Multi-Objective Simulation-Based Optimization of Production Systems with Consideration Noise

Anna Syberfeldt[1], Henrik Grimm[1], Amos Ng[1]
[1]Center for Intelligent Automation, University of Skövde, Skövde, Sweden
anna.syberfeldt@his.se

ABSTRACT

Many production optimization problems approached by simulation are subject to noise. When evolutionary algorithms are applied to such problems, noise during evaluation of solutions adversely affects the evolutionary selection process and the performance of the algorithm. In this paper we present a noise compensation technique that efficiently deals with the negative effects of noisy simulations in multi-objective optimization problems. Basically, this technique uses an iterative re-sampling procedure that reduces the noise until the likelihood of selecting the correct solution reaches a given confidence level. The technique is implemented in MOPSA-EA, an existing evolutionary algorithm designed specifically for real-world simulation-optimization problems. In evaluating the new technique, it is applied on a benchmark problem and on two real-world problems of manufacturing optimization. A comparison of the performance of existing algorithms indicates the potential of the proposed technique.

Keywords: simulation, optimization, noise.

1. INTRODUCTION

Real-world production problems often contain nonlinearities, combinatorial relationships and uncertainties that are too complex to be modelled analytically. In these scenarios, simulation-based optimisation (SO) is a powerful tool to determine optimal system settings [1]. SO is the process of finding the best values of some parameters for a system, where the performance of the system is evaluated based on the output from a simulation model of the system. In a production system, for example, one might be interested in finding the optimal buffer settings with respect to throughput of the system and average cycle time of products. Finding the optimal parameter values is an iterative simulation-optimisation process. An optimisation procedure generates a set of parameter values and feeds them to a simulation that estimates the performance of the system. Based on the evaluation feedback from the simulation, the optimisation procedure generates a new set of parameter values and the generation-evaluation process continues until a user-defined stopping criterion is satisfied.

While traditional analytical optimization methods have been unable to cope with the challenges imposed by many real-world production problems approached by simulation, such as multimodality, non-separability and high dimensionality, evolutionary algorithms (EAs) have proven to be powerful in searching reasonably good solutions. EAs are also particularly suitable to solve problems that require simultaneous optimization of more than one objective, which is often the case in real-world applications [2]. The difficulty with problems of multiple objectives is that there is usually no single optimal solution with respect to all objectives, as improving performance on one objective deteriorates performance of one or more of the other objectives. Instead of a single optimum, there is a set of optimal trade-offs between the conflicting objectives, known as the Pareto-optimal solutions or the Pareto-front. A solution is Pareto-optimal if it is not dominated by any other solution, i.e. there exist no other solution that is superior in one objective without being worse in another objective. Different ranks of domination can also be assigned, where rank 1 includes the Pareto-optimal solutions in the complete population, rank 2 the Pareto-optimal solutions identified when temporarily discarding all solutions of rank 1, etc. Since EAs maintain a population of solutions they can, contrary to many other optimization techniques, capture multiple Pareto-optimal solutions in one single optimization run.

In real-world simulation-optimization using EAs, one has not only to cope with multiple optimization objectives, but also with noise. In practical production problems, the existence of noise during evaluation of solutions is inevitable [3-4] For example, when trying to optimize the operation of an assembly machine by tuning machine control parameters, the outcome from different evaluations will not be identical even though all parameters have been fixed. The sources of noise causing the unpredictable outcome can be manifold, such for example human operators or worn-out parts of the machine. To capture the stochastic behaviour of systems like this, simulations contain randomness. Instead of modelling only a deterministic path of how the system evolves in the process of time, a stochastic simulation deals with several possible paths based on random variables in the model. In situations like these, improving the outcome can be a risky endeavour because one can never be sure that a seeming improvement obtained by a certain control parameter change is a real improvement. If estimates of the objective function come from only one simulation replication the normal path of the EA can be severely disturbed with substantial performance degradation as a consequence. Since noisy fitness values are used for selection, the algorithm can be mislead to propagate inferior solutions; bad ones might be kept for the next generation, and the good ones might be excluded [4].

