in this study a seven day of production period was simulated in the model. Such a period includes

[3] www.simul8.com
[4] www.3ds.com

scheduling about 500 batches, and the task of the optimisation algorithm is to set a unique priority for each batch and also to define its path through the machining line. It takes approximately 5 minutes to simulate seven days of production, input and output processing included. The company's time budget allowed for the performance of 600 simulations.

5 Multi-Objective Parallel Surrogate-Assisted Evolutionary Algorithm

The proposed CDR technique is integrated in the parallelised algorithm "Multi-Objective Parallel Surrogate-Assisted EA" (MOPSA-EA) (Syberfeldt et al., 2008). This section outlines the fundamentals of MOPSA-EA and presents how its parameters have been configured in the optimisation problems described in the previous section. Pseudo code for MOPSA-EA can be found on-line alongside the electronic version of the paper (Algorithm 1).

5.1 Basic algorithm

In MOPSA-EA, initially, the first generation of the population P is filled with as many random solutions as there are processing nodes. These are created by the master node and sent to the slave nodes for exact evaluation. When evaluated solutions are returned from the slaves, the master immediately creates new offspring to be evaluated through crossover and mutation (the implementation of these operators is problem specific). Parents in P to be used for the creation of offspring are chosen using *crowding tournament selection* (Deb, 2004). With tournament selection, solutions that have bad objective values may also be selected, which maintains diversity in the population and prevents premature convergence. For the tournament, two solutions A and B are chosen randomly and a non-dominated sort is performed. A is declared the winner if either

(i) A has a better Pareto rank than B, or

(ii) A and B have the same Pareto rank, but A has a larger crowding distance than B [5].

In the tournament selection, noise in the evaluation of solutions is taken into consideration using the CDR technique described in Section 3.

When generating offspring, a pool of λ candidate offspring is created (in this study, λ is considered being constant). An offspring pool, called O, is created as soon as a processing node becomes available. The solutions in O are evaluated by the surrogate, and since the computational cost of surrogate evaluations can be neglected in real-world optimisations (Emmerich et al., 2002), the size of the pool can be large. The surrogate objective values assigned to solutions in O are adjusted to take the imprecision of the surrogate into consideration. This is done by modifying the values based on the calculated error of the surrogate (the details of this procedure are described in Syberfeldt et al., 2008). Based on the adjusted surrogate objective values, the most promising solution in O is selected to be inserted into P. In this procedure, through a non-dominated sort, all solutions of rank 1 in O are identified (called

[5] It can be noted that the hypervolume indicator has been suggested as a powerful alternative to the crowding distance measure (see for example Emmerich et al., 2005).

O_{R1}) and checked for domination against all solutions of rank 1 in P (called P_{R1}). By only identifying O_{R1} and P_{R1}, a full non-dominating sort is avoided. The solution in O_{R1} that dominates the most solutions in P_{R1} is selected, mutated and precisely evaluated. If several solutions in O_{R1} share the position of dominating most solutions in P_{R1}, the one having the largest Euclidean distance to its closest neighbour in P_{R1} is selected. Before the selected offspring is inserted into P, the worst solution in P is removed by performing a non-dominated sort and discarding the solution with the smallest crowding distance in the last rank. An elitistic approach is used, in which an offspring is only inserted into P if it is not dominated by any solution in P.

The sample obtained from a newly inserted offspring may be used to update the surrogate. An update does not need to take place every time a new sample becomes available, but can occur only every N:th sample. Less frequent updates can save a lot of time, but can also decrease the quality of the surrogate. In this study, N is set to 10. Since the algorithm is neutral with respect to the surrogate technique, it does not specify how to update the surrogate. Surrogate update strategies vary between different techniques, and are also highly problem dependent (Jin, 2005).

