Convergence Acceleration Operator for Multiobjective Optimization

Salem F. Adra, *Member, IEEE*, Tony J. Dodd, Ian A. Griffin, and Peter J. Fleming

***Abstract*— A convergence acceleration operator (CAO) is described which enhances the search capability and the speed of convergence of the host multiobjective optimization algorithm. The operator acts directly in the objective space to suggest improvements to solutions obtained by a multiobjective evolutionary algorithm (MOEA). The suggested improved objective vectors are then mapped into the decision variable space and tested. This method improves upon prior work in a number of important respects, such as mapping technique and solution improvement. Further, the paper discusses implications for many-objective problems and studies the impact of the use of the CAO as the number of objectives increases. The CAO is incorporated with two leading MOEAs, the non-dominated sorting genetic algorithm and the strength Pareto evolutionary algorithm and tested. Results show that the hybridized algorithms consistently improve the speed of convergence of the original algorithm while maintaining the desired distribution of solutions. It is shown that the operator is a transferable component that can be hybridized with any MOEA.**

***Index Terms*— Evolutionary multiobjective optimization, neural networks.**

I. Introduction

REAL-WORLD problems commonly require the simultaneous consideration of multiple competing performance measures. Without loss of generality, a multiobjective optimization problem (MOP) can be formulated as a minimization of a function $Z(X)$, where $Z(X) = \{Z_1(X) \cdots Z_n(X)\}$ is a vector of objective functions, n is the number of objectives, and X is a vector of decision variables. The optimization problem consists of finding the decision vector, or set of vectors, that results in the best solution or set of solutions in the objective space. For multiobjective problems in which objectives are competing, no single optimal solution exists, but rather a set of candidate solutions known as the approximation set. The ideal approximation set of decision vectors will be characterized by the fact that no other solution within another approximation set offers better objective function values across all objectives. This set of candidate solutions is said to be non-dominated and is known as the Pareto-optimal set, from which the decision maker ultimately selects an acceptable solution. The associated objective vectors form the tradeoff surface (or Pareto front) in the objective space. Fig. 1 shows an optimization problem where three decision variables (X_1, X_2 and X_3) are optimized with respect to two competing objectives (Z_1 and Z_2), illustrating the mapping of a decision vector into objective space and showing the Pareto front for this idealized case.

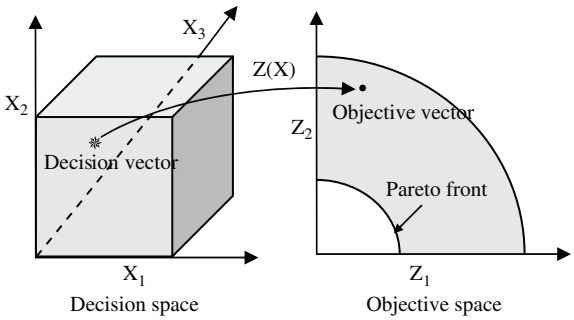

Fig. 1. Multiobjective problem domain.

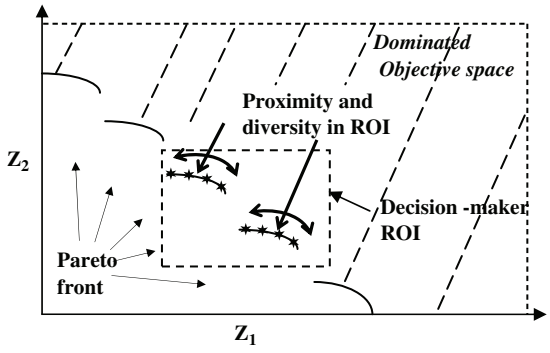

Fig. 2. Good set of solutions to a multiobjective optimization problem in terms of proximity, diversity, and relevance (i.e., location in ROI).

The approximation set offered to the decision maker is required to be as close as possible to the true Pareto front. The approximation set is also required to be well spread across objective space, presenting the decision maker with a well-distributed set of solutions within the region(s) of interest (ROI) [1]. These two characteristics of an approximation set are termed proximity and diversity, respectively, and are illustrated in Fig. 2.

To be of practical use, a multiobjective (MO) optimizer must produce an approximation set with acceptable proximity and diversity within acceptable computational resources (most importantly, a fixed and limited budget of objective function evaluations). The time taken by an algorithm to perform a given number of search iterations for a particular

Manuscript received January 30, 2008; revised October 7, 2008; accepted November 29, 2008. Current version published August 14, 2009.

S. F. Adra is with the Department of Computer Science, University of Sheffield, Sheffield S1 4DP, U.K. (e-mail: S.Adra@sheffield.ac.uk).

T. J. Dodd and P. J. Fleming are with the Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield S1 4DP, U.K. (e-mail: t.j.dodd@sheffield.ac.uk; P.Fleming@sheffield.ac.uk).

I. A. Griffin is with Rolls-Royce plc, Derby, DE24 8BJ, U.K. (e-mail: Ian.Griffin@Rolls-Royce.com).

Digital Object Identifier 10.1109/TEVC.2008.2011743

problem is dependent upon the available computing power. However, the efficiency of a multiobjective evolutionary algorithm (MOEA) can be determined by the proximity and diversity of the approximation sets produced from a given number of iterations over multiple runs of the algorithm [2] and within a fixed budget of objective function evaluations. The use of population-based optimization techniques, such as evolutionary algorithms (EAs), is a suitable approach for addressing MO problems. The population-based nature of EAs makes them well suited to addressing non-commensurate MO problems, as they simultaneously explore a family of points in the search space. However, EAs are known to present two main shortcomings that are addressed in this paper.

The first shortcoming is that an EA offers no guarantee of finding optimal solutions within a specified number of iterations, i.e., a single run or more. Traditional evolutionary computation (EC) techniques usually consist of an explorative set of procedures operating in the decision variable space. The operators within these algorithms mimic Darwinian biological principles of stochastic selection followed by recombination and mutation [3], [4]. Starting either from a random population of candidate solutions or from a previously known set of solutions in decision variable space, EAs calculate the corresponding objective function values, assign them fitness scores reflecting their utility in the application domain, and bias the search toward high-potential areas of the space by forcing the "survival-of-the-fittest" solutions. Through the variation operators operating in the decision variable space, new solutions are produced with the assumption that "good" parents are more likely to produce "good" offspring and hence should contribute more to the next generations.

The second shortcoming of EAs is that, in many application domains, calculating the true objective function may be computationally expensive; for example, in some applications, a single objective function evaluation can require hours to compute [5]. Given their generational population-based approach, EAs require a significant number of objective function calculations to be performed. The use of approximated models using neural networks (NNs), or other metamodelling techniques, such as Kriging-based approximations, or response surface models [6], [7], provides low computational burden alternatives to full objective function evaluation [8], [9]. *Metamodeling* is a well-established research discipline that focuses on building approximated models which reduce the computational effort needed to compute exact and expensive objective functions. While the most common use of the metamodeling technique is to substitute the use of expensive objective functions [10], metamodeling techniques are also used to model noisy [11] and ill-defined [12] objective functions. The approximated models, also called metamodels or surrogate models, are generally models of higher level of abstraction compared with the exact models they represent. Hence, an essential matter that should be noted when deploying metamodeling techniques is that the solutions achieved for a metamodel should be cautiously analyzed before being considered as solutions to the exact model, which is usually of higher fidelity [13]. For a comprehensive survey about fitness approximation in EAs, the interested reader is directed to [14].

In an attempt to address the two shortcomings discussed above, a convergence accelerator is proposed that maps from the objective space to the decision variable space (in the reverse direction to a metamodeling technique). This operator is a transferable component that can be hybridized with any MOEA. The purpose of this convergence acceleration operator (CAO) is to enhance the performance of the host MOEA in terms of the proximity of the approximation set for a given number of objective function calculations without impeding the active diversification mechanisms of these search strategies. In this paper, the CAO is hybridized with two widely used MOEAs, the non-dominated sorting genetic algorithm (NSGA-II) [15] and the strength Pareto evolutionary algorithm (SPEA2) [16]. EAs operate in decision space and perform decision space to objective space mapping but tend to fail to exploit direct use of the objective space—a lost opportunity. In contrast to this, the CAO features a direct search in objective space and then uses predictions to map from objective space to decision space.

The idea of performing local search in the objective space and seeking to map a certain objective vector back to its corresponding decision vector was first introduced in [17], [18], and applied to a bi-objective real-world problem in [19]. They proposed a method to accelerate the search of an MOEA by approximating the function that maps from the objective space to the decision space using NN techniques. More specifically, their method, which is hybridized with the reduced Pareto set genetic algorithm (RPSGA) [20], used a multilayer perceptron (MLP) approach [21] to map in the reverse direction (i.e., objective vectors as inputs and decision vectors as outputs). The trained MLP is then deployed to predict the approximate vectors of decision variables which should correspond to the objective vectors introduced by a local search around the nondominated solutions arising from the previous generation. The local search suggested by Gaspar-Cunha *et al*. attempts to improve the locally nondominated solutions by minimizing their objective values (normalized in the range [0, 1]) directly in the objective space. Each objective value is minimized by an absolute and fixed step length. Gaspar-Cunha *et al*. tested their technique on a set of bi-objective test functions [22] and a bi-objective optimization problem of screw geometry and reported an accelerated convergence on these bi-objective problems compared to the stand-alone RPSGA.

Independently of Gaspar-Cunha *et al*., Adra *et al*. [8] investigated the utility of deploying direct local search in the objective space and inverse NN predictions on a *many*-objective optimization problem. They applied it to an eight-objective problem of aircraft control system design and used the multiobjective genetic algorithm (MOGA) [23] which integrated Fonseca and Fleming's preferability operator [24] for incorporating DMs preferences for search space reduction.

The contribution of this paper is to develop further the approach of Adra *et al*., while seeking to improve the efficiency and effectiveness of Gaspar-Cunha *et al*.'s original approach. The improvements consist of the following:

1) a variable step length approach for the local improvement step;
2) the use of radial basis function neural networks for the mapping stage;

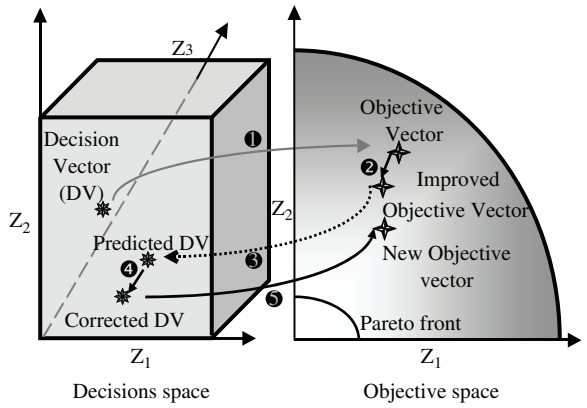

Fig. 3. Convergence acceleration operator steps used in generating a single candidate solution.

3) the use of a correction step to ensure that the decision variables remain feasible;
4) the computation of exact objective values for the improved solution.

The paper includes a rigorous assessment of this new approach using two widely used MOEAs and accepted performance criteria. Recognized test functions from the literature are used to assess the effect of the CAO, when deployed on challenging bi-objective problems, problems with 3, 8, and 12 objectives and, finally, on a real-world computationally expensive problem comprising 14 objectives. The study concludes with an analysis of the computational effort required for different stages of the method. The paper is organized as follows. In Section II, the proposed CAO operator is introduced and described. Section III describes the test procedures used in the comparative testing of the standard and CAO-enhanced algorithms. Section IV presents results of the tests described in Section III, and concluding remarks are provided in Section V.

II. Proposed Convergence Acceleration Operator

A. Overview

Fig. 3 illustrates the actions of the hybridized MOEA which includes the CAO. Trajectories 2–5 describe the specific actions of the CAO.

Trajectory 1: the mapping between a decision variables vector realized by a MOEA and its corresponding computed objective values vector.
Trajectory 2: the resulting objective vector—a member of the approximation set at generation t—is improved in the objective space using a local search (described in the next section).
Trajectory 3: a prediction of the decision variables vector corresponding to the improved objective vector using an NN trained with the exact data resulting from earlier evaluations of objective functions—at the same generation t—during the MOEA search.
Trajectory 4: rectification of any invalid decision variable vector introduced by the NN mapping by reflecting out-of-bounds values of the produced decision variables to their nearest values in their domain of definition.
Trajectory 5: finally, calculation of the exact objective values vector for the proposed decision variables vector in the normal way. These candidate solutions will then compete for archive update and insertion with the best solutions currently stored in the online archive.

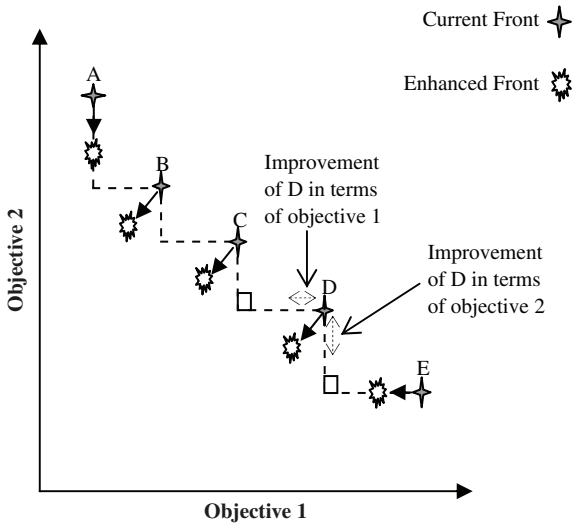

Fig. 4. Deterministic improvement of the tradeoff surface in objective space.

When the CAO is launched, it starts by deterministically improving the best solutions achieved; these solutions are the elite solutions stored in the online archive of the host algorithm. This improvement takes place in the *objective* space and produces an enhanced version of the archive. The CAO then uses a trained NN mapping procedure to predict the corresponding *decision* vectors for the enhancements to the archive. A check of these new decision vectors is made, aimed at reflecting any out-of-bounds decision variables arising from the mapping back into their allowed domain. A correction step is then applied, whereby the true objective values corresponding to all of these new decision vectors are calculated, thereby maintaining the fidelity of the optimization problem.[1] Thus the correction step establishes a correct mapping from the decision space to the objective space; the CAO, therefore, does not depend on the fidelity of the inverse mapping step. The correction step of the CAO is an enhancement to the technique suggested in Gaspar-Cunha and Vieira [17] and Gaspar-Cunha *et al.* [18]; their approaches did not rectify any predictions inaccuracy introduced by the NN. After the correction step, the enhanced and the original archive of solutions compete to populate the new archive for the next generation, which will represent the pool from which solutions are selected and recombined. The two components of the CAO are described in detail in the following sections.

B. Local Improvement in Objective Space

The CAO takes place after the recombination and mutation processes and operates on the elitist solutions that would normally propagate to the following generation (or get presented to the DM). The CAO is an auxiliary local improvement operator that does not replace the variation operators in EAs. The first CAO step is a deterministic local improvement procedure in the objective space. This is the component responsible for

[1]This is quite different from metamodeling techniques, where the optimized model is of lower fidelity compared to the exact model.

speeding up convergence and hence reducing computational effort by producing better results within a fixed budget of objective function evaluations. It achieves this by steering the objective values obtained by the MOEA toward an improved Pareto front. The objective space local improvement process is implemented in this paper for any number n of objectives, and is illustrated in Fig. 4 on a bi-objective problem ($n = 2$) for simplicity.

