A Parallel Surrogate-Assisted Multi-Objective Evolutionary Algorithm for Computationally Expensive Optimization Problems

Anna Syberfeldt, Henrik Grimm, Amos Ng, and Robert I. John

***Abstract*— This paper presents a new efficient multi-objective evolutionary algorithm for solving computationally-intensive optimization problems. To support a high degree of parallelism, the algorithm is based on a steady-state design. For improved efficiency the algorithm utilizes a surrogate to identify promising candidate solutions and filter out poor ones. To handle the uncertainties associated with the approximative surrogate evaluations, a new method for multi-objective optimization is described which is generally applicable to all surrogate techniques. In this method, basically, surrogate objective values assigned to offspring are adjusted to consider the error of the surrogate. The algorithm is evaluated on the ZDT benchmark functions and on a real-world problem of manufacturing optimization. In assessing the performance of the algorithm, a new performance metric is suggested that combines convergence and diversity into one single measure. Results from both the benchmark experiments and the real-world test case indicate the potential of the proposed algorithm.**

I. Introduction

EVOLUTIONARY algorithms (EAs) are powerful techniques for solving complex optimization problems [1]. While traditional analytical optimization methods have been unable to cope with the challenges imposed by many real-world systems, such as multimodality, non-separability and high dimensionality, EAs have proven to be highly useful in searching reasonably good solutions. During the years, EAs have shown to produce convincing results for a wide variety of problems such as engineering design, operational planning, and scheduling.

When applying EAs in real-world scenarios, the simultaneous optimization of more than one objective is often required. The difficulty with problems of multiple objectives is that there is usually no single optimal solution with respect to all objectives, as improving the performance on one objective would deteriorate the performance of one or more of the other objectives. Instead of a single optimum, there is a set of optimal trade-offs, known as the Pareto set. For deriving the Pareto set, most multi-objective optimization algorithms use the concept of dominance [1]. A solution *x*' is said to dominate another solution *x*'' if *x*' is no worse than *x*'' in all objectives, and *x*' is strictly better than *x*'' in at least one objective. The solutions that are not dominated by any other solutions are called Pareto-optimal and form the Pareto set. Solutions can also be sorted into different non-dominated ranks; rank 1 is made up of the Pareto-optimal solutions among the complete set of solutions, rank 2 of the Pareto-optimal solutions identified when temporarily discarding all solutions associated with rank 1, etc. Since EAs maintain a population of solutions, it is possible to capture multiple Pareto-optimal solutions in one single optimization run. This makes EAs very suitable for the handling of multi-objective problems [1].

Although EAs have achieved great success in many applications, these algorithms have also encountered some technical hurdles. Among these, efficiency is a major challenge. Real-world optimization problems often involve an immense number of possible solutions, and an EA needs a large number of simulation evaluations before an acceptable solution can be found [2]-[3]. Even with improvements in computer processing speed, one single simulation evaluation may take a couple of minutes to hours or days of computing time [4]-[5]. This poses a serious hindrance to the practical application of EAs in real-world scenarios, and to tackle this problem various approaches have been suggested. A commonly used technique to address the problem of computationally expensive simulations is parallelization. With parallel processing nodes, several solutions can be effectively simulated simultaneously. Another technique for increased efficiency is the incorporation of computationally efficient surrogates (also called metamodels). A surrogate is an approximation of the simulation; if the simulation is represented as $y = f(x)$, then a surrogate is represented as $\hat{y} = \hat{f}(x)$, such that $\hat{f}(x) = f(x) + e(x)$, where $e(x)$ is the approximation error. For constructing surrogates, a variety of different techniques have been proposed, among the most popular in evolutionary optimization being Artificial Neural Networks, Radial Basis Function Networks, and Kriging models [6]. The application of surrogates to EAs is, however, not completely straightforward. Constructing a surrogate that represents the complete search space correctly is hard, especially in real-world problems with sparse data samples [6]. If not handled properly, the error of the surrogate may mislead the EA to propagate inferior individuals; weak offspring might be chosen for the next generation while good ones might be excluded. For successful results, the imprecision of the surrogate must

Manuscript received March 1, 2008. This work was supported in part by the Knowledge Foundation, Sweden and Volvo Aero, Sweden.