5.2 Surrogate configuration

MOPSA-EA allows for any kind of surrogate, and in this paper two different surrogate techniques are used. In the Volvo Aero problem, an artificial neural network is used. Several properties of artificial neural networks (ANNs) make them beneficial for use as surrogates, including universal approximation characteristics, good extrapolation/generalisation ability, applicability to multivariate non-linear problems, ability to handle noise in data sets, and no inherent assumption about data correlations. The ANN used has one hidden layer, since it has been shown that one hidden layer is sufficient for nearly all problems (e.g. Chen et al., 1995). The number of hidden nodes in the ANN is dynamically adapted depending on the number of samples available. For good performance, it is recommended that the number of weights in the ANN is proportional to the size of the training data set (Mehrotra et al., 1996). Since the number of samples continuously increases during the optimisation, a static number of hidden nodes is not appropriate. Therefore, the optimisation started with an ANN with one single hidden node. When the number of available samples exceeded five times the number of weights in the network, a new hidden node was added (according to the weight-sample ratio suggested in Mehrotra et al., 1996). The ANN was trained using back-propagation with the sigmoid function and a learning rate of 0.5. For each 10th simulation, the ANN was re-trained with samples from the most recently simulated solutions (at most 50 samples). In case any of these solutions had been simulated more than once, the mean simulation values were used in the training. The idea of regularly re-training the ANN with the most recent samples is to have a local surrogate defined over the current search region. Local ANNs are preferable to global ANNs in surrogate-assisted optimisation (Giotis et al., 2000; Jin, 2005), since they reduce the time consumption of the training process (Jin and Branke, 2005). To avoid overfitting, 10-folded cross validation was used in the training.

In the Volvo Cars Engine problem, constructing a useful ANN is not possible since the number of simulation inputs is very large (about 500) and dynamic. An ANN with over 500 inputs includes tens of thousands of network weights, and such a network cannot perform well when the problem is complex

and the number of data samples is limited. Therefore, we have constructed a so called surrogate model instead of an ANN for the Volvo Cars Engine problem. While an ANN treats the simulation as a black box, knowing nothing about its inner workings, a surrogate model treats the simulation as a white box and explicitly attempts to imitate its internals. The surrogate model is built in the C# programming language and solves the same problem as the simulation through a number of simplifications (for example, carts transporting camshaft between machines are not modelled). Since it is less complex than the simulation, it is also computationally cheaper and thereby serves the same purpose as an ANN.

Note that no surrogate was needed in the optimisation of the ZDT benchmark problems since these are considerably faster to run than the real-world problems.

5.3 Algorithm parameter settings

The configuration of the parameters of MOPSA-EA is presented in Table 1. The ZDT functions use common settings, while for the real-world problems the parameter values were set in discussion with system experts on the basis of domain knowledge. The system experts (simulation technicians and production engineers) were introduced to the various parameters and asked to provide proper values based on their knowledge and a limited number of simulation tests. Thorough parameter pre-tuning could not be carried out within a practical time frame due to the time-consumption of the simulations used to evaluate solutions.

	ZDT	Volvo Aero	Volvo Cars Engine
Population size	50	40	60
Number of offspring	25	20	30
Mutation step size	0.5	1.0	1.0
Crossover operator	Single-point	Single-point	Single-point
Crossover probability	0.8	0.8	0.8

Table 1: Algorithm parameter settings.

In the next section, the evaluation of the CDR technique when integrated in MOPSA-EA is presented.

6 Evaluation

In assessing the performance of the proposed noise handling technique, it is compared to four existing techniques to reduce noise, which are described in Section 6.1. The performance metrics used in the comparison are presented in Section 6.2. Results of the different noise handling techniques when integrated in MOPSA-EA are presented in Section 6.3, and an overall analysis of these follows in Section 6.4.

6.1 Performance comparison

As described in Section 2, there are four major techniques for handling noise: (*i*) static resampling, (*ii*) modified Pareto ranking scheme, (*iii*) dominance-dependent lifetime, and (*iv*) fitness inheritance. The CDR technique was compared to all four of these. The implementation of the techniques in the evaluation is described below. It is important to note out that with all techniques (including CDR), the total number of simulations performed in an optimisation problem is the same. Consequently, the number of unique solutions evaluated may vary with the different techniques.

Static resampling

In the implementation of this technique, each solution is sampled five times and the average objective values are used. This is the same number of samplings used at maximum in the CDR technique, which means that the noise reduction of the static resampling scheme will never be worse than that of CDR.