To address the problem of noise in simulations is critical for successful results in real-world production problems. Compared to its practical relevance, however, noise compensation has gained only limited attention in EA research [5], and especially in multi-objective optimization there are few studies devoted to optimization of noisy problems [4]. The common technique to handle noise is to send the algorithm with the average values of output samples obtained from a large number of simulation replications. Although this technique is easy to implement, the large number of replications needed to obtain statistically confident estimates from computationally expensive simulation models of complex systems can easily render the approach to be totally impractical.

In this paper we present new noise compensation technique for evolutionary selection in multi-objective problems that the efficiently deal with negative effects of noise in simulations. This technique uses an iterative re-sampling procedure that reduces the noise until the likelihood of selecting the correct solution reaches a given confidence level. The noise compensation technique is integrated in an existing EA called "Multi-Objective Parallel Surrogate-Assisted EA" (MOPSA-EA) (described in [6]). MOPSA-EA is designed to reduce the huge time-consumption associated with many simulation-based optimization problems. Real-world optimization problems often involve an immense number of possible solutions, and an EA requires a large number of simulations before an acceptable solution is be found [7-8]. This holds true especially in multi-objective problems, where a significantly larger portion of the search space needs to be explored to obtain the whole Pareto-optimal set [9]. Even with improvements in computer processing speed, one single simulation evaluation may take a couple of minutes to hours or even days of computing time [10]. To tackle the problem of efficiency, MOPSA-EA supports a high degree of parallelism by implementing the master-slave parallelization scheme in combination with a steady-state design. For improved efficiency, the algorithm also utilizes a simulation surrogate (also called metamodel). The surrogate is used to screen candidate solutions and identify the most promising one. Instead of generating only a single offspring, which is normally done in steady-state algorithms, a pool of multiple offspring is created. Each of the offspring is evaluated by the surrogate, and the best one is simulated and inserted into the population. In the next section, the details of MOPSA-EA are further described.

2. DESCRIPTION OF MOPSA-EA

In MOPSA-EA, initially, the first generation of the population P is filled with μ random solutions. While the population is not full and there are processing nodes available, new random solutions are being created and sent to the simulation for evaluation. When μ solutions have been simulated, offspring generation is initiated. Offspring are created from parents in P chosen using crowding tournament selection (described in [2]). With tournament selection, solutions with worse fitness 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 is declared as the winner over B if either (*i*) A has a better rank than B, or (*ii*) A and B have the same rank, but A has a larger crowding distance than B.

When generating offspring, instead of creating only a single new solution from a pair of parents, which is the commonly used strategy in steady-state algorithms, a pool of λ candidate offspring is created. Such 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 optimizations [11], the size of the pool might 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. For each offspring, the objective errors of its parents are calculated by evaluating the parents using the surrogate and taking the difference between their assigned simulation objective values and the obtained surrogate objective values. Since the accuracy of the surrogate might change dynamically, surrogate values are calculated every time a solution is chosen as parent. The surrogate objective values of an offspring are modified by adding the weighted mean of the error values of the offspring's parents. The weighting is based on each parent's influence on the child during crossover. In case no crossover is applied when creating the child (i.e. only mutation is performed), the influence of one of the parents is 100%. The described method of considering surrogate imprecision automatically adapts to the quality of the surrogate. A larger error of the surrogate leads a higher degree of randomness in the offspring selection, and hence the less the risk that the search is misled by the surrogate evaluations. In the same way, the smaller the error of the surrogate, the more the surrogate will impact selections.

Based on the adjusted surrogate objective values, the most promising solution in O to insert into P is selected. In this procedure, all solutions of rank 1 in O are identified (called 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} dominating most solutions in P_{R1} is selected and simulated. 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 neighbor in P_{R1} is selected (note that crowding distance cannot be used since the solutions of O_{R1} and P_{R1} might be of different ranks if merged). 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 distances in the last rank. An elitistic approach is also possible, in which an offspring is only inserted into P if it is not dominated by all solutions in P. The simulation sample obtained from a newly inserted offspring may be used to update the surrogate. To save time, an update does not need to take place every time a new sample becomes available, but only every *N*:th sample.

When noise during evaluation of solutions is present, this adversely affects the parental selection process and the performance of the algorithm. If this is not considered, the algorithm can be mislead to propagate inferior parents. In the next section, we present a new technique for reducing noise to be used in the parental tournament selection in MOPSA-EA.