A. Syberfeldt, H. Grimm, and A. Ng are with the Centre for Intelligent Automation, University of Skövde, Skövde, PO 54148 Sweden (corresponding author to provide phone: 46-500-448577; fax: 46-500-448598; e-mail: {anna.syberfeldt, henrik.grimm, amos.ng}@his.se).

R. I. John is with the Centre for Computational Intelligence, De Montfort University, Leicester LE1 9BH, UK (e-mail: rij@dmu.ac.uk).

therefore be considered in the optimization; otherwise it is very likely that the search would converge to a false optimum [3].

In this paper, a new multi-objective EA that incorporates a surrogate for solving computationally-intensive optimization problems is described. This algorithm is based on a steady-state design to support a high degree of parallelism. Most multi-objective EAs use a generational approach [5], which is not optimal with respect to parallel efficiency. In generational algorithms, results for a complete generation must be awaited in order for the search to proceed. This is inefficient if the population size is not divisible by the number of processing nodes, or if simulations on different nodes take different amounts of time. Furthermore, if the population size is less than the number of processing nodes, all computing resources will not be utilized. In comparison, a steady-state design enables a higher degree of parallelism, since new solutions are continuously created and the number of parallel evaluations is not limited by the population size. Besides their parallel efficiency, steady-state algorithms have also been considered being more efficient on complex optimizations problems and are able to find good solutions in less time compared to generational algorithms [7].

In the proposed algorithm, the surrogate is used to screen candidate solutions and identify the most promising ones. 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 selection of the offspring to include in the population, the imprecision associated with the surrogate is considered using a new approach for multi-objective optimization.

In the next section, the fundamental design of the proposed Multi-Objective Parallel Surrogate-Assisted EA, called MOPSA-EA, is presented.

II. Algorithm Description

The basic algorithm is described with pseudo code in Fig. 1. 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 [1]). 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.

In the algorithm, a slightly optimized implementation of the standard crowding tournament selection operator is used in which a pre-control is made if A dominates B. By first verifying if a dominance relationship exists between the two solutions, an unnecessary non-dominated sort can be avoided.

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 [8], 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 (the details of this procedure are described in the next section). 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. Since the algorithm is neutral with respect to surrogate modeling technique, it does not specify *how* to update the surrogate. Surrogate update strategies vary between different techniques, and are also highly problem-dependent.

```
function Main( )
  P ← ∅
  iter ← 0
  while (not StopOptimization( )) do
    while (ProcessingNodeAvailable( )) do
      if (|P| = μ) then
        BeginSimulation(GenerateNewSolution(P))
      else
        BeginSimulation(GenerateRandomSolution( ))
      endif
    endwhile
    p ← WaitForFinishedSimulation( )
    if (Mod(iter, surrogateUpdateFrequencey) = 0) then
      surrogate.Update( )
    endif
    if (|P| = μ) then
      P.Remove(Select Worst(P))
    endif
    P.Add(p)
    iter ← iter + 1
  endwhile

function GenerateNewSolution(P)
  O ← ∅
  repeat λ times
    parent1 ← SelectForReproduction(P)
    parent2 ← SelectForReproduction(P)
    o ← Crossover(parent1, parent2)
    Mutate(o)
    SurrogateEvaluation(o)
    O.Add(o)
  end
  return Select Best(O)
```

Fig: 1. Multi-Objective Parallel Surrogate-Assisted Evolutionary Algorithm (MOPSA-EA).