Modified Pareto ranking scheme

There are two different approaches for modifying the original Pareto ranking scheme to handle noise; the probability-based technique and the cluster-based technique. Considering the drawbacks of the probability-based technique discussed in Section 2.2, cluster-based ranking was used in the evaluation performed in this study. The cluster-based technique involves the user-defined parameter K, defining the neighbourhood restriction factor. Babbar et al. (2003) set K according to Equation 7

$$K = C * \left(1 - e^{-\frac{\beta}{Ge}}\right), \tag{7}$$

where C and β are problem-specific constants, and Ge is the current generation. In this study, the technique is implemented in a steady-state algorithm and Ge is therefore undefined, since the steady-state algorithm does not use a generation-based population refinement strategy. K can therefore not be set according to Equation 7, but the formula is therefore adjusted to fit a steady-state approach according to Equation 8

$$K = C * (maxSim - numSim), \tag{8}$$

where C is a constant set to 0.0001 (a value found through experimental tuning[6]), $maxSim$ is the maximum number of simulations allowed for the optimisation (a constant value), and $numSim$ is the number of simulations performed at the current point in time (a variable number). Similar to Equation 7, the neighbourhood decreases over time with Equation 8.

Dominance-dependent lifetime

In the implementation of this technique, the lifetime lt of a new solution i entering the population is set according to Equation 9

[6]Parameter tuning is usually not applicable but was necessary in this case since no information about appropriate C values was available.

$$i_{lt} = popSize - i_{nd}, \tag{9}$$

where $popSize$ is the population size, and i_{nd} is the number of solutions i is dominating. Equation 9 ensures that a short lifetime is assigned if a large number of solutions in the population are dominated and vice versa. A non-dominating solution whose lifetime has expired is resampled once and added to the population with its new objective values.

Fitness inheritance

With this technique, a child inherits the mean objective value $\bar{\mu} = \frac{\mu_{parent1}+\mu_{parent2}}{2}$ and the mean standard deviation $\bar{\sigma} = \frac{\sigma_{parent1}+\sigma_{parent2}}{2}$ from its parents. The child is evaluated once, and if each obtained objective value falls within the confidence interval $\left(\bar{\mu} - 3 * \bar{\sigma} \leq f \leq \bar{\mu} + 3 * \bar{\sigma}\right)$, the inherited fitness is accepted. Otherwise, the child is resampled four times and assigned the mean values and standard deviation of its samplings.

CDR

In the implementation of this technique, two parameters must be specified: confidence level and maximum number of samplings. In the evaluation, these parameters are set according to Table 2. Note that the parameters have not been tuned before the evaluation. As previously discussed, elaborate parameter pre-tuning can usually not be afforded in real-world applications with time-consuming simulations, and it is therefore necessary that the technique is not too sensitive with respect to its parameter settings.

	Confidence level	Max samplings
Rank 1	0.75	5
Rank 2	0.70	4
Rank 3	0.65	3
Rank 4	0.60	2
Rank 5 and higher	0.55	2

Table 2: Confidence levels and maximum number of samplings for different ranks.

In Section 3.2, two potential improvements in the implementation of the CDR technique were discussed. The first one was to check if the two solutions being compared are significantly different with respect to the highest confidence level of rank, and thereby avoid a non-dominating sort to be performed unnecessarily. The second one was to perform less frequent resamplings in the beginning of the search, and increase the resampling frequency when the optimisation begins to converge. In the evaluation, CDR was implemented with the extra confidence check, but not with the variable resampling frequency in order to avoid additional user-defined parameters.

6.2 Evaluation metrics

An overall goal in multi-objective optimisation is convergence to the Pareto-optimal front. A commonly used measure for evaluating convergence in problems having a known true optimal front (which is the case with the ZDT functions) is the Υ metric (Deb et al., 2002). This metric measures the degree of convergence by calculating the average minimum Euclidean distances from each of the obtained non-dominated solutions to the closest solution in the true Pareto front. The smaller the value of Υ, the better the convergence of the algorithm.