3. A MODIFIED EVOLUTIONARY SELECTION OPERATOR CONSIDERING NOISE

In real-world problems, the characteristics of the noise are unknown and re-sampling of solutions is therefore always necessary to form an estimate of the noise size. However, since sampling *n* times increases the computational time by a factor of *n*, it is important that the number of samplings is kept at a minimum. Many noise handling methods use an approach of re-sampling solutions a fixed number of times in each generation, which is inefficient from two perspectives: (*a*) the noise size might vary in the search space, and (*b*) simulations are allocated in a suboptimal way (efforts are wasted on inferior solutions, and solutions subject to less noise are sampled unnecessarily and those suffering from much noise are not sampled enough). For improved efficiency, we therefore introduce a technique that vary the number of samples used per solution based on the present noise size in combination with the required confidence level, representing how certain we want to be to choose the better of two solutions. Re-sampling of solutions is performed iteratively until the noise is sufficiently reduced. We refer to this technique as *confidence-based dynamic re-sampling*. Basically, the technique is implemented by using an iterative re-sampling procedure that continues sampling two solutions until the likelihood of selecting the correct solution reaches the given confidence level. Below, the procedure of the proposed confidence-based dynamic re-sampling technique is described in further detail:

Step 1: Initial sampling

Initially, the two solutions being compared are sampled two times each, which is the minimum number of samples needed to form an initial estimate of the present noise size.

Step 2: Calculation of mean and sample standard deviation

Based on the collected samples, the mean μ and sample standard deviation s of each objective of the two solutions are calculated. The sample standard deviation, measuring the variability in samples (i.e. the noise size), of objective i is calculated according to Equation 1

$$s_i = \sqrt{\frac{1}{N-1}\sum_{j=1}^{N}\left(x_j - \mu_i\right)^2} \qquad (1)$$

where N is the number of samples, and $x_1,...,x_N$ are the sample values.

Step 3: Selection of confidence level

The confidence level is defined by the user and represents the required certainty of the relation between two solutions. Between two solutions A and B, three types of relations are possible: (*i*) A dominates B, or (*ii*) B dominates A, or (*iii*) A and B are mutually non-dominating.

The confidence level α specifies that in at least α of the cases the selection between two solutions should be correct. One confidence level is defined for each Pareto-rank, and generally the higher the rank of a solution, the higher its confidence level. This is because a high precision is usually more important for solutions in, or nearby, the Pareto-front. The confidence level to use in a comparison of two solutions is derived from the one with highest rank, which means that a non-dominating sort must be performed in this step to derive the ranks of the solutions.

In this study, the following confidence levels are being defined:

Rank 1: 0.75

Rank 2: 0.70

Rank 3: 0.65

Rank 4: 0.60

Rank 5 and higher: 0.55

Step 4: Confidence test

In a noisy context, the true relation between two solutions is only possible to determine by taking the mean of all possible samples of the solutions. In reality, however, it is not possible to collect the complete set of samples, but only a limited number of samplings can be performed. Instead, the probability that the solutions' relation is the same given the collected samples as given all samples has to be established (i.e. the probability of making a correct selection between the solutions). In doing this, the method of Welch Confidence Interval (WCI) is being used. WCI can be used in comparing whether or not there is a significant difference between two solutions of unknown and possibly unequal variances with respect to a given confidence level.

WCI values are calculated for each objective i according to Equation 2 [12]

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

where A and B are the two solutions being compared, μ is the mean of objective i, N is the number of samplings of a solution, s^2 is the variance of objective i (Equation 1), and f is the estimated degree of freedom (Equation 3), α is the confidence level, and M is the number of objectives. In a multi-objective problem, the confidence level α has to be divided by 2 M (and not by 1, as normally done) due to the Bonferroni Inequality [12]. This means that for a problem of two objectives, to obtain a 0.95 confidence each objective has to be compared with a confidence level of 0.975.

$$f = \frac{\left[\frac{s_A^2}{N_A} + \frac{s_B^2}{N_B}\right]}{\frac{\frac{s_A^2}{N_A}}{N_A - 1} + \frac{\frac{s_B^2}{N_B}}{N_B - 1}} \qquad (3)$$