In general, nonboundary solutions in terms of any specific objective (solutions B, C, and D in Fig. 4) will be improved in terms of all the performance measures by steering their objective values into a region of improved objective function values. The new "improved" values for the objectives (of each nonboundary solution) are determined by linearly interpolating a new value for each objective, between its current value and the next best value(s) achieved for that objective within the population.[2] This is described by

$$Z_{D'} = Z(x_D - (x_D - x_C)/h, y_D - (y_D - y_E)/h)$$

where $Z(x, y)$ represents a point in the bi-objective space, $Z_{D'}$ is the "improved" objective value, and h is the interpolation step factor. This process is annotated for solution D in Fig. 4. Compared to solution D, solution C has the next best value in terms of objective 1 while solution E possesses the next best value in terms of objective 2. Boundary solutions in terms of a certain objective (solutions A and E in Fig. 4) are improved in terms of the remaining objectives. In other words, solution A will be improved in the *"Objective 2"* axis direction, thereby enhancing its overall quality by improving it in terms of objective 2, and solution E will be improved in the *"Objective 1"* axis direction, consequently improving its overall worth by enhancing it in terms of objective 1.

It should be noted that the suggested local search within the CAO was aimed at addressing the primary MO requirement of convergence [25]. The diversity requirement is not addressed but, instead, is dealt with by the active diversification mechanism of the MOEA hosting the CAO. It should also be noted that as the dimensionality of the objective space increases, the diversity requirement becomes easily achievable and needs to be controlled and limited to avoid hampering the convergence process (see, for example, [26]).

In their proposed technique, Gaspar-Cunha *et al*. used a fixed step length for the local search in the objective space. In this paper, an adaptive step length, which is controlled by step factor h, is proposed for the local improvement step in the objective space. The step factor h is an application-dependent parameter and should be carefully chosen. A smaller h value (larger step length) is recommended for early generations of the optimization, with its value gradually increasing.

Moreover, since the decision vectors of the improved front of solutions are to be predicted by the NN (a process described in the following section), it is essential that the new introduced solutions in the objective space reside within the neural network's reliable zone of prediction. NNs are very practical tools for regression problems and data fitting, but in common with other curve fitting and data modeling techniques, they are known to be unsuitable for extrapolation tasks.

As a result, the step factor h should be chosen in a way that maximizes the *local* improvement step in the objective space while preventing the introduction of solutions which reside outside the NN's local region of training. In this paper, a relatively small value is chosen for the initial value of h at the start of the optimization process for each of the MOPs investigated, i.e., a relatively large step. Based on the CAO's rate of success R_S, the initial value of h is then gradually increased (i.e., step length decreased) as the optimization process progresses.

The CAOs rate of success R_S is defined as

$$R_s = \frac{R_e}{R_t} \times 100\%$$

where R_e is the number of "effective" solutions which are introduced by the CAO, corrected at the correction step, and successfully chosen to propagate to the following generation, and R_t is the total number of solutions which are introduced by the CAO.

When the CAO's rate of success falls below a certain threshold τ the step factor h is increased by a certain cooling factor ε for use at the next generation. The step factor h at generation $t = i + 1$ is increased using the following rule:

$$if\, R_S(t = i + 1) < \tau \rightarrow h_{t=i+1} = h_{t=i} \times \varepsilon.$$

The CAOs rate of success decreases in one of two cases if the step length is too large:

1) the CAO introduces solutions beyond the Pareto front, or;
2) the CAO introduces solutions outside the NNs reliable region of prediction, and thus the NN is making extrapolation predictions.

As the optimization process progresses, the first scenario is very likely to occur when a fixed value for h is used. From the experiments carried out during the course of this paper, it was observed that fewer than 5% of the solutions suggested by the CAO were effective in some scenarios where the value of h was fixed. In this paper, different values for the initial step factor h the threshold τ and the cooling factor ε were investigated for each of the MOPs used. The different values tested and a general guideline for choosing h, τ, and ε are described in the results section for each of the MOPs.

C. Objective Space to Decision Space Mapping

The description of the mapping method in [19] is very brief and differs from the local mapping approach described in this section. The second component of the CAO consists of a NN trained to map the new solutions thus generated in objective space by the first phase of the convergence accelerator back to the corresponding decision variable vectors. NNs [21] are a powerful approach for modeling patterns of data in order to produce predicted values of unknown systems. The NN needs to be trained to achieve desirable predictions and to model complex functions as closely as possible. The process of training the NN consists of providing it with samples of input–output data and manipulating weighting variables by adjusting their values and minimizing prediction errors.

[2]The nondominated solutions are improved in an objectivewise order, each time sorting the archive of locally nondominated solutions in terms of a certain objective.

The second component of the CAO only aims to build a *local model*[3] of the function which maps from the objective space to the decision space at a certain iteration of the optimization process. This is achieved by training an NN, using exact objective vectors as inputs and their corresponding decision variable vectors as outputs. The training data is the exact data resulting from the objective function values derived within a single iteration of a MOEA. More specifically, at every iteration of the optimization process (or alternatively, when the CAO is called, if its use is optional), a new local model is built based on the locally nondominated solutions (objective vectors and corresponding decision vectors). The local model is then solely used within the same iteration to predict the decision variables of the new objective values introduced by the first component of the CAO that locally steers the local Pareto front toward an enhanced Pareto front.

When training an NN, it is vital to ensure well-spread and problem-defining data. Abundance of data is an essential point for achieving well-trained NN and high-fidelity models, but can be a problem in some computationally expensive applications.

Nevertheless, the CAO is designed for accelerating population-based optimization strategies such as evolutionary algorithms, where data abundance is usually an essential requirement for the success of such techniques. If an application is computationally very expensive, the requirement for data abundance can be addressed by using a cheaper and acceptable metamodel that approximates the objective function.

In the context of this paper, the investigation is confined to the use of NNs within the framework of the CAO, due to their flexibility and good reputation for universal approximation capability, given a sufficient number of hidden units and a suitable choice of parameters [27], [28]. In this paper a specific type of NN, the *radial basis function* (RBF) [21], is used to build the local models of the local Pareto fronts and for predicting the decision variables of the solutions introduced by the local search.

In a comparative study of metamodeling techniques, Jin *et al*. [29] compared the performance of RBF, polynomial regression, the Kriging method, and multivariate adaptive regression splines (MARS) under multiple modeling criteria. They used 14 test problems with representative features of engineering design problems. These features consisted of *problem scale* (with large and small number of variables), *nonlinearity of the performance behavior*, and *noisy versus smooth behavior*. In order to measure the performance of the studied metamodeling techniques with respect to the three previously mentioned metamodeling criteria, Jin *et al*. measured the following aspects for the RBF, Kriging, polynomial regression, and MARS:

1) accuracy of prediction;
2) robustness (i.e., accuracy of prediction when different problem types and sample sizes are used);
3) efficiency (i.e., computational effort required for building models and predicting new values);
4) transparency ("capability of illustrating explicit relationships between input variables and responses"), and;
5) conceptual simplicity (i.e., ease of implementation).

Compared to Kriging, MARS, and polynomial regression, Jin *et al*. concluded that, overall, the RBF NN excelled in terms of robustness and accuracy in most of the studied categories of test problems.

In [17], [18], the authors trained a multilayer perceptron (MLP) using the backpropagation algorithm to learn the function that maps from the objective space to the decision space. MLPs are feed-forward NNs generally trained with the standard backpropagation algorithm [21] (using the gradient descent optimizer) and widely used in the field of pattern classification and recognition. The use of MLPs within an acceleration operator such as the CAO is a component that works against the purpose of an accelerator. This is due to the fact that an MLP is trained iteratively, which can be time consuming.

In the NN literature, the backpropagation algorithm is one of the most studied and used algorithms for training MLPs [21], [30]. When MLPs are trained with the backpropagation learning algorithm, the output results of the MLP and the exact results are compared and an error value (usually the mean squared error) is calculated and fed back through the network. The parameters (weights) of the MLP hidden units are then adjusted and optimized using a nonlinear optimizer, usually the gradient descent algorithm. MLPs have two major drawbacks when trained with the standard backpropagation algorithm and used within a convergence accelerator. These drawbacks are the slow convergence and the susceptibility of getting stuck at local minima in terms of the error functions (and hence suboptimal weights for the MLP units). It should be noted, however, that nowadays many alternatives and modifications to the backpropagation algorithm and the gradient descent optimizer are commonly used [21]. The conjugate gradient descent, the Levenberg–Marquardt algorithm, quasi-Newton methods, and Delta-bar-Delta [31] are examples of such alternatives and usually perform significantly better than the backpropagation algorithm.

RBF NNs, on the other hand, are two-layered NNs and well known to be practical alternatives to MLPs due to their much faster two-stage training process [21], [28]. In RBF NNs the activation functions of the hidden layer consist of RBFs, most commonly a Gaussian function, which replaces the nonlinear activation functions (sigmoidal) used in MLPs. A RBF [presented in (1)] is a real-valued function whose values depend on the distance of a certain input 'x' from a certain center 'c'

$$\phi(x, c) = \phi(\|x - c\|). \tag{1}$$

A Gaussian function is a common type of RBF and is described in (2), where c is the center of the Gaussian, and w is its width

$$\phi(x) = \exp\left(-\frac{(x-c)^2}{w^2}\right). \tag{2}$$

Unlike the MLP training process where the activation of the hidden units consists of nonlinear computation of the scalar product of the input vectors and the weight vectors of the

[3]This is in contrast to standard metamodeling techniques which build global models of a certain objective function.

hidden neurons, the hidden neurons of an RBF network are activated by calculating a nonlinear function of the distance between the input and the RBF centers. The RBF network mapping to the output layer is described in (3), where x is the input data, k is the number of output units, M is the number of RBFs Φ, and w_{kj} are the output layer weights

$$y_k(x) = \sum_{j=0}^{M} w_{kj}\Phi(x). \tag{3}$$

The input data is passed through the input layer and then processed by the RBFs of the hidden layer. The outputs of the hidden layer units are then linearly combined and processed at the output layer of the NN. The linear mapping of the hidden layer's values into the output layer of an RBF network is an advantageous feature compared to MLPs. This advantage is due to the fact that training an RBF consists of adjustments to a linear mapping from the hidden layer to the output layer. As a result, the manipulation of a linear error function in terms of the RBF weights makes it straightforward to efficiently use linear algebra techniques to find the global optima in the parameter space of the error function and hence the optimal values for the RBF weights.

RBF NNs, therefore, do not suffer from the problem of getting stuck at local minima in the parameter space because of their quadratic error functions whose global minima can be easily found. The parameter estimates are guaranteed to correspond to the global minimum for a given RBF structure. However, it should be noted that similar to the MLP, the choice of an RBF structure (number of units, the position and widths of the basis functions) is an optimization problem. In this paper, an unsupervised training process is used to set the number of RBF units, widths, and centers. This is described in the next section. Nonetheless, using a RBF NN simplifies the training process because the parameter estimates will at least be optimal. RBFs are a more suitable choice for deployment within the CAO than MLPs due to the considerably faster learning process of RBFs. This makes it feasible to initialize and train a different RBF to model a local model at every iteration at which the CAO is executed.

Two possible approaches to training the NN component of the CAO hybridized with a MOEA are proposed: online and offline training modes. In this paper, the online training mode is further elaborated and investigated. However, the interested reader is directed to [32] where the offline training mode is explored and investigated. The online mode consists of concurrently training and validating the NN during the execution of the MOEA. In the online mode, and at every generation of a MOEA, the training data is collected and instantly used for training a RBF NN, thereby building a local model of the mapping function from the objective space to the decision space.

III. Experimental Framework

In this section the different experiments investigated for testing the CAO are introduced. The experiments are divided into two different categories, reflecting the dimensionality of the optimization problem in the objective space: bi-objective optimization and many-objective optimization problems. Without any loss of generality, all the optimization problems used in this paper consist of minimization problems. The different MOEAs investigated in this paper are benchmarked in a way that is similar to the approach used in [25]. The number of objective function evaluations is fixed beforehand and the performances over multiple runs of the MOEAs are determined and compared. As a result, MOEA "A" is deemed more competent than MOEA "B" if its average performance over multiple runs is superior to the performance of B. This approach of benchmarking MOEAs is more efficient than the approach where resources are determined for achieving the optimal results, which are known *a priori*, for a certain optimization problem. Bosman and Thierens [25] state that this way of benchmarking "represents a more practical situation, since we usually do not assume that an unlimited number of function evaluations is available."

In this paper a well-established set of optimization problems is first investigated and used to test the performance of the introduced convergence accelerator. These optimization problems represent a subset of test functions that belong to a test suite of bi-objective problems presented in [22], and which will be referred to as the ZDT test functions. The ZDT suite is comprised of six problems, each one presenting a specific characteristic that generally cause difficulties to major evolutionary optimization strategies. The bi-objective test functions used to examine the effect of the introduced CAO are the ZDT1 (convex test function), the ZDT3 (discontinuous test function), and the ZDT6 (nonuniform test function). The ZDT test functions (1, 3, and 6) are presented below.

$$\begin{aligned} \text{Minimize} \quad & F(x) = (f_1(x_1), f_2(x)) \\ \text{subject to} \quad & f_2(x) = g(x_2, \ldots, x_m) \\ & \quad \cdot h(f_1(x_1), g(x_2, \ldots, x_m)) \\ & \text{where } \mathbf{x} = (x_1, \ldots, x_m) \end{aligned}$$

ZDT1 Convex Pareto front formed with $g(x) = 1, m = 30$, and $x_i \in [0, 1]$

$$f_1(x_1) : x_1$$
$$g(x_2, \ldots, x_m) : 1 + 9\sum_{i=2}^{m} x_i/(m-1)$$
$$h(f_1, g) : 1 - \sqrt{f_1/g}$$

ZDT3 Discrete Pareto front formed with $g(x) = 1, m = 30$, and $x_i \in [0, 1]$

$$f_1(x_1) : x_1$$
$$g(x_2, \ldots, x_m) : 1 + 9\sum_{i=2}^{m} x_i/(m-1)$$
$$h(f_1, g) : 1 - \sqrt{f_1/g} - (f_1/g)^2 \sin(10\pi f_1)$$

ZDT6 Nonuniform distribution across a non convex Pareto front formed with $g(x) = 1, m = 10$, and $x_i \in [0, 1]$

$$f_1(x_1) : 1 - e^{-4x_1}\sin^6(6\pi x_1)$$
$$g(x_2, \ldots, x_m) : 1 + 9\left(\left(\sum_{i=2}^{m} x_i\right)/9\right)^{\frac{1}{4}}$$
$$h(f_1, g) : 1 - (f_1/g).$$

Studies have shown that conclusions drawn from bi-objective optimization frameworks cannot be generalized to the many-objective optimization frameworks with more than two competing objectives [2], [26]. In order to rigorously investigate the effect of the CAO, optimization scenarios with more than two objectives and various objective relationships were investigated. Hence, 3-, 8-, and 12-objective versions of DTLZ2, which is a real-parameter scalable test function introduced in [33] to test the effectiveness of MOEAs in dealing with increasing numbers of objectives, were used. DTLZ2 is presented in (4), where M presents the number of objectives, $n = M+K-1$ is the number of decision variables, and K is a "difficulty parameter" ($K = 10$ in this paper). DTLZ2(M) denotes an M-objective instance of DTLZ2.

$$
\begin{aligned}
\min . \quad & z_1(x) = [1 + g\,(x_M)] \cos\left(x_1 \frac{\pi}{2}\right) \cos\left(x_2 \frac{\pi}{2}\right) \cdots \\
& \qquad \times \cos\left(x_{M-2} \frac{\pi}{2}\right) \cos\left(x_{M-1} \frac{\pi}{2}\right), \\
\min . \quad & z_2(x) = [1 + g\,(x_M)] \cos\left(x_1 \frac{\pi}{2}\right) \cos\left(x_2 \frac{\pi}{2}\right) \cdots \\
& \qquad \times \cos\left(x_{M-2} \frac{\pi}{2}\right) \sin\left(x_{M-1} \frac{\pi}{2}\right), \\
\min . \quad & z_3(x) = [1 + g\,(x_M)] \cos\left(x_1 \frac{\pi}{2}\right) \cos\left(x_2 \frac{\pi}{2}\right) \cdots \\
& \qquad \times \sin\left(x_{M-2} \frac{\pi}{2}\right), \\
& \vdots \quad \vdots \qquad \vdots \\
\min . \quad & z_{M-1}(x) = [1 + g\,(x_M)] \cos\left(x_1 \frac{\pi}{2}\right) \sin\left(x_2 \frac{\pi}{2}\right), \\
\min . \quad & z_M(x) = [1 + g\,(x_M)] \sin\left(x_1 \frac{\pi}{2}\right), \\
& w.r.t \quad x = [x_1, \ldots, x_n], \\
& \text{where } g\,(x_M) = \sum x_i \in x_M\,(x_i - 0.5)^2, \\
& \text{with } \quad x_M = [x_M, \ldots, x_n], \ \textit{and } 0 \le x_i \le 1 \\
& \text{for } i = 1, 2, \ldots, n, \ \textit{with } n = M + \kappa - 1
\end{aligned} \tag{4}
$$

DTLZ2 possesses a continuous and non-convex global Pareto front and comprises two types of decision variables responsible for controlling the solution convergence toward the global Pareto front and the solution distribution in the objective space, respectively. The scalable DTLZ2 test function belongs to the DTLZ test suite which covers many problem characteristics such as discontinuity and multimodality. DTLZ2 has been previously demonstrated in [26] as a challenge for MOEAs especially as the number of objectives increases. Since the CAO does not require any assumptions about the nature of the optimization problem, DTLZ2 was deemed sufficient to test the performance of the CAO on optimization problems with increasing number of objectives.