A commonly used measure for comparing the results of multi-objective EAs is the S metric (also called the hypervolume metric). The S metric is one of the most frequently applied measures for multi-objective optimisation (Emmerich et al., 2005). Basically, S measures the volume in objective space dominated by obtained solutions. The larger the volume, the better the results of the algorithm. The S metric is a combined measure of convergence and diversity in the set of non-dominated solutions. It does not assume that the true Pareto-optimal front is known and can therefore also be applied to real-world problems.

Another metric that is a combined measure of convergence and diversity is the inverted generational distance metric (Coello Coello and Reyes-Sierra, 2004). Inverted generational distance (IGD) aims at being a combined measure of convergence and diversity in the set of non-dominated solutions. This metric is calculated by taking the average of all Euclidean distances from each true Pareto front sample to the closest solution generated by the algorithm. The rationale behind this metric is that for a low IGD value, a well spread front and a good convergence is necessary at the same time.

6.3 Results

Benchmark problems

Results from the noisy ZDT functions are shown in Table 3, including CPU time consumption on a single workstation[7] stated in seconds (sec) and hundreds of a second (hsec). The metric values are based on normalised objective values and constitute the average of 500 independent runs. The results have a 99% confidence probability according to Welch's t-test (defined in Law and Kelton, 2000). In calculating the Υ and IGD metrics, a set of 500 uniformly-distributed solutions of the true Pareto set was derived. The S metric requires a reference point, which was set to (x,y)-coordinates that are just outside the worst values in objective space taken by the function, namely 1.5, 8.0. Based on the reference point, the S value was normalised between [0,1], where a value of 1 coincides with the optimal value possible (i.e., the hypervolume of the true Pareto front). The optimisation was performed for 5000 function evaluations, which is a relatively small number of evaluations compared to other studies on the ZDT functions (usually, about 25,000 evaluations are used). A small number of evaluations have been allowed to emulate a scenario in which the evaluation function is computationally expensive. The function can therefore only be called a relatively small number of times.

As Table 3 illustrates, the best results with respect to the performance metrics are achieved with the CDR technique. The techniques of static resampling and cluster-based Pareto ranking achieve about

[7] Equipped with Intel dual core processors of 2.13GHz, 2 GB of RAM, and Microsoft Windows XP Professional.

the same results and share second place. Dominance-dependent lifetime is the fourth best technique, whilst the fitness inheritance technique achieves the worst results. With respect to CPU time consumption, the differences between the five techniques are only a couple of milliseconds.

	Υ (minimise)	IGD (minimise)	S (maximise)	CPU time (sec:hsec)
ZDT1				
Conf. based dynamic resampling	0.209	0.14	0.801	5:96
Static resampling	0.228	0.161	0.742	5:31
Dominance-dependent lifetime	0.216	0.168	0.635	5:47
Cluster-based Pareto ranking	0.213	0.16	0.66	5:39
Fitness inheritance	0.331	0.196	0.637	5:51
ZDT2				
Conf. based dynamic resampling	0.464	0.396	0.756	5:2
Static resampling	0.573	0.622	0.618	4:77
Dominance-dependent lifetime	0.56	0.617	0.622	4:85
Cluster-based Pareto ranking	0.501	0.492	0.691	5:01
Fitness inheritance	0.628	0.784	0.405	4:96
ZDT3				
Conf. based dynamic resampling	0.193	0.105	0.901	4:83
Static resampling	0.325	0.212	0.815	4:22
Dominance-dependent lifetime	0.468	0.279	0.777	4:44
Cluster-based Pareto ranking	0.359	0.22	0.817	4:59
Fitness inheritance	0.516	0.302	0.742	4:54
ZDT4				
Conf. based dynamic resampling	0.881	0.342	0.873	4:06
Static resampling	0.927	0.389	0.86	3:27
Dominance-dependent lifetime	1.423	1.101	0.802	3:63
Cluster-based Pareto ranking	1.166	0.477	0.858	3:85
Fitness inheritance	1.512	1.018	0.8	3:92
ZDT6				
Conf. based dynamic resampling	0.601	0.503	0.716	3:91
Static resampling	0.682	0.589	0.691	3:24
Dominance-dependent lifetime	0.742	0.57	0.714	3:57
Cluster-based Pareto ranking	0.676	0.584	0.71	3:8
Fitness inheritance	0.817	0.628	0.683	3:72

Table 3: Benchmark results.