If the WCI resulting from Equation 3 does not cover 0, there is a significant difference between the two solutions in the *i*:th objective. If none of the WCIs for the *M* objectives cover 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, i.e. if any of the objectives' intervals cover 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 re-sampling

When the given confidence level is not reached, additional re-sampling is required. Ultimately, the confidence level should be reached using as few re-samplings as possible to save resources. Therefore, the strategy adopted in this step is to resample only one of the solutions, and select the one having the largest sample standard deviation in the objective with the largest overlapping intervals (i.e. with the largest potential to eliminate the undesired overlap). When a re-sampling has been performed, the procedure is repeated from step 2 and a new check is made if yet another sampling is necessary.

To prevent two solutions that are close to each other in objective space to be re-sampled forever, the number of times a solution can be sampled 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 is usually allowed for solutions in higher ranks where a higher precision is needed. The 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 re-sampled instead, unless it has also reached its maximum number. In such case the dominating solution (or the one with largest crowding distance, if the solutions are mutually non-dominating) is returned and the procedure is terminated. In this study, the following maximum number of samplings is being defined:

Rank 1: 5 sampling

Rank 2: 4 sampling

Rank 3 and higher: 3 samplings

Since the ranks of solutions is calculated in every iteration of the procedure, the maximum number of samplings of solutions, as well as the confidence level to use when comparing them, may change between iterations. In this way, the re-sampling strategy becomes dynamic and automatically adjusted to the current situation.

In the following sections, an evaluation of the confidence-based dynamic re-sampling technique is presented when integrated in MOPSA-EA When MOPSA-EA is implemented to handle noise using this technique, the algorithm is called "N-MOPSA-EA".

4. OPTIMIZATION PROBLEMS

Three optimization problems are used to assess the performance of N-MOSPA-EA; one theoretical benchmark problem and two real-world problems from the manufacturing domain.

Problem 1: Benchmark problem

The function "ZDT1", described in Table 1, is a multi-objective benchmark problem used in many research articles. In this study, artificial noise is added to the function generated from a Gaussian distribution with $\mu = 0$ and σ representing the noise size. Three different noise sizes are being tested: 0.10, 0.15, and 0.20.

Table 1: ZDT1 benchmark function

No. of inputs	Input bounds	Function	Optimal solution	Shape
30	[0,1]	$f_1(x) = x_i$ $f_2(x) = g(x)\left[1 - \sqrt{x_1 / g(x)}\right]$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right) / (n-1)$	$x \in [0,1]$ $x_i = 0$ $i = 2,...,n$	Convex Pareto-front

Problem 2: Engine Component Manufacturing

The first real-world problem considered concerns optimal production planning at a Volvo Aero factory in Sweden. The factory produces engine components to civilian and military airplanes, as well as to space rockets. The focus of the study is a manufacturing cell that processes a wide range of different engine components. The inflow of the cell is controlled by using fixed inter-arrival times of components. The inter-arrival time does not only specify when a component should enter in the system, but it also determines the component's due date since an overall production strategy is to process no more than one component of a specific type in the cell at a time. For an efficient production, the inter-arrival times should be specified in a way that maximizes the utilization of the cell and simultaneously minimizes overdue components (i.e. tardiness). For a high utilization, short inter-arrival times are needed in order to obtain a high load of the cell and thereby avoid machine starvation. However, avoiding overdue components requires generous due dates; that is long inter-arrival times. This means that the two objectives of maximal utilization and minimal tardiness are conflicting with each other.

Problem 3: Camshaft Machining Line

The second real-world problem considered is a camshaft machining line at Volvo Cars in Sweden. The line is responsible for producing about 15 different camshaft variants. The machining line consists of 14 different machine groups with one to seven parallel machines in each group; totally 34 machines. 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. The machining line is semi-automated with robots that feed machines inside cells, while the loading and unloading of camshafts are performed by operators.