A. Offspring Selection with Consideration to Surrogate Imprecision

A new method for handling surrogate imprecision is used in the offspring selection procedure in MOPSA-EA. Basically, surrogate objective values assigned to offspring are adjusted to consider the 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. For example, if the simulation values assigned to a parent are $(51,63)$ and a surrogate evaluation returns the values $(45,77)$, then the error of that parent is $(51,63)-(45,77)=(6,-14)$. 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 (Fig. 2). 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%.

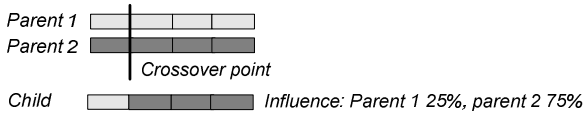

Fig: 2. Example of parents' influence on child.

The adjustment of surrogate objective values can be illustrated by an example: if an offspring is assigned the values $(50,81)$ from the surrogate and the error values of its parents are $(6,-14)$ and $(8,2)$, respectively, then the new objective values of the offspring become $50+0.25*6+0.75*8=56.5$ and $81+0.75*-14+0.25*2=69$ (assuming that parents' influence is 25% and 75%, respectively).

This approach of obtaining offspring error values has similarities with the concept of fitness inheritance, in which an offspring's objective values are based on its parents' objective values, and not on a simulation. In multi-objective optimization, fitness inheritance has shown to work well (see for example [9]-[11]) and this has motivated the use of the concept in this work.

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. In comparison with existing methods that augment uncertainty information into surrogate evaluations, such as Lower Confidence Bound, Probability of Improvement and Expected Improvement (all described in [12]), advantages of the proposed method include:

- Simple to understand and to implement
- Requires no user-defined parameters
- Does not include any expensive computations (i.e. integrals)
- Can handle an arbitrary number of objectives without performance degradation
- Independent of the surrogate modeling technique

III. BENCHMARK OPTIMIZATION

A set of guidelines about systematic development of test problems for multi-objective optimization was first proposed in [15]. Based on these guidelines, [16] suggest six benchmark functions that have been used extensively in the literature for the analysis and comparison of multi-objective

EAs: ZDT1, ZDT2, ZDT3, ZDT4, ZDT5, and ZDT6. The features of these problems represent aspects that are known to cause difficulties in converging to the true Pareto-optimal front, and they reflect properties of real-world problems (such as multimodality and non-separability). This has motivated the use of these functions in assessing the performance of MOPSA-EA. However, function ZDT5 has been omitted since it defines a Boolean function over binary-strings, and such binary encoded solutions are not relevant in this study.

The metrics used to assess the performance of MOPSA-EA on the benchmark functions are described in the next section. The configuration of the surrogate and the algorithm parameters settings adopted in the evaluation are presented in Section B and C, respectively. In Section D, three existing surrogate-assisted multi-objective algorithms used for comparison are outlined, followed by a presentation of results in Section E.

A. Evaluation Metrics

In multi-objective optimization, there are two overall goals: convergence to the Pareto-optimal front, and maximal diversity among Pareto-optimal solutions. Two commonly used measures for evaluating convergence and diversity, respectively, for problems having a known true Pareto-optimal front are the Y and Δ metrics [13]. The Y metric measures the degree of convergence by calculating 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 Y, the better the convergence of the algorithm. The Δ metric measures the spread among solutions in the obtained non-dominated set using the following formula:

$$\frac{d_f + d_l + \sum_{i=1}^{N} |d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}}.$$

In this formula, d_f and d_l are the Euclidean distances between the extreme solutions of the true Pareto front and the boundary solutions of the obtained non-dominated set, and $\bar{d}$ is the average of all distances d_i in the obtained non-dominated set ($N-1$ distances with N solutions). The smaller the value of Δ, the better the spread of solutions.