It should be noted that the ZDT and DTLZ2 test functions present common similarities in the decision space. In particular, the different versions of the DTLZ2 test function are characterized by the fact that the last k decision variables of any Pareto optimal solution presented the same value. This last feature is rarely present in real-life applications. As a result, the performance of the CAO is also investigated on two challenging test functions with nonseparable decision variables.

A nonseparable MOP is a problem with variable dependencies. Nonseparability is a feature that is common in real-life applications. In [40] and [41], Huband *et al*. introduced a tool kit which allows the designer to construct scalable multiobjective test functions with well-defined Pareto fronts and desired characteristics. Using this tool kit, Huband *et al*. proposed a test suite of nine scalable multiobjective problems featuring important characteristics such as nonseparability. In this paper, the CAO will be additionally assessed on three-objective instances of the test functions WFG6 and WFG9 [WFG6(3) and WFG9(3)], two of the most complex and non-separable test functions suggested by Huband *et al*.[4] WFG9, in particular, presents a deceptive decision space and represents a formidable challenge for most MOEAs.

The elitist non-dominated sorting genetic algorithm NSGA-II [15] and the strength pareto evolutionary algorithm SPEA2 [16], which are two well-established MOEAs and highly cited second-generation optimizers in the EMO community, were chosen as the comparison benchmark optimizers for the problems used in this paper. Each was also hybridized—NSGA-II/CAO, SPEA2/CAO—with the introduction of the CAO into their cycles to test its effect. In Figs. 5 and 6, the hybridization interface of the CAO into SPEA2 and NSGA-II is illustrated within the pseudocode descriptions of the hybrid versions of the two optimizers SPEA2/CAO and NSGA-II/CAO.

The artificial neural networks (RBF and MLP) used in the CAO are implemented, initialized, trained, and validated using the utility functions provided in NETLAB [28], which is a free neural network toolbox for MATLAB[5] and which can be downloaded from http:www.ncrg.aston.ac.uk/netlab/index.php. Optimizer configurations used in the experiments involving these four optimizers—NSGA-II, SPEA2, NSGA-II/CAO, and SPEA2/CAO—are given in Table 1. In this paper, the CAO was operating continuously, from the first to the last generation. At every generation, a different RBF-neural network is trained and then used within the CAO for local improvement and predictions. Training the RBF NN with local and limited data and solely using it as a local model at a specific generation helps to overcome the problem of training the NN with conflicting data resulting from possible one-to-many mappings from objective space to the decision space. This is an important aspect that was not addressed in the work of Gaspar-Cunha *et al*. The test function WFG9, which presents one-to-many mappings from the objective space to the decision space, will test this feature.

Because of the CAO correction step, the number of objective function evaluations per generation required in NSGA-II/CAO and SPEA2/CAO is twice the number of objective function evaluations per generation required in SPEA2 and NSGA-II. In order to compare the algorithms for the same number of objective function evaluations, the CAO-augmented MOEAs were executed for 50 generations per run, while NSGA-II and SPEA2 were executed for 100 generations. The larger level of

[4]C++ codes for WFG9, WFG6 and Huband *et al*.'s toolkit can be downloaded from: http://www.wfg.csse.uwa.edu.au/publications.html

[5]MATLAB is a software package for technical computing, developed by The MathWorks, Inc.

-Initialize Population
-Generate initial random population $\mathbf{P}_0$ of size **Nind** and an initial Archive $\mathbf{A}_0$
-Evaluate objective values
-Calculate fitness values of individuals in $\mathbf{P}_0$ and make $\mathbf{A}_0 = \mathbf{P}_0$
For i = 1 to Gen
-Copy all nondominated individuals in $\mathbf{P}_{i-1}$ and $\mathbf{A}_{i-1}$ to $\mathbf{A}_i$.
If size of $\mathbf{A}_i$ > **Nind** then reduce $\mathbf{A}_i$
Else fill $\mathbf{A}_i$ with dominated individuals in $\mathbf{P}_{i-1}$ and $\mathbf{A}_{i-1}$

Apply CAO
-Component: NN Training
-Initialize an RBF NN and train it with $\mathbf{A}_i$
-Input: Objective Vectors of $\mathbf{A}_i$
-Output: Decision Vectors of $\mathbf{A}_i$

*–Component: Objective Space local improvement–***on** A_i
–Component: Objective Space to Decision Space Predictions
–Component: Correction Step
–Update $\mathbf{A}_i$

-Generate new population $\mathbf{P}_i$ from $\mathbf{A}_i$–size **Nind**
-Binary tournament selection
-Recombination
-Mutation
-Evaluate objective values for the offspring population $\mathbf{P}_i$
-Calculate fitness values of individuals in $\mathbf{P}_i$ and $\mathbf{A}_i$ based on the objective vectors of $\mathbf{P}_i$ and $\mathbf{A}_i$ combined
End loop

Fig. 5. SPEA2/CAO pseudocode.

-Initialize Population
-Generate random population $\mathbf{P}_0$–size **Nind**
-Evaluate objective values
For i = 1 to Gen
-Assign rank to $\mathbf{P}_{i-1}$
-Determine crowding distance for each solution in $\mathbf{P}_{i-1}$
-Generate offspring population **Q**–size **Nind**
-Binary tournament selection
-Recombination
-Mutation
-Evaluate objective values for the offspring population **Q**
-Combine $\mathbf{P}_{i-1}$ and **Q**
-Assign rank to the combined population
-Determine crowding distance for each solution in the combined population
-Select **Nind** solutions to form $\mathbf{P}_i$

Apply CAO-
-Component: NN Training
-Initialize an RBF NN and train it with $\mathbf{P}_i$
-Input: Objective Vectors of $\mathbf{P}_i$
-Output: Decision Vectors of $\mathbf{P}_i$

*–Component: Objective Space local improvement–***on** $\mathbf{P}_i$
–Component: Objective Space to Decision Space Predictions
–Component: Correction Step
–Update $\mathbf{P}_i$

End loop

Fig. 6. NSGA-II/CAO pseudocode.

exploration and global search thus afforded to NSGA-II and SPEA2 is an advantage in their favor.

The configuration of the optimizers presented in Table I is a standard configuration commonly used in the EMO community when using NSGA-II, SPEA2, or other MOEAs for optimizing the problems previously presented. The major difference was the number of generations used in this paper for the CAO-augmented MOEAs, which was relatively small compared to the standard number of generations (around 150 generations) normally used in comparative studies, such as the study by Zitzler *et al*. [22].

This choice of configuration was intended to study the effect of the convergence acceleration and any benefits that might be introduced by the CAO. Concatenation of real number decision variables was the convenient choice for encoding the problems under investigation. Due to the stochastic nature of the evolutionary strategies, a well-based judgment concerning the performance of a specific algorithm cannot be stated unless the whole optimization process is repeated a number of times. Here, each algorithm is subjected to 10 runs, each running for 100 generations (for SPEA2 and NSGA-II) and 50 generations (for the CAO-hybridized MOEAs). Moreover, the parameters of the RBF networks for each optimization problem solved are investigated within a set of initial experiments and are set to the best parameters achieved. One of the drawbacks of NNs is the lack of standardization in choosing the number of hidden layers and hidden neurons per layer, which constitutes the architecture of an NN. It is common practice to choose the NN architecture based on previous practice and expertise or based on trial-and-error experiments.

The training of the RBF NN is a two-stage process. The first stage consists of setting the parameters (centers and widths) of the radial functions (Gaussian functions are used in this paper). In the context of the CAO, the training data consists of the elite population of candidate solutions at a certain generation of the MOEA. As stated by Nabney [28]: "One of the main advantages of RBF networks, as compared to MLP, is that it is possible to choose good (though possibly not optimal) parameters for the hidden units without having to perform a full non-linear optimization of all the network parameters."

In this paper, 80% of the population of candidate solutions (objective vectors) are chosen at random and set as the centers of the RBFs. The widths of the RBFs that compose the units of the RBF network are an application-dependent design choice, and should be chosen in a way that allows sufficient overlap between the units. In the context of this paper, fixed width values were chosen for each problem based on trial-and-error experiments and set to the values presented in Table II. The second stage of training an RBF network consists of finding the weights of the output layer by efficiently using linear algebra[6] to solve a quadratic error function in the weights. The optimal weight vector is determined using (5), where Φ is a design matrix of $m \times n$ elements containing the m predicted outputs for the n inputs, and y is an m-dimensional vector containing the training data outputs

$$W = (\Phi^t \Phi)^{-1} \Phi^{\mathrm{T}} y. \tag{5}$$

[6]The weights of the output layer are efficiently optimized by calculating the pseudo-inverse of the matrix of hidden unit activations.

TABLE I
OPTIMIZER CONFIGURATIONS

Size of population	100
Crossover operator	Simulated binary crossover (SBX) [34] Probability: 0.8
Mutation operator	Polynomial mutation probability: 1/n n = number of decision variables
Number of generations	NSGA-II: 100 NSGA-II/CAO: 50 SPEA2: 100 SPEA2/CAO: 50
Number of runs	10
Starting population	Same random population (different at each run)

TABLE II
NEURAL NETWORK AND STEP LENGTH CONFIGURATION

	RBF neural network				
	No of neurons	RBF Widths	h	τ	ε
ZDT1	80	1	5	60%	1.1
ZDT3	80	1	5	60%	1.1
ZDT6	80	1	5	60%	1.1
DTLZ2(3)	80	5	5	50%	1.2
DTLZ2(8)	80	5	8	30%	1.2
DTLZ2(12)	80	5	10	30%	1.2
WFG6(3)	80	5	5	40%	1.1
WFG9(3)	80	5	5	40%	1.1

In Table II, the number of neurons used for the RBF NNs, the values used for the initial step factor h, the CAO R_s, threshold τ, and the step length cooling factor ε are illustrated for each of optimization problem addressed. The values depicted in Table II are the best values (for CAO efficiency) for each of the test functions used. These values were determined based on trial-and-error experiments with different combinations of values for each of the parameters in the ranges below:

1) no of neurons: 50, 60, 70, 80, 90, and 100;
2) RBF widths: 0.5, 1, 2, 3, 5, 10;
3) h: 10, 8, 5, 2, 1.5, 0.8, 0.4, 0.1;
4) τ: 10%, 20%, 30%, 40%, 50%, 60%, 80%, 100%;
5) ε: 0.5, 1.1, 1.2, 1.5, 2, 5, 10.

Recall that the step factor h at generation $t = i + 1$ is increased using the following rule:

$$if\, R_S(t = i) < \tau \rightarrow h_{t=i+1} = h_{t=i} \times \varepsilon.$$

From the experience gained by running tests on these test functions, the following parameters were deemed to be the most suitable, in general, for application to subsequent problems:

Parameter Recommendations for the CAO:
Set $h = 10$, $\tau = 50\%$, $\varepsilon = 1.1$
Consult Nabney [28] for rule-of-thumb RBF parameters (no. of neurons and RBF width).

A. Performance Metrics

In [35], it was shown that there is no finite combination of unary metrics that can determine whether an approximation set "A" outperforms another approximation set "B." Zitzler *et al.* [35] also showed that binary indicators that compare the quality of one approximation set with another in terms of a certain criterion are suitable metrics for concluding that an approximation set is better than another in terms of the inspected criterion. The effectiveness of the CAO when tackling the bi-objective test functions (ZDT1, ZDT3, and ZDT6) and the DTLZ2 test functions with 3, 8, and 12 objectives is assessed by using two well-established binary metrics that simultaneously consider the convergence and the diversity requirements.

1) The *dominated distance metric* (DD-metric), which was originally conceived in [36], computes the dominated distance between two sets of objective vectors in the objective space. More closely, the DD-metric calculates the difference of dominated distances between two approximation sets "A" and "B" produced by MOEAs "A1" and "A2" in the objective space. The dominated distance between an approximation set "A" and an approximation set "B" (ddAB) is the sum of Euclidean distances between each solution A_i in "A" and the closest solution B_i which belongs to the subset of "B" that dominates A_i. The dominated distances ddAB and ddBA are calculated respectively, and their difference forms the value of DD-metric (A, B).
2) The coverage metric (*C-metric*) of Zitzler [36], which calculates the percentage of solutions in a certain approximation set that are dominated by equal to any solution in another competing approximation set.

The significance of the C-metric and the DD-metric results is also statistically analyzed by drawing boxplots, which illustrate the distribution of the metric values achieved (i.e., the 75 percentile, 25 percentile, median, and outliers values). The significance of the metric values is also analyzed using the Wilcoxon rank-sum test [42] for each of the experiments. The Wilcoxon rank-sum test is a nonparametric test that takes two independent samples of data and evaluates the hypothesis that the two samples come from distributions with equal medians. The Wilcoxon rank-sum test returns two values P and H. P is the probability of observing the null hypothesis, i.e., the two samples having the same median. Small values of P cast doubt on the validity of the null hypothesis. Moreover, the Wilcoxon rank-sum test is performed to return the result of the hypothesis test performed at the 0.05 significance level. The null hypothesis can only be rejected at the 5% level if the value of logic variable H is equal to 1.

The *hypervolume* metric [36] is also used to analyze the performance of the optimizers and their CAO-hybridized versions on the bi-objective functions. The hypervolume metric, also known as the S-metric or the Lebesgue integral, is a high-quality unary metric which illustrates the relative quality of an approximation set in terms of both desired criteria–convergence and diversity–by measuring the amount of objective space that the approximation set dominates. Unlike

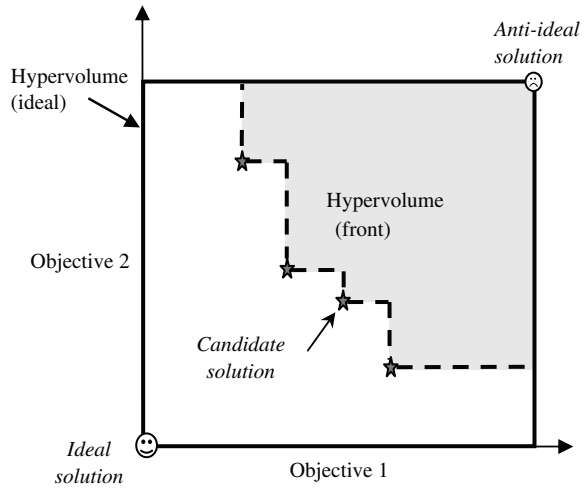

Fig. 7. Hypervolume metric (minimization problem assumed).

other unary metrics requiring some prior knowledge about the Pareto front or the targeted tradeoff surface, the computation of the hypervolume metric requires the proposal of an anti-ideal solution to act as a reference point. The values of the hypervolume metric can then be normalized in terms of the hypervolume measure of the ideal solution.