The processing to be performed in the line is given by a schedule, defining an internal priority order among batches of different product variants and the individual path through machines and operations these batches should take. To be able to handle demand fluctuations and unexpected events such as machine breakdowns or quality defects, it is important that the schedule is defined in a way that makes sure that the number of different camshaft variants in the finished goods stock is above the specified minimum levels. Stock levels are check 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 minimizing shortage, it is also important that schedule results in as high throughput of the line is as possible for maximum efficiency. For high throughput, there should ideally be only one variant produced in the line at the same time to avoid set-up times of machines, which cause a large overhead. Further, for a high throughput, variants with shorter processing times should be prioritized 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 prioritized in a way that avoids shortage. In other words, the objectives of minimum shortage and maximum throughput are conflicting.

5. EVALUATION

In assessing the performance of N-MOPSA-EA a number of evaluation metrics are being used, which are described in the next section. The configuration of the surrogates adopted in the optimization problems is presented in Section 5.2. In Section 5.3, three existing surrogate-assisted multi-objective algorithms used for comparison are outlined.

5.1 Evaluation metrics

An overall goal in multi-objective optimization is convergence to the Pareto-front. A commonly used measure for evaluating convergence for problems having a known true optimal front (which is the case with the ZDT function) is the Υ metric ([13]. 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. The Υ metric is used in assessing the performance in the ZDT1 problem (it cannot be used in the other two problems since the true Pareto front of these problems is unknown, as with real-world problems in general). In the ZDT1 problem, the Ω metric is also used (described in [6]). This metric is a combined measure of convergence and diversity in the in the set of non-dominated solutions. Ω 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 lower the value of Ω, the better the results of the algorithm. Another performance metric that combines both convergence and diversity is the $\mathcal{S}$ metric (also called the hyper-volume metric). Basically, $\mathcal{S}$ measures the volume in objective space dominated by obtained solutions [14]. The larger the volume, the better the results of the algorithm. The $\mathcal{S}$ metric does not assume that the true Pareto-optimal front is known and can therefore also be applied to real-world problems.

In the evaluation of N-MOPSA-EA, all three performance metrics described (Υ, Ω, and $\mathcal{S}$) are being used in the benchmark problem, while only the last metric ($\mathcal{S}$) is being used in the real-world problems.

5.2 Surrogates

MOPSA-EA allows for any kind of surrogates, and in this paper two different surrogate techniques are being used. In the first two problems, Artificial Neural Networks (ANNs) are being used. ANNs have been considered being appropriate for approximation of complex problems with limited number of data samples [15]. The ANN adopted has a feed-forward architecture with one hidden layer. The ANN is trained using backpropagation with a learning rate of 0.5. For each 10th simulation, the ANN is re-trained with the most recent samples (at most 50). 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 have been preferred over global ANNs in surrogate-assisted optimization, mainly because they reduce the time-consumption of the training process [3]. To avoid overfittning, 10-folded cross-validation is used in the training. The number of hidden nodes of the ANN is dynamically adapted to the number of samples available. For a good performance of an ANN, it is recommended that the number of weights of the network is proportional to the size of the training data set [16]. Since the number of samples continuously increase during the optimization, a static number of hidden nodes is not appropriate. Therefore, the optimization starts with an ANN of one single node, and additional hidden nodes are successively being added. When the number of samples available exceeds five times the number of weights in the network, a new hidden node is added (according to the weight-sample ratio suggested in [16]).

In the third problem, constructing a useful ANN is not possible since the number of simulation inputs is very large (>500). An ANN with over 500 inputs involve tens of thousands of network weights, or even more, and such 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 in this 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.

5.3 Performance comparison

To assess the relative performance of N-MOPSA-EA, it is compared to the original MOPSA-EA and to three existing surrogate-assisted multi-objective algorithms, namely "Metamodel-Assisted $\mathcal{S}$ Metric Selection Evolutionary Multi-Objective Algorithm" (SMS-EMOA) [14], " $(\mu+\nu<\lambda)$ Metamodel-Assisted Evolution Strategy (MAES)" incorporated into NSGA-II [11], and NSGA-II integrated with an ANN (NSGA-II-ANN) [17].

SMS-EMOA is a steady-state algorithm that uses the $\mathcal{S}$ metric as selection criterion, both for offspring selection and replacement selection in the population. In both selections, a non-dominated sort takes place and the solution contributing most (in the former) or least (in the latter) to the hyper-volume of the population is selected.