A problem of using separate metrics for convergence and diversity, as done with the Y and Δ metrics, is that a comparative evaluation of two algorithms might not give an answer about which of the algorithms is superior. While the first algorithm may have a low Y-value and a high Δ-value, the inverse may apply for the second algorithm. To address this problem, we suggest a new performance metric for problems having a known true Pareto front that combines convergence and diversity into a single measure. This metric, called 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 (Fig. 3). The rationale behind this metric is that for a low value of Ω, both a well spread front and a good convergence is needed.

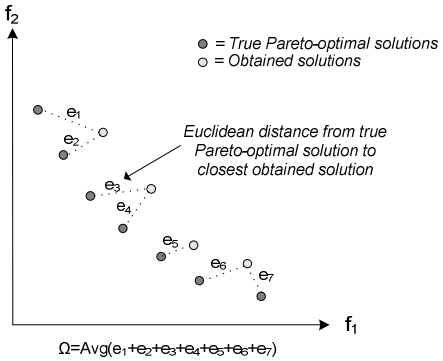

Fig. 3. Ω performance metric.

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 (Fig. 4) [17]. The larger the volume, the better the results of the algorithm.

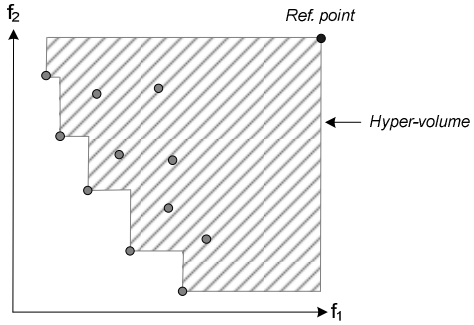

Fig. 4. $\mathcal{S}$ performance metric.

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. A potential problem of $\mathcal{S}$ is, however, that the measure is biased towards a diagonal front. This problem is illustrated in Fig 5. The solutions A and B both contribute to the Pareto front, but B is valued much higher than A since it contributes more to the hyper-volume.

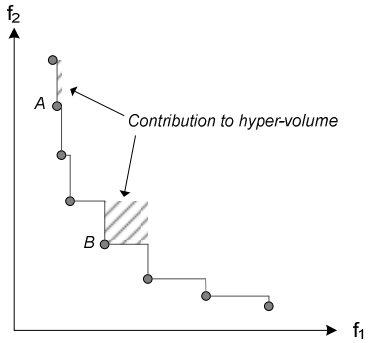

Fig. 5. Illustration of problem with $\mathcal{S}$ metric.

In the benchmark evaluation of MOPSA-EA, all four performance metrics described (Y, Δ, Ω, and $\mathcal{S}$) are being used for performance assessment.

B. Surrogate

MOPSA-EA allows for any kind of surrogates, and in this paper Artificial Neural Networks (ANNs) are being used since ANNs have been considered being appropriate for approximation of complex problems with limited number of data samples [6], [18]. The ANN adopted has a feed-forward architecture with one hidden layer. It has been shown (e.g. [19]) that one hidden layer is sufficient for nearly all problems. The ANN is trained using back-propagation with a learning rate of 0.5. For each 10^{th} 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 [20], mainly because they reduce the time-consumption of the training process [21]. 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 [22]. 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 [22]).

C. Algorithm Parameter Settings

The performance of an EA is highly dependent on the proper settings of its parameters, which are problem-dependent and usually must be obtained through trial-and-error experiments. Therefore, the parameter values of MOPSA-EA are tuned before the algorithm is actually being evaluated. Three different settings are being tested for each parameter:

- Population size: 20, 60, and 100.
- Number of offspring: 20, 60, and 100.
- Mutation step size: 0.5, 1.0, and 1.5.
- Crossover: Single-point (SP), blend, and uniform.
- Crossover probability: 0.4, 0.6, and 0.8.

All combinations of the different settings are tested for all parameters, which mean that in total 3^5 experiments are performed for each of the five benchmark functions. Each experiment is replicated 100 times and the average values of the Y, Δ, Ω, and $\mathcal{S}$ metrics are taken as the result. In finding the optimal parameter settings for a function, metric values from all experiments are collected and ranked according to their achieved values. By taking the sum of the achieved ranks of each metric, the best setting is identified. The optimal settings found for the five functions are presented in Table I.