In Fig. 7, the hypervolume metric is illustrated on a bi-objective optimization problem for ease of visualization. In this, the values of the hypervolume metric are plotted against the total time(s) spent at each of the 10 runs of the MOEAs to illustrate their convergence extent versus their efficiency. This approach was previously adopted for the bi-objective test functions in [17] within a different benchmarking approach that assumes an infinite number of objective function evaluations. The hypervolume metric was not deployed as a performance metric on the *many*-objective optimization problems because of the well-known limitation of the metric's computational complexity, which is exponential in the number of objectives [38]. Deploying the hypervolume metric on the many-objective optimization problems was found to be impractical. It is very time consuming [especially for DTLZ2 (8) and (12)] to calculate this metric.

IV. Results

The performance of the CAO is investigated in this section. The effect of the introduced operator is examined using the specific performance metrics presented in Section III and by comparing the results achieved by NSGA-II and SPEA2 with the results achieved by their hybridized versions NSGA-II/CAO and SPEA2/CAO. A modified version of the CAO is also implemented for comparison. It is similar to the approach described by Gaspar-Cunha and Vieira [17] and uses an MLP NN to replace the RBF NN. The modified version of the CAO will be identified as CAO-MLP, while the promoted acceleration technique will be termed CAO-RBF or simply CAO.

The MLP configurations used when optimizing ZDT1, ZDT3, and ZDT6 were based on trial-and-error experiments, which found the same values as those used by Gaspar-Cunha and Vieira [19]. The number of hidden neurons and the learning rate of the MLP are (10, 0.2) for ZDT1, (20, 0.3) for ZDT3, and (10, 0.2) for ZDT6. At every generation of the MOEAs, 50 iterations[7] of the standard backpropagation algorithm [21], with a gradient descent optimization process, are executed for training the MLP NN and calculating its weights values. The number of hidden neurons and the learning rate of the MLPs used with DTLZ2 (3), WFG6 (3), and WFG9 (3) are (20, 0.3). For DTLZ2 (8) and (12) the number of hidden neurons and the MLP learning rate are, respectively, (30, 0.3) and (40, 0.3). The same values used for the step factor h, τ, and ε in the CAO-RBF are used with the CAO-MLP.

In order to evaluate the utility of the adaptive local search component of the CAO, seven versions of NSGA-II/CAO (and SPEA2/CAO) with different fixed step factors were executed for each of the test functions. The fixed step factors examined are $h = 10, 5, 2, 1.5, 0.8, 0.4,$ and 0.1. Boxplots and the Wilcoxon rank-sum test for the C-metric and the DD-metric results comparing the performance of NSGA-II and SPEA2 with their CAO-hybridized versions (seven with fixed step factors, and two with adaptive step factors—CAO-RBF and CAO-MLP) are produced.

The effect of the CAO correction step is also analyzed on the simple bi-objective test functions by comparing the results achieved by NSGA-II and its CAO-hybridized versions with no correction step.

A. Bi-objective Test Functions: Results

In Fig. 8, the values achieved for the S-metric at each of the 10 runs of NSGA-II, NSGA-II/CAO-RBF, and NSGA-II/CAO-MLP are illustrated. The three MOEAs were optimizing the convex test function ZDT1. The S-metric values achieved at each execution of the algorithms are plotted against the total time spent by each algorithm at the designated execution. The reference point used for calculating the S-metric consisted of the point whose coordinates corresponded to the worst values achieved for each objective by the algorithms combined and within 10 runs.

From Fig. 8, it can be deduced that in 9 out of 10 runs NSGA-II/CAO-RBF achieved larger values for the S-metric than NSGA-II, resulting in improved convergence and diversity. Within just 50 generations per run and a fixed budget of objective function evaluations, the S-metric values achieved by NSGA-II/CAO-RBF were closer to the solid line, which represents the S-metric value for the true Pareto front. Moreover, it was observed that, despite optimizing a straightforward and computationally cheap problem (ZDT1), the time spent by NSGA-II/CAO-RBF at each of the ten runs was comparable to the time spent by NSGA-II. This observation indicates that the CAO-RBF was improving the results achieved by NSGA-II for very little additional cost.

Thus the use of CAO-RBF is practical for addressing a wide variety of problems and not just restricted to computationally expensive optimization problems. On the other hand, NSGA-II/CAO-MLP requires much more time (five times longer)

[7]This is performed for "early stopping" the MLP training process in order to avoid overfitting.

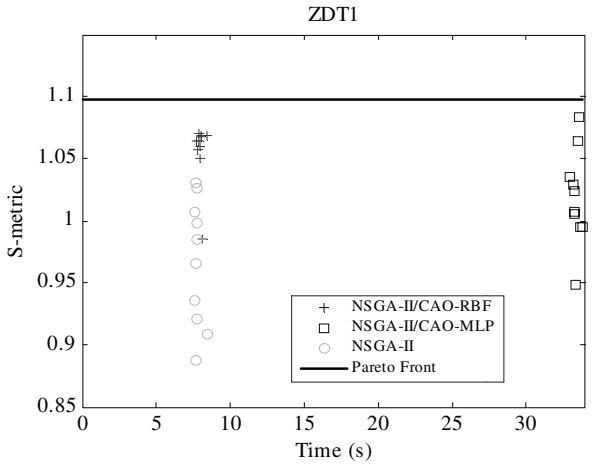

Fig. 8. S-metric values achieved by NSGA-II, NSGA-II/CAO-RBF, and NSGA-II/CAO-MLP on ZDT1 at each of the 10 runs.

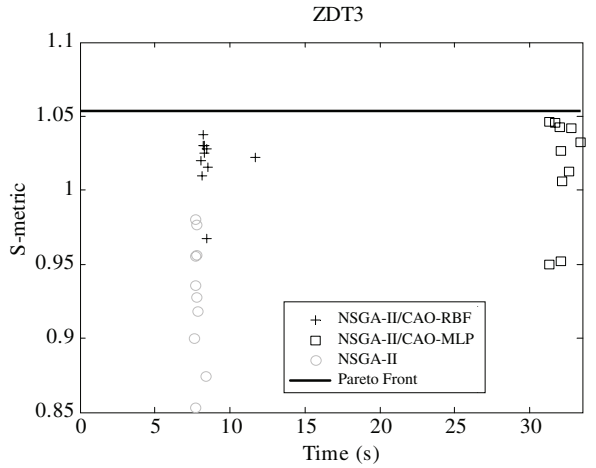

Fig. 9. S-metric values achieved by NSGA-II, NSGA-II/CAO-RBF, and NSGA-II/CAO-MLP on ZDT3 at each of the 10 runs.

per run while presenting some improved results and, for a few runs, some remarkable and near-optimal values for the S-metric. A more consistent behavior for NSGA-II/CAO-MLP can be achieved by optimizing the efficiency of the MLP and training it over more iterations (epochs) or using more sophisticated training algorithms.[8] Such improvement, however, can only be achieved at the expense of increasing the computational time of the algorithm. Such a tradeoff is likely to be unacceptable within the context of straightforward optimization problems such as the ZDTs, but desirable when dealing with computationally expensive problems.

Nevertheless, from Fig. 8 it is clear that on a straightforward and computationally cheap problem such as ZDT1, and within the same budget of objective function evaluations, NSGA-II/CAO-RBF is accelerating the convergence of NSGA-II without requiring any significant increase in computational effort.

The S-metric values achieved by NSGA-II, NSGA-II/CAO-RBF, and NSGA-II/CAO-MLP for the discontinuous and the nonuniform test functions ZDT3 and ZDT6 are illustrated in Figs. 9 and 10, and, except for the running times, in Fig. 10, similar results are observed. The same observations that highlighted the utility of the CAO (either the MLP version or the RBF version in particular) on ZDT1 are observed for ZDT3 and ZDT6.

[8]As part of the trial-and-error experiments for setting the NN parameters, the following MLP training algorithms were examined: conjugate gradient descent, Newton, and Quasi-Newton methods.

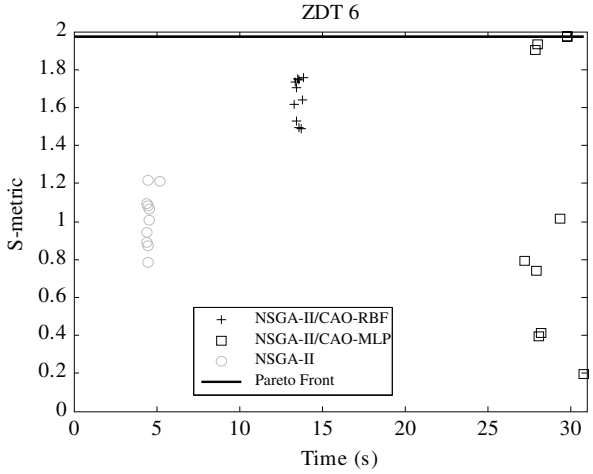

Fig. 10. S-metric values achieved by NSGA-II, NSGA-II/CAO-RBF, and NSGA-II/CAO-MLP on ZDT6 at each of the 10 runs.

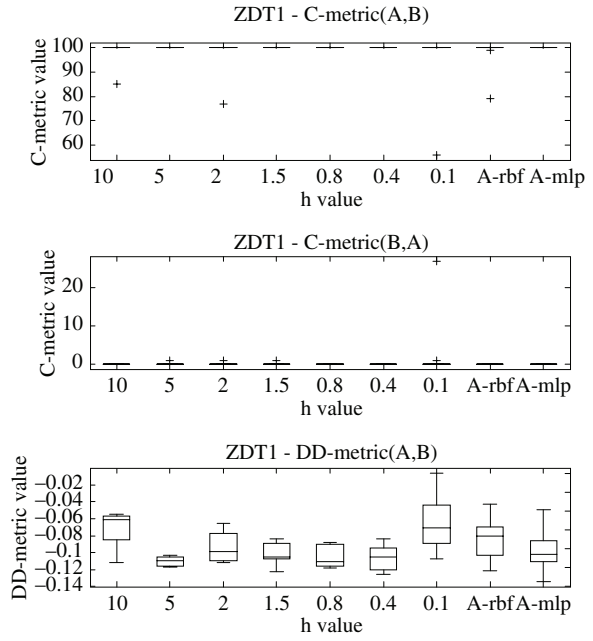

Fig. 11. Boxplots of the C-metric and DD-metric values achieved on ZDT1-(fixed and adaptive local search).

In Figs. 11 and 13, boxplots of the C-metric and the DD-metric values achieved for ZDT1, ZDT3, and ZDT6 are illustrated. In these figures, optimizer "A" denotes NSGA-II/CAO and "B" is NSGA-II. The first seven columns represent the boxplots of the C-metric and the DD-metric values comparing NSGA-II and NSGA-II/CAO-RBF with fixed step factors for the local search. The values of the fixed step factor for each of the seven instances of NSGA-II/CAO-RBF are depicted on the x-axis. The last two entries on the x-axis show the results when NSGA-II/CAO has adaptive step factors for the local search (A-rbf = adaptive CAO-RBF, A-mlp = adaptive

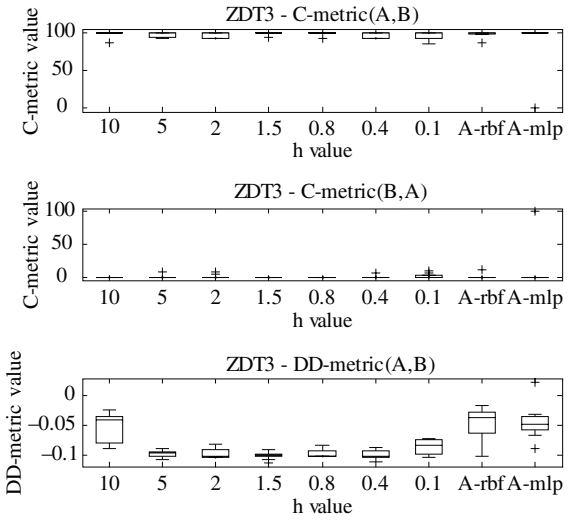

Fig. 12. Boxplots of the C-metric and DD-metric values achieved on ZDT3- (fixed and adaptive local search).

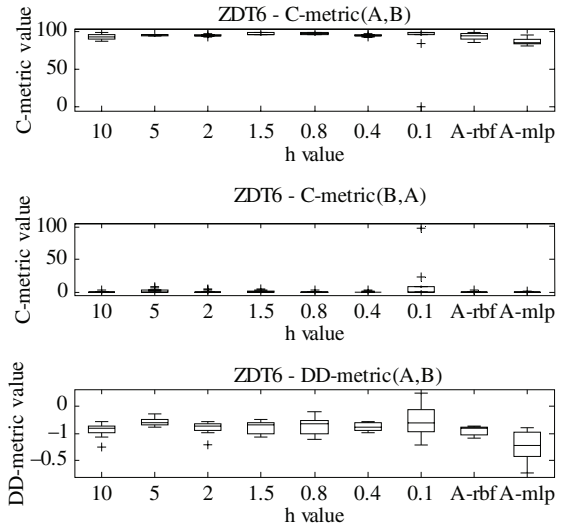

Fig. 13. Boxplots of the C-metric and DD-metric values achieved on ZDT6- (fixed and adaptive local search).

CAO-MLP). Recall that Gaspar-Cunha *et al.* [17, 18] did not propose an adaptive step factor, but did propose MLP as the NN approach to be adopted. Inclusion of A-mlp in experiments is to compare its performance with A-rbf, both using an adaptive step factor.

The upper sections of Figs. 11–14 illustrate that the approximation sets produced by the nine different versions of NSGA-II/CAO were overall achieving a near-optimal cover (100%) of the results produced by NSGA-II. Except for some outlier values, the boxplots of the C-metric values for all nine versions of NSGA-II/CAO were mostly collapsed (nearly overlapping values for the median, 25th, and 50th percentile) around the value "100" indicating a very consistent performance for NSGAII/CAO. In Fig. 13, however, NSGA-II/CAO-MLP achieves a lower median value (around 80%) for the C-metric on ZDT6 compared with the remaining eight algorithms. The approximation sets produced by the stand-alone NSGA-II were on the other hand consistently achieving zero coverage of the results produced by the nine versions of NSGA-II/CAO for ZDT1, ZDT3, and ZDT6. This is highlighted by the collapsed boxplots around the value zero, and presented in the middle sections of Figs. 11–13.

The DD-metric is computed for ZDT1, ZDT3, and ZDT6 and boxplots of the results are shown for each algorithm in the lower sections of Figs. 11–13. Similar to the C-metric, the DD-metric is a binary metric that highlights whether an approximation set resulting from an algorithm "A" is better than another approximation set resulting from an algorithm "B." A negative DD-metric value denotes that the first input of the metric [e.g., Algorithm A in DD-Metric (A,B)] is better than and dominates most or part of its second input (Algorithm B). Similar to the C-metric results, the boxplots of the DD-metric values produced for each algorithm consist of negative values illustrating a better performance for the CAO-hybridized versions of NSGA-II. Overall, the boxplots of the DD-metric values achieved at each of the 10 runs of the algorithms were consistent for the three bi-objective test functions.

The Wilcoxon rank-sum test was performed to assess the significance of the C-metric and the DD-metric results shown in Figs. 11–13 for the bi-objective test functions. The results of the Wilcoxon rank-sum test assessing the significance of the C-metric and the DD-metric results are shown in Tables III and IV, respectively. From these tables it can be observed that the values of P were consistently equal to (or very close to) the values 0 and 1 for all three bi-objective test functions, confirming the statistical significance of the C-metric and DD-metric results.