MAES also uses the $\mathcal{S}$ metric for selections, but is based on a generational approach. From the population of μ solutions, λ offspring are generated and evaluated using the surrogate. Out of the λ offspring, the ν solutions contributing most to the hyper-volume of the population is selected and simulated. The next generation of the population is then formed from the combined set of μ parents and ν offspring.

NSGA-II-ANN works like the standard NSGA-II (see [13]), except that a simulation and an ANN is used alternately to evaluate generations. In every cycle of *m* generations, the simulation is first used to evaluate *n* of the generations and the surrogate is then used to evaluate the remaining *m-n* generations. A new surrogate is constructed in every cycle based on the last *n* simulation samples. Similar to MOPSA-EA, the idea is to adopt local surrogates defined over a small search region.

For a fair performance comparison of the five algorithms used in the evaluation, all algorithms start the optimization from the same initial population, use the same surrogate configuration, and have the same parameter settings (Table 2). Note, however, that NSGA-II-ANN does not make use of offspring candidates and therefore the parameter "number of offspring" does not apply to this algorithm. Instead, this algorithm uses the parameters *m* and *n*, which are set to 13 and 3 (respectively), according to the recommendations in [13].

Table 2: Algorithm Parameter Settings

	Problem 1	Problem 2	Problem 3
Population size	50	40	60
No. of offspring	25	20	30
Mutation step size	0.5	1.0	1.0
Crossover	Single-point	Single-point	Single-point
Crossover prob.	0.8	0.8	0.8

6. RESULTS

This section presents the results of the three optimization problems. In MOPSA-EA, SMS-EMOA, MAES and NSGA-II-ANN, two different strategies of sampling solutions are being tested: (*i*) one sampling of each solution (i.e. no noise reduction), and (*ii*) five samplings of each solution. With both strategies, the total number of simulation replications used in an optimization is the same. Consequently, the number of unique solutions evaluated will be different in the two strategies; given *n* simulation replications and *s* samplings, *n*/*s* unique solutions are being evaluated.

Problem 1: Benchmark problem

Results from the ZDT1 function are shown in Table 3-5. The optimization is performed for 5000 function evaluations and the results presented are an average of 500 replications. All results have a confidence probability of 0.99 or more, calculated using Welch's t-test (defined in [12]). Note that a low value of Υ and Ω, and a high value of $\mathcal{S}$ is desirable. In calculating the first two metrics, a set of 500 uniformly-distributed solutions of the true Pareto front is derived. When calculating the performance metrics, the objective values *without* noise is used.

As shown from the tables, MOPSA-EA achieves the best results with noise size is 0.1, while N-MOPSA-EA achieves the best results with size 0.15 and size 0.2. This indicates that the original algorithm could be used when there is little noise present, and that N-MOPSA-EA is most efficient with more substantial noise.

MOPSA-EA, SMS-EMOA, MAES and NSGA-II-ANN all obtain better results when only performing one sampling compared to five. These results may indicate that the noise sizes used represent a relatively small amount of noise. This theory is supported by the fact that the benefit of performing replications generally seems to increase with the noise size.

Table 3: Results ZDT1 Noise Size 0.1

	Υ		Ω		$\mathcal{S}$	
N-MOPSA-EA	0.887		0.825		0.84	
Samplings	*1*	*5*	*1*	*5*	*1*	*5*
MOPSA-EA	0.815	1.349	0.781	0.977	0.842	0.8
SMS-EMOA	2.052	2.278	1.577	1.931	0.708	0.657
MAES	1.094	1.932	1.065	1.823	0.794	0.696
NSGA-II-ANN	0.98	2.226	1.089	1.882	0.809	0.644

Table 4: Results ZDT1 Noise Size 0.15

	Υ		Ω		$\mathcal{S}$	
N-MOPSA-EA	0.997		0.845		0.815	
Samplings	*1*	*5*	*1*	*5*	*1*	*5*
MOPSA-EA	1.001	1.455	1.22	1.249	0.782	0.752
SMS-EMOA	2.146	2.358	1.822	2.128	0.675	0.626
MAES	1.462	2.051	1.15	1.833	0.76	0.669
NSGA-II-ANN	1.285	2.092	1.24	1.829	0.761	0.664