Volvo Aero

Table 4 presents the results from the optimisation of the Volvo Aero problem. The result values constitute the average of ten independent runs (due to the computational expense associated with the simulation, only a relatively small number of runs could be undertaken). The results have an 80% confidence probability. Since the true Pareto-optimal front of the Volvo Aero problem is unknown, as with real-world problems in general, only the S metric could be calculated. The reference points used in the S metric were set just outside the theoretically best and worst values, respectively, for the system (specified by domain experts).

As shown in the table, the CDR technique achieves the best value. The static resampling technique and the cluster-based Pareto ranking technique are on second place (these achieve almost identical values). Dominance-dependent lifetime is the fourth best technique, whilst the fitness inheritance technique is the worst.

	S
Conf. based dynamic resampling	0.496
Static resampling	0.467
Dominance-dependent lifetime	0.411
Cluster-based Pareto ranking	0.468
Fitness inheritance	0.393

Table 4: Results Volvo Aero problem.

Volvo Cars Engine

The results of the optimisation of the Volvo Cars Engine problem are presented in Table 5 (average of ten independent runs). The optimisation was performed for 600 simulations and the results have a 80% confidence probability. As in the Volvo Aero problem, the true Pareto-optimal front is unknown and only the S metric was calculated. Also in this case, the reference points for the S metric were set just outside the theoretically best and worst values, respectively, for the system.

	S
Conf. based dynamic resampling	0.511
Static resampling	0.483
Dominance-dependent lifetime	0.409
Cluster-based Pareto ranking	0.485
Fitness inheritance	0.398

Table 5: Results Volvo Cars Engine problem.

6.4 Analysis

In this subsection, the results of the five noise handling techniques are discussed. The sensitivity of the user-defined parameters of the proposed CDR technique is also analysed.

Performance

In the optimisations, the CDR technique has the best performance in all problems. A key factor contributing to its good results is most likely to be the dynamic resampling approach. The other techniques, except dominance dependent lifetime, use a static strategy in which all solutions entering the population are resampled a fixed number of times. Such a static approach is inefficient when considering that: (*i*) the same number of simulations are spent on solutions of higher ranks as on inferior solutions of lower ranks, (*ii*) all solutions are simulated the same number of times although they are

subject to different amounts of noise[8], and (*iii*) simulations are wasted on solutions that never take part in the evolutionary process, that is, solutions which never influence the search. In contrast, the proposed technique uses a resampling scheme in which the number of samplings is automatically adjusted to the rank and noise of a solution. Furthermore, only solutions that are part of the evolutionary process are resampled. In this way, an efficient utilisation of simulations is achieved and the desired confidence level is reached using as few resamplings as possible.

The second best results in the optimisations are achieved by the techniques of static resampling and cluster-based Pareto ranking (these two achieve about the same overall results). The technique of static resampling is surprisingly efficient, especially in the real-world problems. Static resampling has previously been considered inappropriate when the number of simulations is limited due to their high computational cost (Branke et al., 2001; Branke and Schmidt, 2003). However, with respect to its simplicity and relative efficiency, it can be argued that this technique should not be rejected. There are also enhanced versions of the static resampling technique which may have potential to perform even better than the basic version, for example the approach described by Tan et al. (2007) in which the number of samplings increases with the generations.

Ranked as number four in most of the optimisations is the technique of dominance-dependent lifetime. As previously discussed in Section 2, this technique does not explicitly resample solutions, but instead evaluates solutions anew and adds them to the population with their new objective values. Basically, this means that a noisy sample of a solution is simply replaced by another noisy sample, which might explain the weak results of the technique.

The worst results in the optimisations are achieved by the technique of fitness inheritance, especially in the real-world problems. Similar to the technique of dominance dependent lifetime, the poor results might be explained by an inadequate noise reduction approach. In the technique of fitness inheritance, the greater the standard deviation of a solution's parents (that is, the larger their amount of noise), the greater the probability that the inherited values are accepted without resampling. From a noise reduction perspective, this seems counter-intuitive.