From the results shown in Figs. 11–13 and Tables III and IV CAO-RBF and CAO-MLP, both proved to be competent, introducing improvements to the results achieved by NSGA-II. Moreover, for the bi-objective test functions investigated, the CAO proved to be robust in terms of the local search step factor h, and both the adaptive and the fixed approach for the local search improved upon the results achieved by NSGA-II for the same number of objective function evaluations.

The use of a RBF NN within the CAO is shown, however, to be more practical than a MLP NN, due to its much faster training process, which makes it efficient for deployment within a convergence acceleration technique. Similar observations are made when the CAO is hybridized with SPEA2. The results achieved for the S-metric, C-metric, and DD-metric when the CAO is hybridized with SPEA2 are illustrated in the Appendix, which again demonstrate the impact of the CAO on one of the best-performing MOEAs.

Additional experiments were produced to assess the significance of the correction step used within the CAO. Accordingly, the nine different versions of NSGA-II/CAO used in Figs. 11–13 were re-run on ZDT1, ZDT3, and ZDT6. However, in this new set of experiments, the CAO correction step was deactivated and the results produced by the different versions of NSGA-II/CAO (with fixed h, and adaptive h with RBF and MLP) were compared with the results produced by the standalone NSGA-II.

Boxplots of the C-metric and the DD-metric results comparing NSGA-II and NSGA-II/CAO with an inactive correction

TABLE III
WILCOXON RANK-SUM TEST OF THE C-METRIC VALUES

	ZDT1		ZDT2		ZDT3	
	P	*H*	*P*	*H*	*P*	*H*
10	0	1	0	1	0	1
5	0	1	0	1	0	1
2	0	1	0	1	0	1
1.5	0	1	0	1	0	1
0.8	0	1	0	1	0	1
0.4	0	1	0	1	0	1
0.1	0	1	0	1	0.	1
A-RBF	0	1	0	1	0	1
A-MLP	0	1	4.10^{-4}	1	0.005	1

TABLE IV
WILCOXON RANK-SUM TEST OF THE DD-METRIC VALUES

	ZDT1		ZDT2		ZDT3	
	P	*H*	*P*	*H*	*P*	*H*
10	0	1	0	1	0	1
5	1.10^{-4}	1	1.10^{-4}	1	0	1
2	1.10^{-4}	1	1.10^{-4}	1	0	1
1.5	0	1	0	1	0	1
0.8	0	1	0	1	0	1
0.4	0	1	0	1	0	1
0.1	2.10^{-4}	1	1.10^{-4}	1	2.10^{-3}	1
A-RBF	0	1	0	1	0	1
A-MLP	0	1	1.10^{-4}	1	0	1

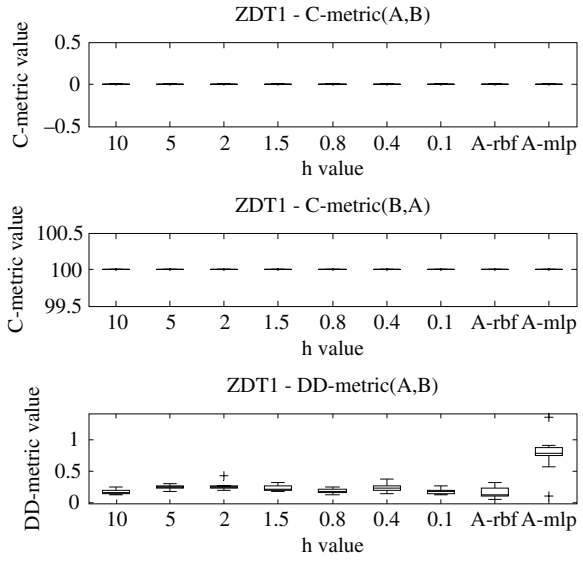

Fig. 14. Boxplots of the C-metric and DD-metric values achieved on ZDT1—No correction step.

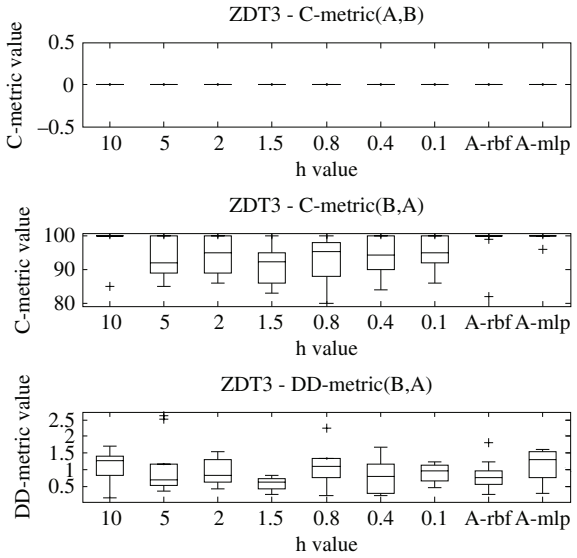

Fig. 15. Boxplots of the C-metric and DD-metric values achieved on ZDT3—No correction step.

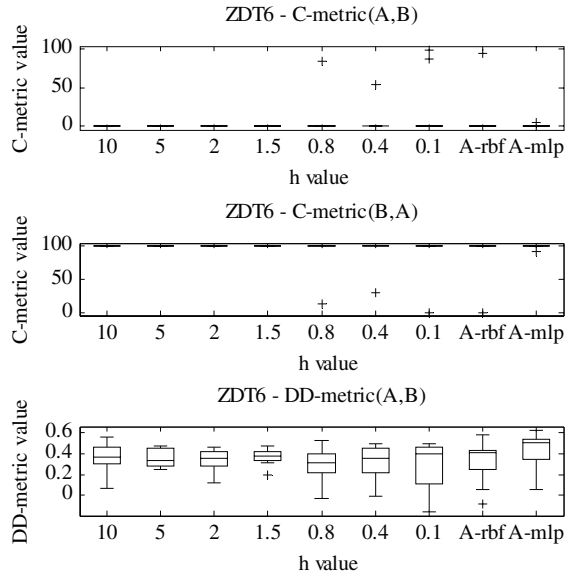

Fig. 16. Boxplots of the C-metric and DD-metric values achieved on ZDT6—No correction step.

step are shown in Figs. 14–16. From these figures, it is clear that the results previously illustrated in Figs. 11–13 are reversed (i.e., now zero coverage and positive DD-metric values are produced by NSGA-II/CAO for ZDT1, ZDT3, and ZDT6), and NSGA-II now outperforms NSGA-II/CAO. The outcome of these experiments clearly demonstrates the importance of the correction step, even on simple nonconstrained MOPs such as the ZDTs.

B. Scalable Test Function DTLZ2: Results

Figs. 17–19 and Tables VI and VII, illustrate the results highlighting the effect of the CAO on optimization problems with a larger number of objectives. The scalable test function DTLZ2 with 3, 8, and 12 objectives was chosen to investigate

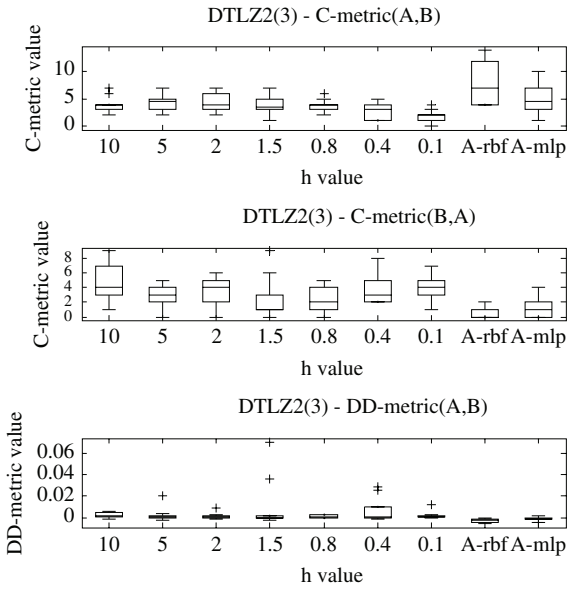

Fig. 17. Boxplots of the C-metric and DD-metric values achieved on DTLZ2 (3).

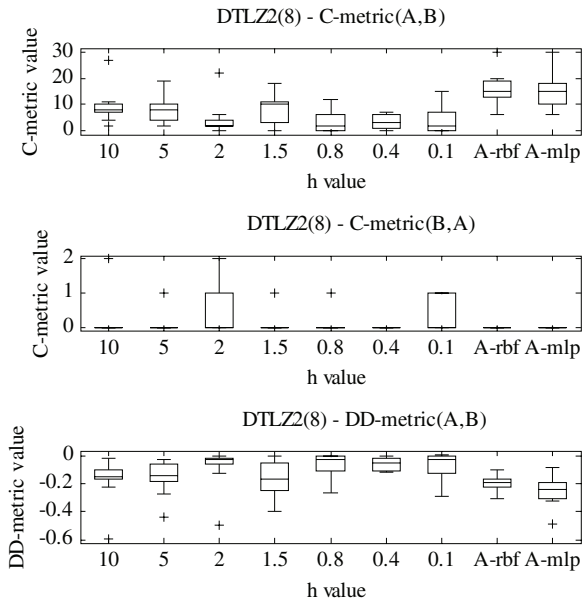

Fig. 18. Boxplots of the C-metric and DD-metric values achieved on DTLZ2 (8).

the performance of the CAO. In a manner similar to the experiments carried out on the bi-objective problems, the effect of the CAO was underlined by contrasting NSGA-II with its CAO-hybridized counterparts (NSGA-II/CAO-RBF and NSGA-II/CAO-MLP and NSGA-II/CAO with different fixed step factors h).

In Table V, the computational time (mean and median values) expended for the 10 runs of the algorithms optimizing DTLZ2 (3), (8), and (12) is illustrated. The running time of NSGA-II/CAO-MLP was on average around three times longer than the running time of NSGA-II/CAO-RBF. The use of simple and computationally cheap test functions (ZDTs and DTLZ2) for assessing the CAO has helped emphasize the efficiency of CAO-RBF over CAO-MLP.

The experiments presented in Figs. 17–19 show that the fronts achieved by the CAO-hybridized versions of NSGA-II (running for just 50 generations) frequently achieve a higher coverage compared to the coverage achieved by NSGA-II in 100 generations.

Moreover, for the *many*-objective optimization problems, it became more apparent that the adaptive approach for the step factor h was, in general, performing better than the fixed step factor approach, introducing improvements to NSGA-II for DTLZ2 (3), (8), and (12). Over the 10 runs of the algorithms, NSGA-II/CAO-RBF produced an average of 6, 18, and 3.1% coverage of the results achieved by NSGA-II for the 3, 8, and 12 objective versions of DTLZ2, respectively.

On the other hand, NSGA-II only achieved an average of 0.09, 0.01, and 0.02%, coverage of the results achieved by NSGA-II/CAO-RBF for DTLZ2 (3), (8), and (12) including several runs with 0% coverage. NSGA-II/CAO-MLP has similarly produced a coverage of NSGA-II results that is higher than the coverage achieved by NSGA-II on all three versions of DTLZ2.

On average, NSGA-II/CAO-MLP covered 4, 15.5, and 2.8% of the results produced by NSGA-II for DTLZ2 (3),

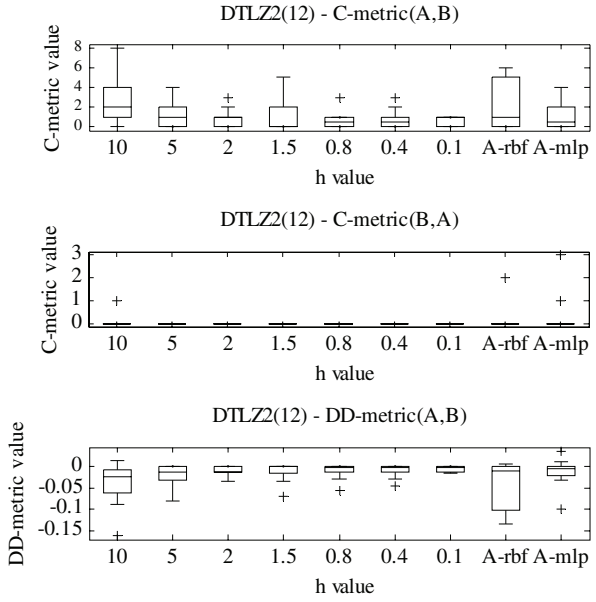

Fig. 19. Boxplots of the C-metric and DD-metric values achieved on DTLZ2 (12).

TABLE V
DTLZ2 COMPUTATIONAL TIMES

		Computational time per 10 executions (s)		
		NSGA-II	NSGA-II/ CAO-RBF	NSGA-II/ CAO-MLP
DTLZ2 (3)	Median	5.1	10.1	32.2
	Mean	5.9	10.8	34.8
DTLZ2 (8)	Median	10.2	12.4	36.6
	Mean	10.8	12.6	37.3
DTLZ2 (12)	Median	11.4	16.2	40.2
	Mean	12.1	16.8	41.1

TABLE VI
WILCOXON RANK-SUM TEST OF THE C-METRIC VALUES

	DTLZ2 (3)		DTLZ2 (8)		DTLZ2 (12)	
	P	*H*	*P*	*H*	*P*	*H*
10	0.5	0	0	1	0	1
5	0.2	0	0.01	1	0	1
2	0.5	0	0.02	1	0	1
1.5	0.2	0	0.01	1	0.5	0
0.8	0.1	0	0.05	1	1	0
0.4	0.3	0	0.01	1	0.3	0
0.1	0.5	0	0.04	1	0.2	0
A-RBF	0	1	0	1	0	1
A-MLP	0	1	0	1	0	1

TABLE VII
WILCOXON RANK-SUM TEST OF THE DD-METRIC VALUES

	DTLZ2 (3)		DTLZ2 (8)		DTLZ2 (12)	
	P	*H*	*P*	*H*	*P*	*H*
10	0.02	1	0	1	0	1
5	0.2	0	0	1	0	1
2	0.15	0	0	1	0	1
1.5	0.8	0	0	1	0.2	0
0.8	0.6	0	0.02	1	0.5	0
0.4	0.2	0	0.01	1	0.2	0
0.1	0.2	0	0.1	0	0.2	0
A-RBF	0	1	0	1	0	1
A-MLP	0	1	0	1	0	1

(8), and (12), respectively, while NSGA-II only achieved an average coverage of 1.8% for DTLZ2 (3) and 0.001% for DTLZ2 (8) and (12).

Based on the C-metric results (upper and middle sections of Figs. 17–19), it can be seen that, when hybridized with NSGA-II, the two adaptive versions of CAO were producing higher C-metric values compared with the stand-alone NSGA-II. Similar observations apply to the DD-metric (lower sections in Figs. 17–19). The DD-metric has consistently produced results (<0) which favor NSGA-II/CAO (RBF and MLP) over NSGA-II for all dimensions of the problems investigated.

The significance of the C-metric and the DD-metric results achieved for, the DTLZ2 test functions is highlighted in Tables VI and VII, respectively. The values of the Wilcoxon rank-sum test outputs P and H were zero and one for NSGA-II/CAO using an adaptive step factor, both for RBF and MLP NNs. The benefit of using an adaptive step factor is clear when these rank-sum results are compared with those for fixed step sizes.

The biggest improvements introduced by the CAO in terms of coverage and dominated distance measures were exhibited for the 8-objective version of DTLZ2 (more than 10% coverage of NSGA-II solutions was achieved alongside a median value of -200×10^{-3} for the DD metric). Similar to NSGA-II/CAO, when the CAO is hybridized with SPEA2, SPEA2/CAO has outperformed SPEA2 on all three versions of DTLZ2 (see Appendix).