TABLE I
OPTIMAL PARAMETER SETTINGS FOR MOPSA-EA

	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
Population size	20	20	60	20	20
No. of offspring	60	100	60	100	60
Mutation step size	0.5	0.5	0.5	0.5	1.0
Crossover	SP	SP	SP	SP	SP
Crossover prob.	0.8	0.6	0.8	0.8	0.8

D. Performance Comparison

To assess the relative performance of MOPSA-EA, it is compared to three existing surrogate-assisted multi-objective algorithms, namely "Metamodel-Assisted $\mathcal{S}$ Metric Selection Evolutionary Multi-Objective Algorithm" (SMS-EMOA) [17], "$(\mu+\nu<\lambda)$ Metamodel-Assisted Evolution Strategy (MAES)" incorporated into NSGA-II [12], and NSGA-II integrated with an ANN (NSGA-II-ANN) [23].

SMS-EMOA (described with pseudo code in Fig. 6) 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.

```
P ← Create( )
Simulate(P)
while (not Stop_Optimization( )) do
  O ← ∅
  parent1 ← Select(P)
  parent2 ← Select(P)
  repeat λ times
    o ← Crossover(parent1, parent2)
    o ← Mutate( )
    O.Add(o)
  end
  SurrogateEvaluation(O)
  q ← SelectBest(O)
  Simulate(q)
  P.Add(q)
  P.Remove(SelectWorst(P))
endwhile
```

Fig. 6. Metamodel-Assisted $\mathcal{S}$ Metric Selection Evolutionary Multi-Objective Algorithm (SMS-EMOA).

MAES (described with pseudo code in Fig. 7) 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.

```
P ← Create( )
Simulate(P)
while (not Stop_Optimization( )) do
  O ← ∅
  repeat λ times
    o ← Crossover(Select(P),Select(P))
    o ← Mutate( )
    O.Add(o)
  end
  SurrogateEvaluation(O)
  Q ← Select Best(O)
  Simulate(Q)
  P ← Select(P ∪ Q)
endwhile
```

Fig. 7. $(\mu+\nu<\lambda)$ Metamodel-Assisted Evolution Strategy (MAES).

NSGA-II-ANN (described with pseudo code in Fig. 8) works like the standard NSGA-II, 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.

```
P ← Create( )
Simulate(P)
numEvaluations ← 0
simulate ← true
while (not Stop_Optimization( )) do
  O ← ∅
  repeat λ times
    o ← Crossover(Select(P),Select(P))
    o ← Mutate( )
    O.Add(o)
  end
  if simulate
    Simulate(O)
  else
    SurrogateEvaluation(O)
  endif
  numEvaluations ← numEvaluations +1
  if numEvaluations > n
    simulate ← false
  else if numEvaluations > m
    simulate ← true
    numEvaluations ← 0
  endif
  P ← Select(P ∪ O)
  endwhile
```

Fig. 8. NSGA-II integrated with an ANN (NSGA-II-ANN).

For a fair performance comparison, tuning of parameter settings have been applied also to SMS-EMOA, MAES, and NSGA-II-ANN. In Table II-IV, the optimal parameter settings found for these algorithms for the five benchmark functions are shown. Note 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 [12]. The surrogate configuration of SMS-EMOA, MAES, and NSGA-II-ANN is the same as for MOPSA-EA.