User-defined parameters

All five noise handling techniques studied in this work suffer from arbitrary user-defined parameters. CDR requires the specification of two parameters: (*i*) confidence level, and (*ii*) maximum number of samplings. These parameters must be determined by the user based on the total number of simulations allocated, the desired level of accuracy of the optimisation, and the noise characteristics of the problem. The total number of simulations is usually known, but this might not be the case with the desired level of accuracy and certainly not with the noise characteristics. It is therefore interesting to analyse whether or not the user must have a good understanding of how to set the confidence level and maximum number of samplings, that is, whether their configuration has a significant impact on the performance of the optimisation. To investigate this, we compared the results of a number of different parameter configurations on the ZDT1 benchmark problem with noise size 0.2 (Table 6). In the

[8]Not that this applies only to the real-world problems – in the ZDT problems all solutions are subject to the same amount of noise.

experiments, different confidence levels were tested in combination with three different settings of the maximum number of samplings. The column to the left in Table 6 defines the confidence level for rank 1. The confidence level for rank 2 is the confidence level for rank 1 minus 0.05, the confidence level for rank 3 is the confidence level for rank 2 minus 0.05, etc.

Confidence level	Max. number of samplings (rank1-rank2-rank3)	Υ (minimise)	IGD (minimise)	S (maximise)
0.95	5-4-3	0.22	0.146	0.767
0.95	7-5-3	0.223	0.139	0.812
0.95	5-2-2	0.218	0.188	0.778
0.85	5-4-3	0.21	0.164	0.756
0.85	7-5-3	0.209	0.148	0.758
0.85	5-2-2	0.208	0.125	0.831
0.75	5-4-3	0.209	0.14	0.801[9]
0.75	7-5-3	0.185	0.108	0.83
0.75	5-2-2	0.189	0.138	0.811
0.65	5-4-3	0.178	0.117	0.834
0.65	7-5-3	0.179	0.109	0.829
0.65	5-2-2	0.207	0.117	0.799
0.55	5-4-3	0.212	0.162	0.748
0.55	7-5-3	0.208	0.15	0.767
0.55	5-2-2	0.214	0.147	0.785

Table 6: Experiments on ZDT1 noise level 0.2.

As Table 6 illustrates, the configuration of the two parameters has some effect on the optimisation results. For the parameter "confidence level", it seems that the extremes (that is, level 0.95 and level 0.55) give the worst results, while levels in between give the best results (0.65 and 0.75). For the parameter "maximum number of samplings", a trend seems to be that doing fewer samplings is advantageous in combination with a high confidence level. An explanation of this might be that EAs tolerate, and can even be helped by, a small amount of randomness in the evolutionary process. Excessive noise reduction may therefore lead to a waste of simulations.

The impact of parameter settings on performance is, however, relatively small and regardless of which configuration is considered, the CDR technique is still better than the other techniques. This indicates that the proposed technique does not rely upon a perfect parameter configuration, and that its performance is satisfactory even if the user is not sure of how to set the parameters. For maximum efficiency, however, trial-and-error in finding the optimal settings is needed.

7 Conclusions

The new technique to deal with noise uses an iterative resampling procedure that efficiently reduces the noise by varying the number of samples used per solution based on the amount of noise in the local area of the search space. This dynamic strategy avoids wasteful samplings when additional sampling is of little benefit, and promotes additional samplings when this is beneficial. Contrary to several

existing techniques for noise handling, the new technique does not reduce noise for all solutions in the population, only for those participating in the evolutionary selection. The motivation for this approach is that noise is not harmful to every element of an EA, only to those evolutionary processes that involve comparative selection. In other evolutionary operations, such as mating or mutation, noise is of minor importance.