Further experiments were undertaken in an attempt to quantify the extent of superiority of the CAO-hybridized optimizers. It was noted that, on average, the population size of NSGA-II and SPEA2 must be increased to a minimum of 150 individuals (1.5 × the population size of NSGA-II/CAO and SPEA2/CAO) in order to match the quality of the fronts achieved by their hybridized counterparts. Thus, SPEA2 and NSGA-II require more objective function evaluations (around 50% more evaluations) to match the performance of their CAO-hybridized equivalent optimizer. This conclusion holds for all the test functions used in this paper. The set of experiments conducted in this section highlights the benefits of the CAO in general and the CAO-RBF in particular and demonstrates the improvement it confers to two of the most established MOEAs.

Finally, it is observed that the C-metric and DD-metric results for DTLZ problems were lower than the results obtained for the ZDT functions. This was anticipated since the proportion of nondominated solutions increases with increasing objectives.

TABLE VIII
WFG6 (3) AND WFG9 (3) COMPUTATIONAL TIME

	Computational time per 10 executions (s)			
		NSGA-II	NSGA-II/ CAO-RBF	NSGA-II/ CAO-MLP
WFG6 (3)	Median	32.4	27.1	78.9
	Mean	37.3	27.9	87.4
WFG9 (3)	Median	29.5	26.3	57.2
	Mean	34.2	29.1	57.9

C. Nonseparable Test Functions WFG6 (3) and WFG9 (3): Results

In this section, the utility of the CAO is investigated on two nonseparable test functions WFG6 and WFG9. Three objective instances of these scalable test functions are used [WFG6(3) and WFG9(3)]. WFG9 in particular is used to assess the significance of using local models, which approximates the mapping from the objective space to the decision space at a certain generation, as opposed to using global models which try to capture the overall mapping. The use of RBF within the CAO is practical for building such local models and overcomes the problem of training the NN with conflicting data due to possible one-to-many mappings from the objective space to the decision space.

In Table VIII, the median and mean computational time spent by NSGA-II, NSGA-II/CAO-RBF, and NSGA-II/CAO-MLP over ten runs optimizing WFG6(3) and WFG9(3) are shown. Again, the efficiency of using CAO-RBF was apparent, and NSGA-II/CAO-RBF required even less computational time than the stand-alone NSGA-II. NSGA-II/CAO-MLP, on the other hand, required at least twice the computational time required by NSGA-II and NSGA-II/CAO-RBF for optimizing WFG6 and WFG9.

Similar to the results shown in the previous sections, in Figs. 20 and 21, the boxplots of the C-metric and the DD-metric results achieved for each of the algorithms optimizing WFG6 and WFG9 are illustrated.

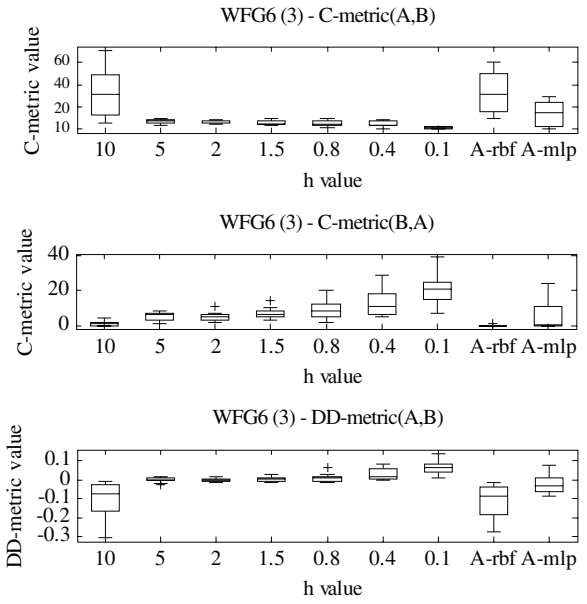

Fig. 20. Boxplots of the C-metric and DD-metric values achieved on WFG6 (3) (fixed and adaptive local search).

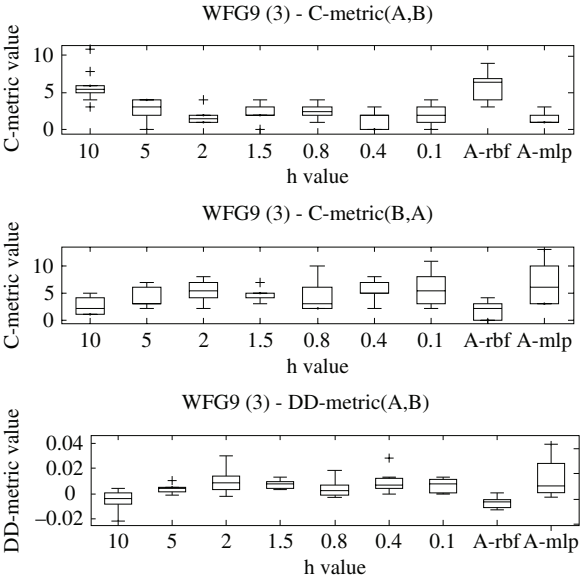

Fig. 21. Boxplots of the C-metric and DD-metric values achieved on WFG9 (3) (fixed and adaptive local search).

For the nonseparable test function WFG6, NSGA-II/CAO-RBF, NSGA-II/CAO-RBF with fixed step factor $h = 10$, and NSGA-II/CAO-MLP offered the most improvements to the results achieved by NSGA-II.

In particular, NSGA-II/CAO-RBF achieved more than 20% coverage of the results achieved by NSGA-II, with the latter almost covering none of the solutions produced by NSGA-II/CAO-RBF. The DD-metric values achieved by NSGA-II/CAO-RBF provided the best results when compared to NSGA-II and the remaining hybrid algorithms. The significance of these results was assessed by calculating the Wilcoxon rank-sum test whose results are shown in Table IX

TABLE IX
WILCOXON RANK-SUM TEST OF THE C-METRIC VALUES

	WFG6 (3)		WFG9 (3)	
	P	*H*	*P*	*H*
10	0.03	1	0	1
5	0.2	0	0.06	0
2	0.02	1	0.04	1
1.5	0.01	1	0.03	1
0.8	0.4	0	0.65	0
0.4	0.46	0	0.01	1
0.1	0.02	1	0.01	1
A-RBF	0	1	0	1
A-MLP	0	1	0	1

TABLE X
WILCOXON RANK-SUM TEST OF THE DD-METRIC VALUES

	WFG6 (3)		WFG9 (3)	
	P	*H*	*P*	*H*
10	0	1	0.02	1
5	0.7	0	0.01	1
2	0.03	1	0.01	1
1.5	0.15	0	0.01	1
0.8	0.3	0	0.19	0
0.4	0.5	0	0.01	1
0.1	0	1	0.01	1
A-RBF	0	1	0	1
A-MLP	0	1	0	1

(note: $P = 0$ and $H = 1$ for NSGA-II/CAO-RBF and NSGA-II/CAO-MLP). Again, the benefit of using an adaptive step factor is clear when these rank-sum results are compared with those for fixed step sizes.

In Fig. 21, the results achieved for the nonseparable test function WFG9(3) are presented. It was notable that NSGA-II/CAO-MLP was, in this case, performing worse than the remaining algorithms, including NSGA-II (positive DD-metric values and a very low coverage of NSGA-II results). Due to the iterative nature of the MLP training process, NSGA-II/CAO-MLP needed to continuously train the NN to achieve good predictions. As a result, NSGA-II/CAO-MLP was training the MLP to model the global mapping from the objective space to the decision space which, in the WFG9 case, was a one-to-many mapping. This led to the deterioration of the MLP prediction quality and consequently the deterioration of the end results produced by NSGA-II/CAO-MLP. NSGA-II/CAO-RBF, on the other hand, performed well and achieved the highest coverage and the lowest DD-metric values. The significance of the results shown in Fig. 21 is also highlighted in Table X by calculating the Wilcoxon rank-sum test ($P = 0$ and $H = 1$ for NSGA-II/CAO-RBF and NSGA-II/CAO-MLP).

Similar experiments to the ones shown in Figs. 20 and 21 were conducted by replacing NSGA-II with SPEA2 and similar results highlighting the efficiency of the CAO-RBF were achieved (see Appendix).

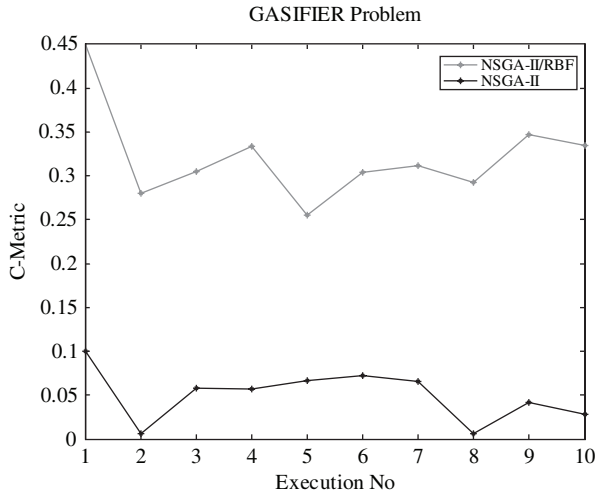

Fig. 22. C-metric values achieved by NSGA-II/CAO-RBF and NSGA-II on the gasifier problem.

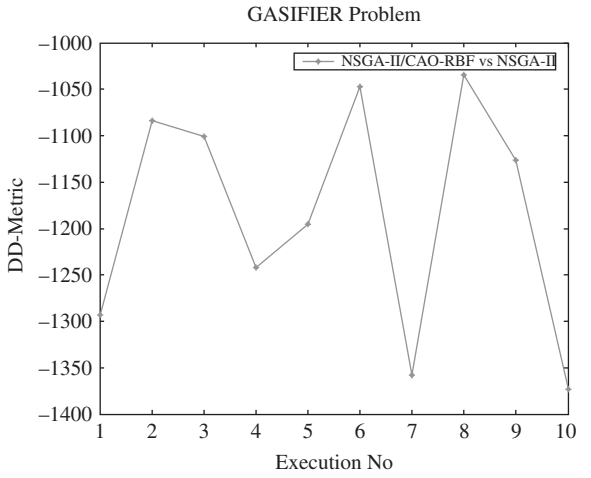

Fig. 23. DD-metric values achieved by NSGA-II/CAO-RBF and NSGA-II on the gasifier problem.

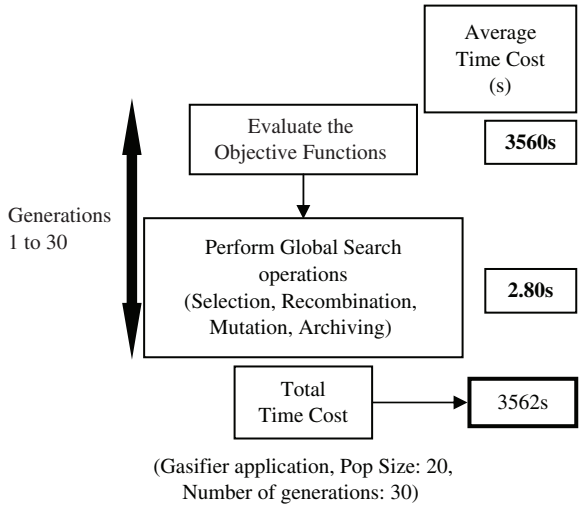

Fig. 24. NSGA-II computational effort.

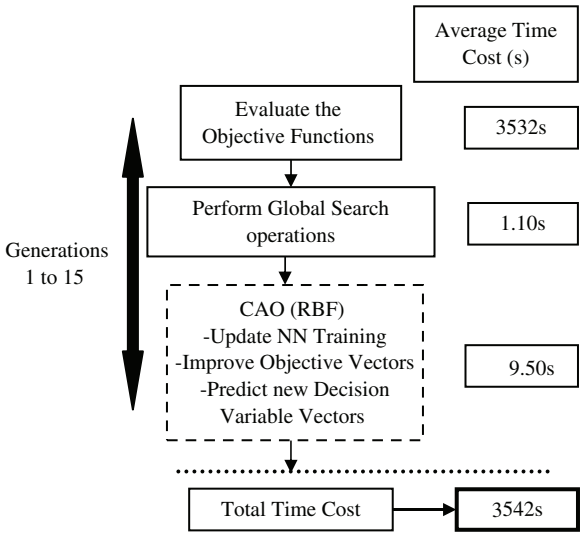

Fig. 25. NSGA-II/CAO computational effort.

D. Running Times

Experiments were undertaken to compare the run-time efficiency of NSGA-II and NSGA-II/CAO, particularly to assess the additional effort required as a result of the NN mapping employed by the CAO. These experiments were undertaken on a real-world problem comprising cost-intensive objective function evaluations. Computational times were averaged over ten runs.

The additional computational effort required by the CAO-RBF is reported. It should be noted that such computational effort measurements depend on the hardware/software resources available. In this paper, MATLAB was used for implementing, executing, and testing all the optimization frameworks presented in this paper. Furthermore, all the experiments were undertaken on a Pentium 4 machine with 512 megabyte of RAM. NETLAB [28], which is an open source neural network toolbox for use with MATLAB, was used for implementing, training, and validating the ANN used in the CAO context.

The computational effort required for training the neural network is influenced by the number of inputs, outputs, weights, and parameters, rather than the complexity of the objective function being solved by the hosting MOEA. The CAO was hybridized with NSGA-II in an optimization framework attempting to solve a benchmark control system design problem involving a gasifier [39]. This is a relatively expensive (computationally) problem (for example, 14 objectives) chosen to set the CAO computational effort in context with the computational demands of evaluating objectives for a real-world problem.

In Fig. 22, the coverage achieved by NSGA-II/CAO and NSGA-II at each of the ten executions is shown. NSGA-II/CAO covered an average of 34% of the solutions found by NSGA-II, while NSGA-II only covered an average of 5% of the solutions found by its CAO-hybridized counterpart. In a similar way, the DD-metric (negative) values presented in Fig. 23 highlight the outperformance of NSGA-II/CAO over NSGA-II for all ten runs.

The computational effort measurements (averaged over ten runs) of the major components of NSGA-II and NSGA-II/CAO

optimizing the gasifier problem for 30 and 15 generations, respectively, are presented in Figs. 24 and 25. Except for the reduced number of generations (15 for NSGA-II/CAO and 30 for NSGA-II), population size (= 20), and the initial value of the step factor h(= 20), the same configuration used for NSGA-II and NSGA-II/CAO in the previous sections was deployed.

Note that in [39], a larger population size and number of generations were deployed. However, the configuration used in this section was deemed sufficient since the goal of the presented experiments was to contrast the efficiency of NSGA-II and NSGA-II/CAO within a limited budget of objective evaluations rather than solving the gasifier problem itself.

The results presented in Figs. 24 and 25 demonstrate that the total computational time required by NSGA-II and NSGA-II/CAO is comparable. In fact, NSGA-II/CAO (3542s) required 20 s less than NSGA-II (3562s).

Moreover, compared with the total computational time used for calculating the gasifier's objective functions (3532 s), the total computational time spent for training and validating the RBF neural network within the CAO is negligible (9.5 s).

V. Conclusion

A portable CAO has been proposed for incorporation into existing algorithms for evolutionary multiobjective optimization. Two leading MOEAs have been hybridized through introduction of the CAO and tested on a variety of recognized test problems and a real-world application. These problems consisted of convex, discontinuous, nonseparable, nonuniform, and multimodal objective functions, with the number of objectives ranging from 2 to 14. In all cases, the introduction of the CAO led to improved results for comparable numbers of function evaluations. This operator works by suggesting improved solutions in objective space and using neural network mapping schemes to predict the corresponding solution points in decision variable space.