TABLE II
OPTIMAL PARAMETER SETTINGS FOR SMS-EMOA

	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
Population size	20	20	20	20	60
No. of offspring	60	100	60	100	60
Mutation step size	1.5	1.5	1.5	1.5	1.5
Crossover	SP	Blend	SP	SP	SP
Crossover prob.	0.8	0.6	0.8	0.8	0.6

TABLE III
OPTIMAL PARAMETER SETTINGS FOR MAES

	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
Population size	100	100	100	100	100
No. of offspring	100	100	100	100	60
Mutation step size	0.5	0.5	0.5	0.5	0.5
Crossover	SP	Blend	SP	SP	SP
Crossover prob.	0.4	0.4	0.8	0.8	0.6

TABLE IV
OPTIMAL PARAMETER SETTINGS FOR NSGA-II-ANN

	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
Population size	60	60	60	100	100
Mutation step size	0.5	0.5	0.5	0.5	0.5
Crossover	Blend	Blend	SP	Blend	Blend
Crossover prob.	0.6	0.4	0.8	0.6	0.4

E. Results

Results from the benchmark functions are shown in Table V. The optimization is performed for 3000 function evaluations and the results presented are an average of 300 replications. All results have a confidence probability of 0.99 or more, calculated using Welch's t-test (defined in [24]). Note that a low value of Y, Ω and Δ, and a high value of $\mathcal{S}$ is desirable. In calculating the first three metrics, a set of 500 uniformly-distributed solutions of the true Pareto front is derived.

In the table, alongside each metric value, a rank (R) is provided stating the relative results of the four algorithms. Taking the sum of ranks of the metrics on all functions shows that MOPSA-EA obtains the best results concerning the Y, Ω and $\mathcal{S}$ metrics. On the Δ metric, MOPSA-EA and SMS-EMOA achieves the same total rank. Interestingly, MOPSA-EA obtains better overall results than both SMS-EMOA and MAES on the $\mathcal{S}$ metric, even though these algorithms are explicitly designed to maximize this metric.

TABLE V
BENCHMARK RESULTS

	Υ	R	Δ	R	Ω	R	$\mathcal{S}$	R
ZDT1								
MOPSA-EA	0.066	1	0.883	1	0.056	1	0.962	1
SMS-EMOA	2.347	4	0.898	2	1.844	4	0.687	4
MAES	0.103	2	0.975	4	0.085	2	0.957	2
NSGA-II-ANN	1.364	3	0.939	3	0.975	3	0.82	3
ZDT2								
MOPSA-EA	0.112	1	1.009	3	0.576	1	0.872	1
SMS-EMOA	2.073	4	0.847	1	1.269	4	0.707	4
MAES	0.21	2	1.006	2	0.736	2	0.85	2
NSGA-II-ANN	1.006	3	1.039	4	1.065	3	0.792	3
ZDT3								
MOPSA-EA	0.016	1	0.842	1	0.188	1	0.981	1
SMS-EMOA	2.073	4	0.847	2	1.269	4	0.707	4
MAES	0.054	2	1.04	4	0.201	2	0.975	2
NSGA-II-ANN	0.449	3	0.927	3	0.259	3	0.926	3
ZDT4								
MOPSA-EA	21.86	1	1.102	3	5.569	2	0.975	2
SMS-EMOA	66.76	4	0.94	1	60.98	4	0.748	4
MAES	28.15	2	1.14	4	4.57	1	0.98	1
NSGA-II-ANN	29.57	3	1.085	2	10.64	3	0.956	3
ZDT6								
MOPSA-EA	2.712	2	0. 93	1	2.52	1	0.728	1
SMS-EMOA	5.659	4	0.958	3	5.333	4	0.712	4
MAES	2.824	3	0.974	4	2.691	3	0.7274	3
NSGA-II-ANN	2.498	1	0.95	2	2.591	2	0.7279	2
Total								
MOPSA-EA		6		9		6		6
SMS-EMOA		20		9		20		20
MAES		11		18		10		10
NSGA-II-ANN		13		14		14		14

In the next section, the evaluation of MOPSA-EA on a real-world optimization problem from the manufacturing domain is presented.

IV. REAL-WORLD OPTIMIZATION

To evaluate the industrial strength of MOPSA-EA, it is applied to a real-world optimization problem. The 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. Recently, a new manufacturing cell has been introduced 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.