Table 5: Results ZDT1 Noise Size 0.2

	Υ		Ω		$\mathcal{S}$	
N-MOPSA-EA	1.147		0.943		0.802	
Samplings	*1*	*5*	*1*	*5*	*1*	*5*
MOPSA-EA	1.228	1.545	1.366	1.498	0.778	0.748
SMS-EMOA	2.219	2.427	1.677	1.919	0.702	0.664
MAES	1.694	2.206	1.477	1.992	0.727	0.681
NSGA-II-ANN	1.319	2.101	1.302	1.852	0.757	0.673

Problem 2: Engine Component Manufacturing

Results of the first real-world optimization problem are presented in Table 6 (average of 10 replications). The optimization is performed for 400 simulations. The results presented have a confidence probability of at least 0.8 (calculated using Welch's t-test). Unlike the theoretical ZDT1 problem, the true objective values cannot be derived in this problems to be used in calculating the performance of the algorithms. Instead, the final Pareto-front found by an algorithm are being replicated 20 times and the mean values of the simulation replications are used when calculating the $\mathcal{S}$ metric (note that all solutions in the front found by the algorithm might not end up in the Pareto-front when the mean values are considered).

As shown in Table 6, N-MOPSA-EA achieves the best results and SMS-EMOA the worst. In difference with the ZDT1 problem, all algorithms benefit from performing multiple samplings of solutions, which might indicate that the noise is stronger and has a more complicated nature in this problem compared to the artificial noise added to the ZDT1 function.

Table 6: Results Engine Component Manufacturing

	$\mathcal{S}$	
N-MOPSA-EA	0.496	
Samplings	*1*	*5*
MOPSA-EA-EA	0.465	0.467
SMS-EMOA	0.374	0.398
MAES	0.426	0.451
NSGA-II-ANN	0.446	0.452

Problem 3: Camshaft Machining Line

In this problem, the optimization is performed for 600 simulations and the results of the different algorithms are presented in Table 7. Similar to the previous problem, the $\mathcal{S}$ metric is calculated by replicating the final Pareto-front found by the algorithm 20 times and taking the mean values of the replications. The confidence probability of the results is at least 0.84 (calculated using Welch's t-test). As shown from the results, the algorithms rank in the same order as in the previous problem. Also in this problem all algorithms benefit from performing multiple samplings of solutions.

Table 7: Results Engine Component Manufacturing

	$\mathcal{S}$	
N-MOPSA-EA	0.511	
Samplings	*1*	*5*
MOPSA-EA	0.481	0.483
SMS-EMOA	0.408	0.44
MAES	0.441	0.469
NSGA-II-ANN	0.471	0.479

7. CONCLUSIONS

In this paper, a new technique that efficiently deals with the negative effects of noise in simulations is presented. Basically, this technique uses an iterative re-sampling procedure that reduces the noise until the likelihood of selecting the correct solution reaches a given confidence level. The proposed noise compensation technique can be used with any EA and can be applied in all evolutionary selections without modifications. There are no limits in the number of objectives that can be handled, and it can be used on single-objective problems as well. While several existing noise compensation approaches assume that all solutions are equally perturbed by noise and/or that the noise characteristics are known at before hand (e.g. [18-20]), the proposed technique does not make any assumption on the noise landscape. For improved efficiency, the technique automatically adapts the number of samplings to the present noise size; solutions subject to much noise are sampled a larger number of times compared to those subject to less noise. The number of samplings are also automatically adjusted according to the importance of the particular solution, as solutions of higher ranks are generally sampled a larger number of times compared to those of lower ranks.

A drawback of the proposed technique is that it involves two user-defined parameters; confidence level and maximum number of samplings. Finding the optimal configuration of these parameters for the problem at hand is a task of trial-and-error. The same drawback of parameters that must be specified by the user also applies to other noise compensation techniques (e.g. [19][21-22]). Although we have shown that the proposed technique works very well with a standard setting of the parameters (c.f. Section 3), ideally there should be no user-defined parameters at all. Investigating efficient noise compensation techniques that are free from user-defined parameters is an important topic for our future research.

ACKNOWLEDGMENTS
This work has been carried out within the OPTIMIST project which is partially financed by the Knowledge Foundation (KK Stiftelsen), Sweden. The authors gratefully acknowledge the Knowledge Foundation for the provision of research funding and also the industrial partner companies Volvo Aero and Volvo Cars for their support in this study.