This paper builds and improves on previous work by Gaspar-Cunha *et al*. [18]–[20] and Adra *et al*. [8]. The main improvements include the use of an adaptive step factor for the local improvement in objective space. In this way, we can avoid the problem of introducing points that reside outside the feasible region or the reliable zone of NN prediction, hence detecting novelty and extrapolation. Another improvement is the use of a RBF NN within the CAO, instead of the MLP NN used in earlier work of Gaspar-Cunha *et al*. and Adra *et al*. The use of a RBF NN within the CAO is more efficient and practical, due to its faster training process and its transparency with respect to the training data. When using an RBF NN within the CAO, an unsupervised training process for the NN parameters (RBF widths and centers) can be applied and is shown to be efficient and competitive with MLPs, iteratively trained with nonlinear optimizers such as the gradient descent. In contrast to the MLP, the use of a RBF NN within an MOEA convergence accelerator makes it practical to use on a variety of problems, rather than being restricted to computationally expensive problems. The CAO proposed in this paper also includes a correction step, whereby the feasibility of the predicted solutions is checked and the exact objective values are evaluated in order to maintain the fidelity of the solutions to the exact model.

Whereas Gaspar Cunha *et al*. [18]–[20] limit their evaluation of their approach to 2-objective problems, this paper considers in detail the implications of using the operator in many-objective problems (ranging from 3 to 14 objectives). When deploying an active strategy for promoting diversity within a slowly converging process to the Pareto front, the convergence process of a MOEA can be hampered and delayed, especially in optimization problems with many competing objectives. Due to the convergence acceleration caused by the CAO, the MOEA selection criteria progressively place more emphasis on the active diversification mechanisms. However, the increasing emphasis of the active diversification mechanisms is manifested at converged and near-optimal regions of the search space rather than at remote and suboptimal regions.

Finally, the paper considers the computational effort involved in incorporating the CAO in the optimization process. It is important to recognize that the CAO introduces additional computational effort through the requirement to train the neural network. When using an RBF NN, this computational effort is small even when compared with the execution time associated with computing a function, such as ZDT, which is inexpensive to compute. Compared with the RBF NN, it has been shown that using an MLP within the CAO leads to increased computational effort. In a paper on a real-world example where objective function computation is nontrivial, it was shown that the computational effort required by the hybridized scheme was almost the same as that required by the standard scheme, while the hybridized scheme obtained superior results.

VI. Appendix

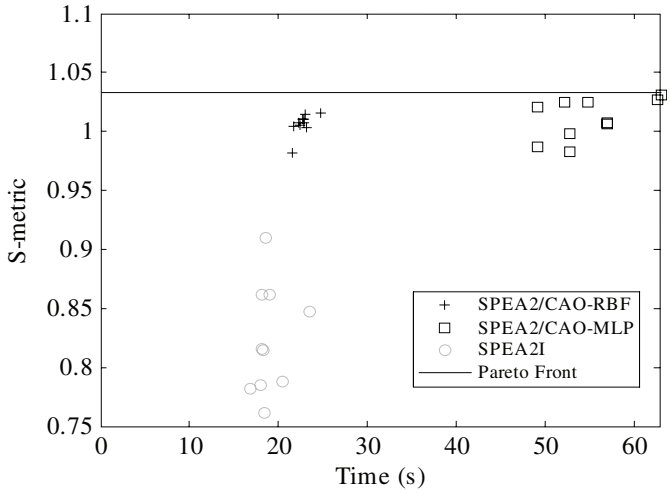

Fig. 26. S-metric values achieved by SPEA2, SPEA2/CAO-RBF, and SPEA2/CAO-MLP on ZDT1 at each of the 10 runs.

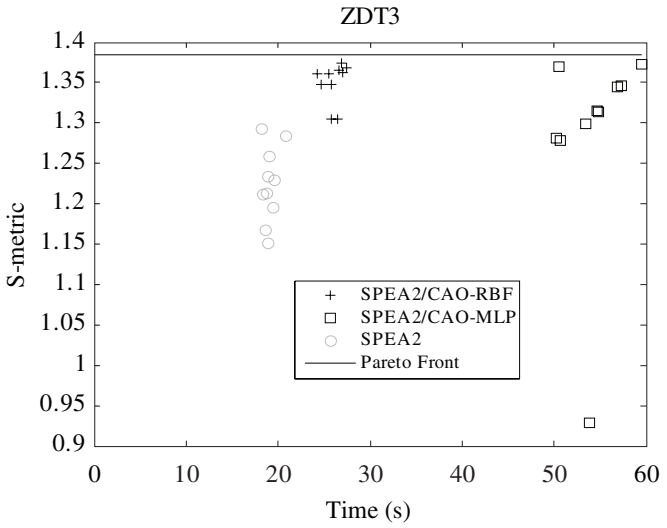

Fig. 27. S-metric values achieved by SPEA2, SPEA2/CAO-RBF, and SPEA2/CAO-MLP on ZDT3 at each of the 10 runs.

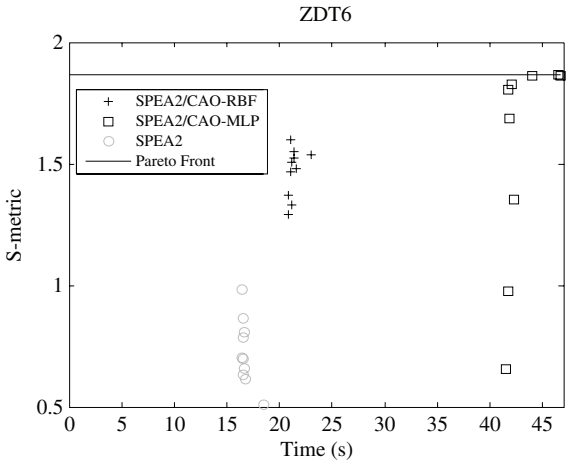

Fig. 28. S-metric values achieved by SPEA2, SPEA2/CAO-RBF and SPEA2/CAO-MLP on ZDT6 at each of the 10 runs.

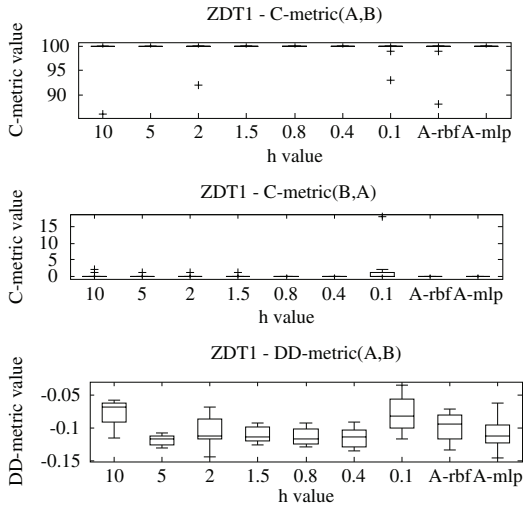

Fig. 29. Boxplots of the C-metric and DD-metric values achieved on ZDT1- (fixed and adaptive local search).

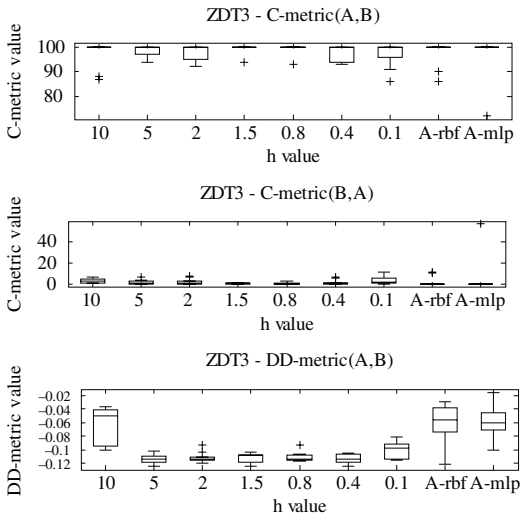

Fig. 30. Boxplots of the C-metric and DD-metric values achieved on ZDT3- (fixed and adaptive local search).

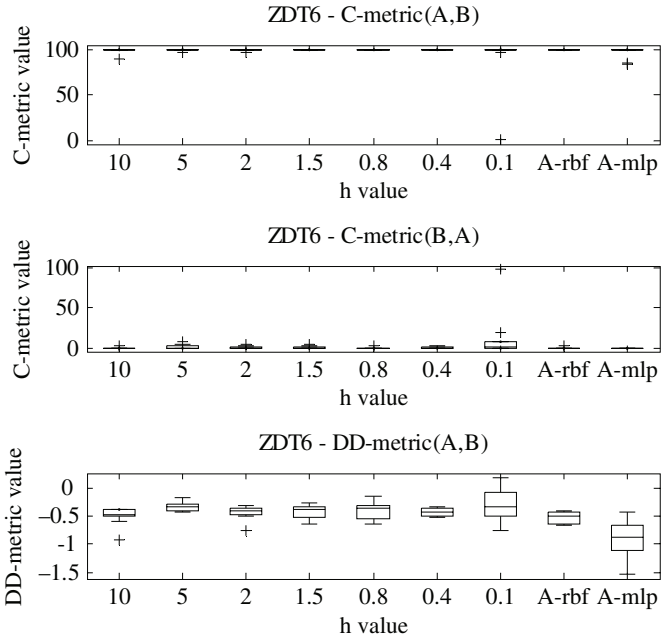

Fig. 31. Boxplots of the C-metric and DD-metric values achieved on ZDT6- (fixed and adaptive local search).

TABLE XI
WILCOXON RANK-SUM TEST OF THE C-METRIC VALUES

	ZDT1		ZDT2		ZDT3	
	P	*H*	*P*	*H*	*P*	*H*
10	0	1	0	1	0	1
5	0	1	0	1	0	1
2	0	1	0	1	0	1
1.5	0	1	0	1	0	1
0.8	0	1	0	1	0	1
0.4	0	1	0	1	0	1
0.1	0	1	0	1	5×10^{-4}	1
A-RBF	0	1	0	1	0	1
A-MLP	0	1	0	1	0	1

TABLE XII
WILCOXON RANK-SUM TEST OF THE DD-METRIC VALUES

	ZDT1		ZDT2		ZDT3	
	P	*H*	*P*	*H*	*P*	*H*
10	1×10^{-4}	1	2×10^{-4}	1	0	1
5	9×10^{-5}	1	1×10^{-4}	1	0	1
2	1×10^{-4}	1	2×10^{-4}	1	0	1
1.5	9×10^{-5}	1	1×10^{-4}	1	0	1
0.8	5×10^{-5}	1	1×10^{-4}	1	0	1
0.4	5×10^{-5}	1	2×10^{-4}	1	0	1
0.1	5×10^{-4}	1	3×10^{-4}	1	1×10^{-3}	1
A-RBF	0	1	0	1	0	1
A-MLP	0	1	0	1	0	1

TABLE XIII
DTLZ2 (3-8-12) COMPUTATIONAL TIME

	Computational Time per 10 executions (s)			
		SPEA2	SPEA2/ CAO-RBF	SPEA2/ CAO-MLP
DTLZ2 (3)	Median	58.3	77.1	111.7
	Mean	58.9	77.4	112.6
DTLZ2 (8)	Median	102.4	101.6	133.2
	Mean	103.5	100.1	133.6
DTLZ2 (12)	Median	114.2	115.7	143.2
	Mean	115.3	115.1	143.5

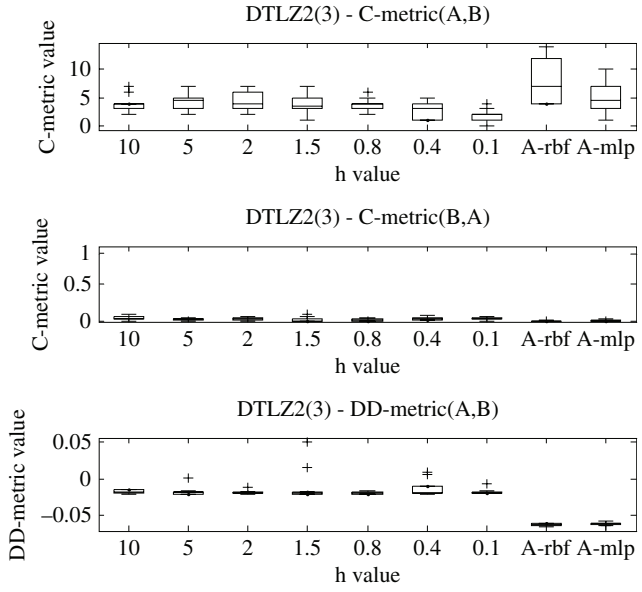

Fig. 32. Boxplots of the C-metric and DD-metric values achieved on DTLZ2 (3).

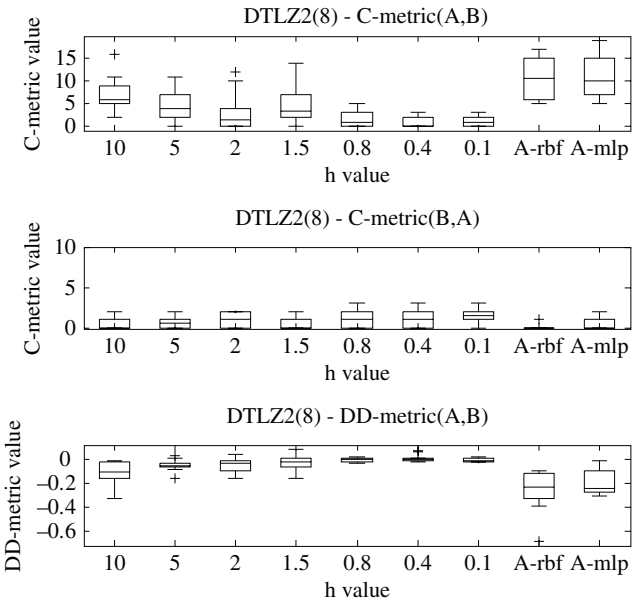

Fig. 33. Boxplots of the C-metric and DD-metric values achieved on DTLZ2 (8).

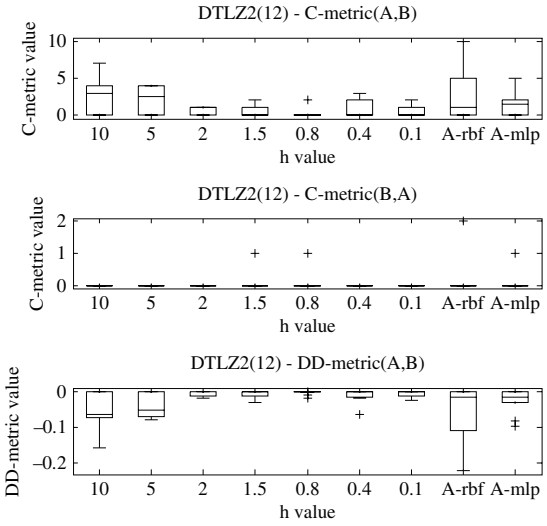

Fig. 34. Boxplots of the C-metric and DD-metric values achieved on DTLZ2 (12).