Similar to other noise handling techniques, the proposed technique involves user-defined parameters. Although the technique works well with a standard setting of the parameters, ideally there should be no user-defined parameters at all. Investigating how to get rid of the user-defined parameters, thereby making the use of the technique simpler, is an important topic for future research. Another potential improvement that we will evaluate in the future is to let a surrogate (e.g. an ANN) represent the noise landscape based on the information received when resampling solutions. If it is possible to construct a surrogate of adequate quality, this can be used to estimate the noise of a solution and simulation evaluations can thereby potentially be saved. A further aspect that we will investigate in the future related to surrogates is the relationship between noisy evaluations and the training of surrogates. It is known that a small amount of noise in the training data often improves the surrogate's ability to generalise on new data (Holmström and Koistinen, 1992). However, in real-world simulation-optimisation *every* data sample used to train the surrogate is subject to a large amount of arbitrary noise. Such high degree of unknown randomness in the data samples causes an unstable training process and a deterioration of the accuracy of the surrogate. For improved efficiency of the optimisation, reducing the noise is therefore not only important with respect to the EA, but also with respect to the training of the surrogate. As far as we know, the problem of noise upon surrogate training has not been previously addressed within the field of multi-objective evolutionary optimisation.

In the next stage of this work, we will also consider robustness. The concept of robustness differs from noise in that the variations are not in the simulation (that is, the objective function), but in the input parameters. Robustness is important since in real-world problems it cannot usually be guaranteed that the exact parameters of a solution are actually implemented, but rather a solution close to the original one. We will investigate if it is possible to utilise the resamplings performed in the noise handling technique to bias the evolutionary process towards more robust solutions without any additional resamplings. This could be done by introducing small changes in the input parameters of a solution being resampled. A non-robust solution will then achieve poor results and be consequently punished in the evolutionary selection.

References

April, J., Better, M., Glover, F. and Kelly, J. (2004). New advances for marrying simulation and optimization, *Proceedings of the 2004 Winter Simulation Conference*, Washington, DC, pp. 80–86.

Arnold, D. and Beyer, H. (2002). *Noisy Local Optimization with Evolution Strategies*, Kluwer Academic Publishers, Norwell, MA. ISBN: 1402071051.

Babbar, M., Lakshmikantha, A. and Goldberg, D. E. (2003). A modified NSGA-II to solve noisy multiobjective problems, *Proceedings of Genetic and Evolutionary Computation Conference*, Vol. 2723

of *Lecture Notes in Computer Handling Uncertainty in Indicator-Based Multiobjective Optimization*, Springer Verlag, Chicago, Illinois, USA, pp. 21–27.

Büche, D., Stoll, P., Dornberger, R. and Koumoutsakos, P. (2002). An evolutionary algorithm for multi-objective optimization of combustion processes, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* **32**(4): 460–473.

Beyer, H. (2000). Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice, *Computer Methods in Applied Mechanics and Engineering* **186**(2-4): 239–267.

Branke, J., Meisel, S. and Schmidt, C. (2007). Simulated annealing in the presence of noise, *Journal of Heuristics* **14**(6): 627–654.

Branke, J. and Schmidt, C. (2003). Selection in the presence of noise, *in* E. C.-P. et.al. (ed.), *Proceedings of Genetic and Evolutionary Computation Conference*, number LNCS 2273, Springer Verlag, Chicago, Illinois, pp. 766–777.

Branke, J., Schmidt, C. and Schmeck, H. (2001). Efficient fitness estimation in noisy environments, *in* L. S. et al. (ed.), *Proceedings of Genetic and Evolutionary Computation Conference*, Morgon Kaufmann, San Francisco, California, pp. 243–250.

Bui, L., Abbass, H. and Essam, D. (2005). Fitness inheritance for noisy evolutionary multi-objective optimization, *Proceedings of Genetic and Evolutionary Computation Conference*, Washington, DC, USA, pp. 779–785.

Chen, T., Chen, H. and Liu, R. (1995). Approximation capability in C(Rn) by multilayer feedforward networks and related problems, *IEEE Transactions on Neural Networks* **6**(1): 25–30.

Coello Coello, C., Lamont, G. and Veldhuizen, D. V. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*, Genetic and Evolutionary Computation, 2nd edn, Kluwer Academic Publishers.

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

Deb, K. (1999). Multi-objective genetic algorithms: Problem difficulties and construction of test functions, *Evolutionary Computation* **7**: 205–230.

Deb, K. (2004). *Multi-Objective Optimization using Evolutionary Algorithms*, second edn, John Wiley & Sons Ltd.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation* **6**(2): 182–197.