8. TEXT STYLES

[1] 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, 80-86.

[2] Deb, K. 2004. Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons Ltd.

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

[4] Bui, L.T., Abbass, H.A. and Essam, D. 2005. Fitness inheritance for noisy evolutionary multi-objective optimization. In Proceedings of Conference on Genetic and Evolutionary Computation, 779-785, Washington DC, USA.

[5] Gosavi, A. 2003. Simulation-based optimization: para-metric optimization techniques and reinforcement learning. Boston, Mass., Kluwer Academic.

[6] Syberfeldt, A., Grimm, H., Ng, A. and John, R.I. 2008. A Parallel Surrogate-Assisted Multi-Objective Evolutionary Algorithm for Computationally Expensive Optimization Problems. In Proceedings of the 2008 IEEE Congress on Evolutionary Computation, Hong Kong, June 1-6.

[7] Ong, Y.S., Nair, P.B., Keane, A.J. and Wong, K.W. 2004. Surrogate-assisted evolutionary optimization frame-works for high-fidelity engineering design problems. Knowledge Incorporation in Evolutionary Computa-tion, Springer Verlag, 307-332.

[8] Ulmer, H., Streichert, F. and Zell, A. 2003. Evolution strategies assisted by gaussian processes with improved pre-selection criterion. In Proceedings of IEEE Congress on Evolutionary Computation, 692-699, Canberra, Australia, December 8-12, 2003.

[9] Streichert, F., Ulmer, H., and Zell, A. 2005. Parallelization of multi-objective evolutionary algorithms using clustering algorithms. In Proceedings of Third Inter-national Conference on Evolutionary Multi-Criterion Optimization (EMO05), 92-107.

[10] Boesel, J., Bowden, R.O., Glover, J.P., Kelly, F. and Westwig, E. 2001. Future of simulation optimization. In Proceedings of Winter Simulation Conference 2001, 1466-1470, Arlington, VA, USA, December 9-12.

[11] Emmerich, M., Giannakoglou, K. and Naujoks, B. 2006. Single- and multi-objective evolutionary optimization assisted by gaussian random field metamodels. IEEE-TEC Special Issue on Evolutionary Computation in the presence of uncertainty, 10(4): 421-439.

[12] Law A.M. and Kelton, D. 200. Simulation Modeling and Analysis, Mc Graw Hill, 3rd edition.

[13] Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. 2000. A fast and elitist multi-objective genetic algorithm-NSGA-II. KanGAL Report 2000001, Indian Institute of Technology Kanpur.

[14] Emmerich, M. Beume, N. and Naujoks, B. 2005. An EMO algorithm using the hypervolume measure as selection criterion. In Proceedings of Evolutionary Multi-Criterion Optimization, Springer Berlin-Heidelberg, 62-76.

[15] Jin, Y. 2005. A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing, 9: 3-12, Springer-Verlag Germany.

[16] Mehrotra, K. Mohan, C.K. and Ranka, S. 1996. Elements of Artificial Neural Networks, MIT Press.

[17] Nain P.K.S. and Deb, K. 2005. A multi-objective optimization procedure with successive approximate models. KanGAL report no. 2005002, Indian Institute of Technology, Kanpur, India.

[18] Beyer, H.G. 2000. Evolutionary Algorithms in Noisy Environments: Theoretical Issues and Guidelines for Practice. Computer Methods in Applied Mechanics and Engineering 186(2-4): 239-267.

[19] Hughes, E. Evolutionary Multi-objective Ranking with Uncertainty and Noise. 2001. In Proceedings of First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001), Springer Berlin-Heidelberg, 329-343.

[20] Arnold, D. V. Noisy Optimization with Evolution Strategies. 2002. Kluwer Academic Publishers, Genetic Algorithms and Evolutionary Computation Series.

[21] Babbar, M., Lakshmikantha, A. and Goldberg, D. E. 2003. A modified NSGA-II to solve noisy multiobjective problems. In Proceedings of Genetic and Evolutionary Computation Conference (GECCO2003), Springer Verlag, 21-27.

[22] 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: 460-473.