TABLE XIV
WILCOXON RANK-SUM TEST OF THE C-METRIC VALUES

	DTLZ2 (3)		DTLZ2 (**??**)		DTLZ2 (12)	
	P	*H*	*P*	*H*	*P*	*H*
10	0	1	0	1	0.04	1
5	0	1	0	1	0.03	1
2	0	1	0.02	1	0.01	1
1.5	0	1	0.01	1	0.07	0
0.8	0	1	0.04	1	0.12	0
0.4	0	1	0.02	1	0.03	1
0.1	0	1	0.06	0	0.02	1
A-RBF	0	1	0	1	0.01	1
A-MLP	0	1	0	1	0.01	1

TABLE XV

WILCOXON RANK-SUM TEST OF THE DD-METRIC VALUES

	DTLZ2 (3)		DTLZ2 (8)		DTLZ2 (12)	
	P	*H*	*P*	*H*	*P*	*H*
10	0	1	0	1	0.01	1
5	0	1	0	1	0.01	1
2	0	1	0	1	0	1
1.5	1×10^{-3}	1	0	1	0.03	1
0.8	0.6	1	0.03	1	0.04	1
0.4	5×10^{-4}	1	0.01	1	0.02	1
0.1	0	1	0.1	0	0.01	1
A-RBF	0	1	0	1	0	1
A-MLP	0	1	0	1	0	1

TABLE XVI

WFG6 (3) AND WFG9 (3) COMPUTATIONAL TIME

		Computational time per 10 executions (s)		
		SPEA2	SPEA2/ CAO-RBF	SPEA2/ CAO-MLP
WFG6 (3)	Median	139.1	138.4	238.7
	Mean	140.7	137.2	240.1
WFG9 (3)	Median	134.6	135.3	173.1
	Mean	135.8	137.7	175.9

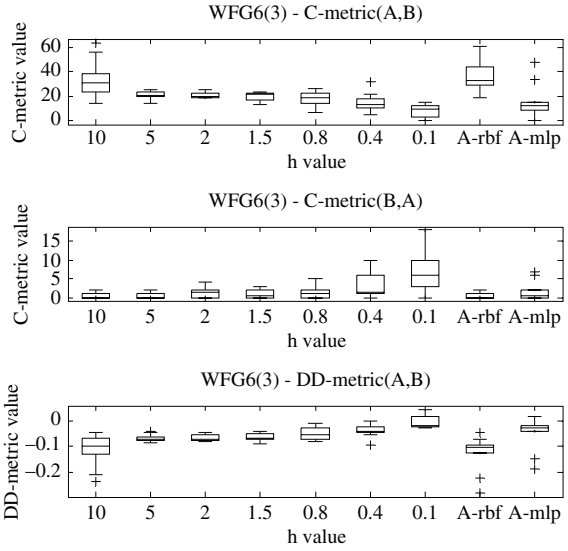

Fig. 35. Boxplots of the C-metric and DD-metric values achieved on WFG6 (3).

TABLE XVII

WILCOXON RANK-SUM TEST OF THE C-METRIC VALUES

	WFG6 (3)		WFG9 (3)	
	P	*H*	*P*	*H*
10	0	1	0.05	1
5	0	1	0	1
2	0	1	0	1
1.5	0	1	0	1
0.8	0	1	0	1
0.4	0.02	1	0.05	1
0.1	1	0	0.1	0
A-RBF	0	1	0	1
A-MLP	0	1	0.05	1

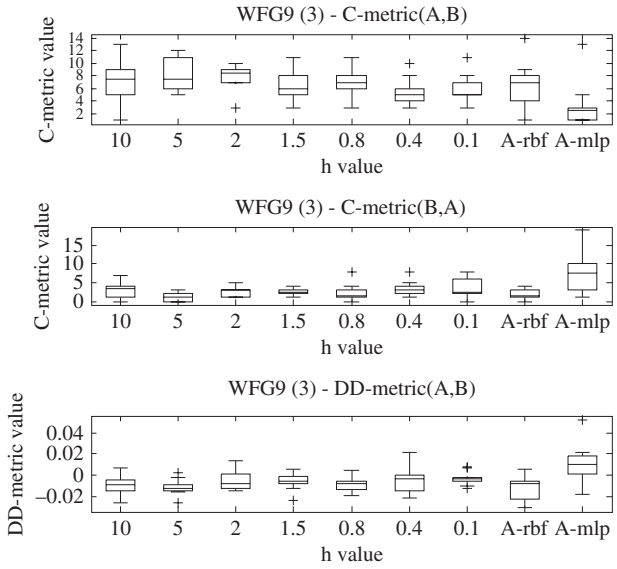

Fig. 36. Boxplots of the C-metric and DD-metric values achieved on WFG9 (3).

TABLE XVIII

WILCOXON RANK-SUM TEST OF THE DD-METRIC VALUES

	WFG6 (3)		WFG9 (3)	
	P	*H*	*P*	*H*
10	0	1	0	1
5	0	1	0	1
2	0	1	0.06	0
1.5	0	1	0.02	1
0.8	0	1	0.01	1
0.4	0	1	0.06	0
0.1	0.6	0	0.03	1
A-RBF	0	1	0	1
A-MLP	0	1	0.02	1

REFERENCES

[1] K. Deb, *Multiobjective Optimization Using Evolutionary Algorithms*. 1st ed. New York: Wiley, 2001, ch. 2, sec. 3, pp. 23–25.

[2] R. C. Purshouse, "On the evolutionary optimization of many objectives," Ph.D. thesis, Dept. Automatic Control and Syst. Eng., The Univ. Sheffield, Sheffield, U.K., 2003.

[3] C. Darwin, *The Origin of Species*. Oxford, U.K.: Oxford Univ. Press, 1996, ch. 1–4, pp. 71–172.

[4] J. H. Holland, *Adaptation in Natural and Artificial Systems*. 1st ed. Cambridge, MA: MIT Press, 1992, ch. 2, pp. 20–31.

[5] G. Gary Wang and S. Shan, "Review of metamodeling techniques in support of engineering design optimization," *J. Mech. Design*, vol. 129, no. 4, pp. 370–380, Apr. 2007.

[6] M. A. Farina, "A neural network based generalized response surface multiobjective evolutionary algorithm," in *Proc. Congr. Evol. Comput. 2002*, Piscataway, NJ: IEEE Service Center, pp. 956–961.

[7] M. El-Beltagy, P. B. Nair, and A. J. Keane, "Metamodeling techniques for evolutionary optimization of computationally expensive problems: Promises and limitations," in *Proc. Genetic Evol. Comput. Conf. 1999*, San Francisco, CA: Morgan Kaufmann, pp. 196–203.

[8] S. F. Adra, A. Hamody, I. Griffin, and P. J. Fleming, "A hybrid multiobjective evolutionary algorithm using an inverse neural network for aircraft control system design," in *Proc. IEEE Congr. Evol. Comput. 2005*, Edinburgh, U.K., pp. 1–8.

[9] N. Nariman-Zadeh, A. Atashkari, A. Jamali, A. Pilechi, and X. Yao, "Inverse modelling of multiobjective thermodynamically optimized turbojet engines using GMDH-type neural networks and evolutionary algorithms," *Eng. Optimization*, vol. 37, no. 5, pp. 437–462, Jul. 2005.

[10] M. Varcol and M. Emmerich, "Metamodel-assisted evolution strategies applied in electromagnetic," in *Proc. Eur. Conf. Evol. Deterministic Methods Design, Optimization Control Applicat. Ind. Societal Problems (EUROGEN)*, Munich, Germany, 2005, pp. 1–12.
[11] Y. Sano and H. Kita, "Optimization of noisy fitness functions by means of genetic algorithms using history of search," in *Parallel Problem Solving from Nature VI*, LNCS vol. 1917, Berlin, Germany: Springer-Verlag, 2000, pp. 360–365.
[12] B. Johanson and R. Poli, "GP-Music: An interactive genetic programming system for music generation with automated fitness raters," in *Proc. 3rd Annu. Conf. Genetic Programming*, Madison, WI, 1998, pp. 181–186.
[13] Z. Michalewicz and D. Fogel, *How to Solve It: Modern Heuristics*. 2nd ed. Berlin, Germany: Springer-Verlag, 2000, ch. 1, sec. 2, pp. 15–18.
[14] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing*, vol. 9, no. 1, pp. 3–12, Jan. 2005.
[15] K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[16] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm," in *Proc. EUROGEN 2001 Evol. Methods Design, Optimization Control with Applicat. to Ind. Problems*, Athens, Greece, pp. 95–100.
[17] A. Gaspar-Cunha and A. Vieira, "A hybrid multiobjective evolutionary algorithm using an inverse neural network," in *Proc. 1st Int. Workshop Hybrid Metaheuristics (HM 2004)*, Valencia, Spain: IOS Press, pp. 25–30.
[18] A. Gaspar-Cunha, A. Vieira, and C. Fonseca, "Multiobjective optimization: Hybridization of an evolutionary algorithm with artificial neural networks for fast convergence," in *Proc. 4th EU/ME Workshop Design Evaluation Advanced Hybrid Meta-Heuristics*, Nottingham, U.K., 2004.
[19] A. Gaspar-Cunha and A. Vieira, "A multiobjective evolutionary algorithm using neural network to approximate fitness evaluations," *Int. J. Comput., Syst. Signals*, vol. 6, no. 1, pp. 18–36, 2005.
[20] A. Gaspar-Cunha and J. A. Covas, "RPSGAe—a multiobjective genetic algorithm with elitism: Application to polymer extrusion," in *Proc. Metaheuristics Multiobjective Optimization. Lecture Notes Econ. Math. Sys.*, Berlin, Germany: Springer-Verlag, 2004, pp. 221–249.
[21] C. M. Bishop, *Neural Networks for Pattern Recognition*. 1st ed. Oxford, U.K.: Oxford University Press, 1995, ch. 4–5, pp. 116–190.
[22] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, Jun. 2000.
[23] C. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Proc. 5th Int. Conf. Genetic Algorithms*, San Mateo, CA: Morgan Kaufmann, 1993, pp. 416–423.
[24] C. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms-part I: A unified formulation," *IEEE Trans. Syst., Man, Cybern., Part A: Syst. Humans*, vol. 28, no. 1, pp. 26–37, Jan. 1998.
[25] P. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 174–188, Apr. 2003.
[26] R. C. Purshouse and P. J. Fleming, "On the evolutionary optimization of many conflicting objectives," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 770–784, Dec. 2007.
[27] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Mar. 1989.
[28] I. Nabney, "NETLAB: Algorithms for pattern recognition," in *Ser.: Adv. Pattern Recognition*, 1st ed. London, U.K.: Springer-Verlag, 2004, ch. 5–6, pp. 149–218.
[29] R. Jin, W. Chen, and T. W. Simpson, "Comparative studies of metamodelling techniques under multiple modelling criteria," *Structural and Multidisciplinary Optimization*, vol. 23, no. 1, pp. 1–13, Dec. 2001.
[30] R. Rojas, *Neural Networks–A Systematic Introduction*. 1st ed. New York: Springer-Verlag, 1996, ch. 7, sec. 2–3, pp. 153–161.
[31] D. Patterson, *Artificial Neural Networks, Theory and Applications*. 1st ed. Englewood Cliffs, NJ: Prentice-Hall, 1996, ch. 6, sec. 2–6, pp. 144–163.
[32] S. F. Adra, I. Griffin, and P. J. Fleming, "An informed convergence accelerator for evolutionary multiobjective optimizers," in *Proc. Genetic Evol. Comput. Conf.*, London, U.K., Jul. 2007, pp. 734–740.
[33] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multiobjective optimization test problems," in *Proc. Congr. Evol. Comput. 2002*, vol. 1. Honolulu, HI, pp. 825–830.
[34] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Syst.*, vol. 9, no. 2, pp. 115–148, Mar. 1995.
[35] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, Apr. 2003.
[36] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications," Ph.D. thesis, Dept Architektur, Swiss Federal Inst. Technol. (ETH), Zurich, Switzerland, 1999.
[37] M. Fleischer, "The measure of Pareto optima: Applications to multiobjective metaheuristics," in *Proc. 2nd Int. Conf. Evol. Multicriterion Optimization 2003*, Berlin, Germany, pp. 519–533.
[38] L. While, "A new analysis of the lebmeasure algorithm for calculating hypervolume," in *Proc. 3rd Int. Conf. Evol. Multicriterion Optimization*, LNCS vol. 3410. Guanajuato, Mexico, 2005, pp. 280–294.
[39] I. Griffin, P. Schroder, A. Chipperfield, and P. J. Fleming, "Multiobjective optimization approach to the ALSTOM gasifier problem," *Proc. Inst. Mech. Engrs., Part I: J. Syst. Contr. Eng.*, vol. 214, no. 6, pp. 453–468, 2000.
[40] S. Huband, L. Barone, L. While, and P. Hingston, "A scalable multiobjective test problem toolkit," in *Proc. 3rd Int. Conf. Evol. Multicriterion Optimization*, LNCS vol. 3410. Guanajuato, Mexico, 2005, pp. 280–295.
[41] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 477–506, Oct. 2006.
[42] J. D. Gibbons and S. Chakraborti, *Nonparametric Statistical Inference*. 3rd ed. New York: M. Dekker, 1992, ch. 9, sec. 2, p. 241.

Salem F. Adra (M'04) received the B.Sc. degree in computer science from the American University of Beirut, Lebanon, in 2002. He obtained the M.Sc. degree in advanced software engineering and the Ph.D. degree in control systems engineering from the University of Sheffield, Sheffield, U.K., in 2004 and 2008, respectively.

He has been a Research Associate in computational systems biology in the Department of Computer Science, University of Sheffield, since September 2007. His current research interests include multiobjective optimization and decision making, evolutionary computation, machine learning, neural networks, and computational systems biology.

Dr. Adra is a member of the Computational Systems Biology Group at the University of Sheffield. He is professional member of ACM and a Member of the Order of Engineers and Architects of Tripoli, Lebanon.

Ian A. Griffin obtained the B.Eng. degree in control engineering and the M.Sc. degree in control systems engineering from the University of Sheffield, Sheffield, U.K., in 1996 and 1997, respectively. In 2001, he was awarded the Ph.D. degree by the same university in automatic control and systems engineering for the research project "Multivariable robust control using evolutionary techniques."

He is currently Control System Design Engineer at Rolls-Royce plc, Derby, joining the company in 2007. From 2000 to 2007, he was a Research Associate in the Rolls-Royce University Technology Centre for Control and Systems Engineering, University of Sheffield, where he was responsible for research in advanced control laws for gas turbine engines. His current research interests include control system design, multiobjective optimization, and aerospace applications.

Dr. Griffin is a Member of the Institute of Engineering and Technology and a member of the IFAC Technical Committee on Optimal Control.

Tony J. Dodd obtained the B.Eng. and Ph.D. degrees from Southampton University, U.K.

He is currently an Lecturer in Aerospace Systems in the Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, U.K. Prior to this, he was a Research Associate at both The University of Sheffield and Southampton University, U.K. He has published or presented over 30 papers in refereed international journals and conferences. His research interests include machine learning, neural networks, data fusion, robotics and autonomous systems, optimization for engineering design and mission planning, and intelligent agents.

Dr. Dodd is an Associate Editor of the *International Journal of Systems Science* and has served on the International Program Committee for a number of conferences.

Peter J. Fleming received the B.Sc. and Ph.D. degrees from The Queen's University, Belfast, U.K.

He joined the University of Sheffield as Professor of Industrial Systems and Control in 1991, having previously been with Syracuse University, NY, NASA, Langley, VA and the University of Wales, Bangor, U.K. Since 1993, he has been Director with Rolls-Royce University Technology Centre in Control and System Engineering, University of Sheffield, Sheffield, U.K. He was the head of Automatic Control and Systems Engineering from 1993 to 1999, Director of Research (Engineering) from 2001 to 2003, and Pro Vice-Chancellor for External Relations from 2003 to 2008. His control and systems engineering research interests include multicriteria decision making, optimization, grid computing, and industrial applications of modeling, monitoring, and control. He has over 400 research publications, including six books, and his research interests have led to the development of close links with a variety of industries in sectors such as aerospace, power generation, food processing, pharmaceuticals, and manufacturing.

Prof. Fleming is a Fellow of the Royal Academy of Engineering, a Fellow of the Institution of Electrical Engineers, a Fellow of the Institute of Measurement and Control, an Advisor to the International Federation of Automatic Control, and the Editor-in-Chief of the *International Journal of Systems Science.*