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

Emmerich, M., Beume, N. and Naujoks, B. (2005). An EMO algorithm using the hypervolume measure as selection criterion, *Proceedings of Evolutionary Multi-Criterion Optimization*, Vol. 3410 of *Lecture Notes in Computer Science*, Springer Verlag, Guanajuato, Mexico, pp. 62–76.

Emmerich, M., Giotis, A., Özdemir, M., Bäck, T. and Giannakoglou, K. (2002). Metamodelassisted evolution strategies, *in* J.J. Merelo Guervós et al. (ed.), *Proceedings of the International Conference on Parallel Problem Solving from Nature*, Springer Verlag, Granada, Spain, pp. 361–370.

Fitzpatric, J. and Grefenstette, J. (1988). Genetic algorithms in noisy environments, *Machine Learning* **3**: 101–120.

Giotis, A., J., K. G. and Periaux (2000). A reduced-cost multi-objective optimization method based on the pareto front technique, neural networks and pvm, *Proceedings of European Congress on Computational Methods in Applied Sciences and Engineering*, CD-ROM, Barcelona , Spain.

Goh, C. and Tan, K. (2007). An investigation on noisy environments in evolutionary multi-objective optimization, *IEEE Transactions on Evolutionary Computation* **11**(3): 354–381.

Holmström, L. and Koistinen, P. (1992). Using additive noise in back-propagation training, *IEEE Transactions on Neural Networks* **3**(1): 24–38.

Hughes, E. (2001). Evolutionary multi-objective ranking with uncertainty and noise, *Proceedings of First International Conference on Evolutionary Multi-Criterion Optimization*, Springer Verlag, Zurich, Switzerland, pp. 329–343.

Jensen, M. T. (2003). Reducing the run-time complexity of multiobjective eas: The NSGA-II and other algorithms, *IEEE Transactions on Evolutionary Computation* **7**(5): 503–515.

Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation, *Soft Computing* **9**: 3–12. o.

Jin, Y. and Branke, J. (2005). Evolutionary optimization in uncertain environments - a survey, *IEEE Transactions on Evolutionary Computation* **9**(3): 303–317.

Law, A. M. and Kelton, D. (2000). *Simulation Modeling and Analysis*, third edn, Mc Graw Hill.

Lee, L., Chew, E., Teng, S. and Chen, Y. (2008). Multi-objective simulation-based evolutionary algorithm for an aircraft spare parts allocation problem, *European Journal of Operational Research* **189**(2): 476–491.

Mehrotra, K., Mohan, C. and Ranka, S. (1996). *Elements of Artificial Neural Networks*, MIT Press. ISBN 0-262-13328-8.

Miller, B. and Goldberg, D. (1996). Genetic algorithms, selection schemes, and the varying effect of noise, *Evolutionary Computation* **4**(2): 113–131.

Ong, Y., Nair, P., Keane, A. and Wong, K. (2004). Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems, *in* Y. Jin (ed.), *Knowledge Incorporation in Evolutionary Computation, Studies in Fuzziness and Soft Computing*, Springer Verlag, pp. 307–332.

Srinivas, N. and Deb, K. (1995). Multiobjective optimization using nondominated sorting in genetic algorithms, *Evolutionary Computation* **2**(3): 221–248.

Syberfeldt, A., Grimm, H., Ng, A. and John, R. (2008). A parallel surrogate-assisted multi-objective evolutionary algorithm for computationally expensive optimization problems, *Proceedings of the 2008 IEEE Congress on Evolutionary Computation*, Hong Kong, pp. 3177–3184.

Tan, K. C. and Goh, C. K. (2008). Handling uncertainties in evolutionary multi-objective optimization, *Computational Intelligence: Research Frontiers*, Vol. 5050 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 262–292.

Tan, K., Cheonga, C. and Goh, C. (2007). Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation, *European Journal of Operational Research* **177**(2): 813–839.

Teich, J. (2001). Pareto-front exploration with uncertain objectives, *Proceedings of First International Conference on Evolutionary Multi-Criterion Optimization*, Springer Verlag, Zurich, Switzerland, pp. 314–328.

Zitzler, E., Deb, K. and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results, *Evolutionary Computation* **8**(2): 173–195.