The optimization reported in this paper considers a production scenario comprising eleven different component types. A simulation model of the manufacturing cell is built using the SIMUL8 software package. As a fast surrogate of the simulation, an ANN is constructed. The ANN has eleven input nodes (each of them corresponds to the inter-arrival time of a specific component type), and two output nodes (corresponding to utilization and tardiness, respectively). The basic configuration of the ANN with respect to topology and training procedure is the same as for the ANN used in the benchmark optimizations (described in Section III-B). Regarding the configuration of the algorithm parameters, the population size is set to 40, the number of offspring is set to 20, the mutation step size is set to 1.0, and single-point crossover is being used with a probability of 0.8.

In Table VI, results from the real-world problem are presented (average of 10 replications). The optimization is performed for 400 simulations, which is the maximum number of simulations that can be performed within the optimization time-budget defined by the company. The results presented have a confidence probability of at least 0.8 (calculated using Welch's t-test). Note that since the true Pareto-optimal front of this problem is unknown, only the $\mathcal{S}$ metric is calculated. As shown in the table, MOPSA-EA achieves the best $\mathcal{S}$ value and SMS-EMOA the worst, similar to the results of the benchmark functions.

TABLE VI
REAL-WORLD OPTIMIZATION

	$\mathcal{S}$
MOPSA-EA	0.465
SMS-EMOA	0.374
MAES	0.426
NSGA-II-ANN	0.435

In the next section, general conclusions from the study are presented.

V. CONCLUSIONS AND FUTURE WORK

In this paper, a new multi-objective EA for solving computationally-intensive optimization problems has been described. This algorithm is relatively simple in its design and implementation. For example, only one population is maintained, unlike several other multi-objective EAs which maintain sub-populations or keep track of an external archive of solutions. Furthermore, in contrast to many other surrogate-assisted algorithms, there is a minimum number of user-defined parameters related to the surrogate usage. Only one parameter, the size of the offspring pool, needs to be specified.

The proposed algorithm also defines an efficient approach of integrating surrogates. Contrary to the many other surrogate-assisted multi-objective EAs (e.g. [20], [25]-[26]), simulation evaluations are not compared with surrogate evaluations. The main problem of such comparison is that surrogate fitness and simulation fitness are treated interchangeably in the evolutionary operators, which are likely to lead to poor performance in complex problems having surrogates associated with a high degree of

imprecision [6]. In addressing the problem of uncertainties in surrogate predictions and to ensure a good convergence, the proposed algorithm adopts a new method for multi-objective optimization. Contrary to previous methods of handling surrogate imprecision, this method is parameterless, generally applicable to all types of surrogate techniques, easy to understand and implement, without expensive computations, and scalable with respect to the number of objectives.

The next step of improving the algorithm will be to extend it to handle simulation noise. To capture the stochastic behavior of complex real-world systems, simulations contain randomness. Instead of modeling 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. To tackle the problem of noise in output samples is crucial because the normal path of the algorithm would be severely disturbed if estimates of the objective function come from only one simulation replication. The common technique to handle noise is to send the algorithm with the average values of output samples obtained from a large number of 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. Finding efficient methods to address the problem of noise is an important topic for further research [27], especially in multi-objective optimization [21].

ACKNOWLEDGMENT

The authors would like to thank Professor Kalyanmoy Deb for his valuable comments on this study.

REFERENCES

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

[2] Y.S. Ong, P.B. Nair, A.J. Keane, and K.W. Wong, "Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems," in *Knowledge Incorporation in Evolutionary Computation*, Springer, 2004, pp. 307-332.

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

[4] J. Boesel, R.O. Bowden, F. Glover, J.P. Kelly, and E. Westwig, "Future of simulation optimization," in *Proc. Winter Simulation Conference*, Arlington, VA, USA, December 9-12, 2001, pp. 1466-1470.

[5] D. Chafeka, J. Xuan, and K. Rasheed, "Constrained multi-objective optimization using steady state genetic algorithms," in *Proc. Genetic and Evolutionary Computation Conference*, Springer-Verlag Germany, 2003, pp. 813-824.

[6] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing*, vol. 9, pp. 3-12, Springer-Verlag Germany, 2005.

[7] T. Lacksonen, "Empirical comparison of search algorithms for discrete event simulation," in *Computers & Industrial Engineering*, vol. 40, no. 1-2, 2001, pp. 133-148.

[8] M. Emmerich, A. Giotis, M. Özdemir, T.H. Bäck, and K. Giannakoglou, "Metamodelassisted evolution strategies," in *Proc. International Conference on Parallel Problem Solving from Nature*, Granada, Spain, September 7-11, 2002, pp. 361-370.

[9] M. Reyes-Sierra and C.A. Coello Coello, "Fitness inheritance in multi-objective particle swarm optimization," in *Proc. IEEE Swarm Intelligence Symposium 2005*, Pasadena, California, June 8-10, pp. 116-123, 2005.

[10] J.H. Chen, D.E. Goldberg, S.Y. Ho and K. Sastry, "Fitness Inheritance In Multi-objective Optimization," in *Proc. Genetic and Evolutionary Computation Conference*, pp. 319-326, 2002.

[11] L.T. Bui, H.A. Abbass and D. Essam, "Fitness inheritance for noisy evolutionary multi-objective optimization", in *Proc. Conference on Genetic and Evolutionary Computation*, Washington DC, USA, pp. 779-785, 2005.

[12] M. Emmerich, K. Giannakoglou, and B. Naujoks, "Single- and multi-objective evolutionary optimization assisted by gaussian random field metamodels," in *IEEE-TEC Special Issue on Evolutionary Computation in the presence of uncertainty*, vol. 10, no. 4, pp. 421-439, 2006.

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

[14] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm," Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), May 2001.

[15] K. Deb, "Multi-objective genetic algorithms: Problem difficulties and construction of test functions," in *Evolutionary Computation*, vol. 7, pp. 205-230, 1999.

[16] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," in *Evolutionary Computation*, vol. 8, no. 2, 2000.

[17] M. Emmerich, N. Beume, and B. Naujoks, "An EMO algorithm using the hypervolume measure as selection criterion," in *Proc. Evolutionary Multi-Criterion Optimization*, Springer Berlin-Heidelberg, 2005, pp. 62-76.

[18] A. Ratle, "Optimal sampling strategies for learning a fitness model," in *Proc. Congress Evolutionary Computation*, Washington DC, USA, July 6-9, 1999, pp. 2078–2085.

[19] T. Chen, H. Chen, and R. Liu, "Approximation capability in $\{C(R^n)\}$ by multilayer feedforward networks and related problems," in *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 25-30, 1995.

[20] A.P. Giotis, K.C. Giannakoglou J., and Periaux, "A reduced-cost multi-objective optimization method based on the pareto front technique, neural networks and PVM," in *Proc. European Congress on Computational Methods in Applied Sciences and Engineering*, Barcelona, Spain, September 15-17, 2000.

[21] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments – a survey," in *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 303-317, 2005.

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

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

[24] A.M. Law and D. Kelton, *Simulation Modeling and Analysis*, Mc Graw Hill, 3rd edition, 2000.

[25] I.I. Voutchkov and A. J. Keane, "Multiobjective optimization using surrogates," in *Proc. 7th International Conference on Adaptive Computing in Design and Manufacture*, Bristol, 2006, pp. 167-175.

[26] M.K. Karakasis and K.C. Giannakoglou, "Metamodel-assisted multi-objective evolutionary optimization," *Proc. Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems*, Munich, Germany, September 12-14, 2005.

[27] A. Gosavi, *Simulation-based optimization: parametric optimization techniques and reinforcement learning*. Boston, Mass., Kluwer Academic, 2003.