An Evolutionary Multi-objective Meta-modeling Procedure Using Artificial Neural Networks

Kalyanmoy Deb and Pawan K. S. Nain

Kanpur Genetic Algorithms Laboratory (KanGAL)
Department of Mechanical Engineering
Indian Institute of Technology Kanpur
Kanpur, PIN 208 016, India
{deb,pksnain}@iitk.ac.in
http://www.iitk.ac.in/kangal

Abstract. This paper explores the possibility of using approximate models in multi-objective optimization. A multi-objective genetic algorithm based optimizer, namely the elitist non-dominated sorting genetic algorithm or NSGA-II, is integrated with an artificial neural network (ANN) for this purpose. The proposed technique makes use of successive fitness landscape modeling for reducing the exact function evaluation calls while retaining the basic search capability of NSGA-II. To test the suggested procedure, several test problems and a couple of practical problems are considered. The simulation results show a considerable savings in exact function evaluations in achieving both tasks of converging close to the true Pareto-optimal frontier and maintaining a good diversity of solutions. Although a neural network is used in this study, other meta-modeling techniques, such as RSM or Kriging methods, can also be used with the proposed methodology.

1 Introduction

One of the main challenges ahead with applied optimization studies is the enormous computational time needed in evaluating real-world optimization problems. In order to reduce the overall computational time, researchers in the area of search and optimization look for efficient algorithms which demand only a few function evaluations to arrive at a near-optimal solution. Although successes in this direction have been achieved by using new and unorthodox techniques (such as evolutionary algorithms, tabu search, simulated annealing etc.) involving problem-specific operators, such techniques still demand a considerable amount of simulation time, particularly in solving computationally expensive problems. In such problems, the main difficulty arises in the large computational time required in evaluating a solution. This is because such problems either involve many decision variables or a computationally involved evaluation procedure, such as the use of finite element procedure or a network flow computation.

Although the use of a parallel computer is a remedy to these problems in reducing the overall computational time, in this paper, we suggest a fundamental algorithmic change to the usual optimization procedure which can be used either serially or parallely. Most search and optimization algorithms begin their search from one or more random guess solutions. Thus, the main task of a search algorithm in the initial few iterations is to provide a direction towards the optimal region in the search space. To achieve such a task, it may not be necessary to use an exact (or a very fine-grained) model of the optimization problem early on. An approximate model of the problem may be adequate to provide a reasonably good search direction. However, as the iterations progress, finer models can be used successively to converge closer to the true optimum of the actual problem. Although this idea of using an approximate model in the beginning of a search algorithm and refining the model with iterations is not new [10, 31, 25], we suggest a generic procedure which can be used in any arbitrary problem.

In the reminder of this paper, we describe the proposed coarse-to-fine grained meta-modeling procedure. Thereafter, we suggest an artificial neural network (ANN) based procedure, specifically to model an approximate version of the actual problem and show simulation results of the proposed technique applied to a number of test problems and to a couple of practical problems. Different variations of the ANN design and training procedures are compared. Simulation results show a large computational advantage of the proposed procedure, thereby suggesting the applicability of the proposed procedure in more complex real-world search and optimization problems.

2 Past Studies

Various groups have reported their studies on the use of approximation models in evolutionary algorithms. A complete survey on the use of fitness approximation in evolutionary algorithms is reported by Jin and Sendhoff [21]. These authors have broadly classified the approximation methods in three categories, namely, response surface methodology, Kriging models and the methodology of artificial neural networks. Other issues of concern are the efficient use of exact function evaluations (termed as model management) and the quality of model itself, which should improve with iterations. Jin et al. [20], while working with approximate models, have also demonstrated the controlled evolution in an evolution strategy. Here, the word control refers to the original fitness function evaluations. So if the entire generation is evaluated using the original fitness function, it is called as controlled generation. Similarly, if only few population members are evaluated using original fitness function, it is referred as controlled individuals. A framework which guarantee the correct convergence while reducing the computational cost is established. The idea of using less function evaluations in order to reach the optima using controlled individuals with the help of clustering and neural network is explored by Jin and Sendhoff [22]. The individuals near the center of the cluster is evaluated using expensive fitness function evaluation to create neural network ensemble which is used for fitness values of remaining

individuals. The structure and parameters of the neural network ensemble are also optimized using a standard evolution strategy.

Branke and Schmidt [2] have used two estimation methods, namely, regression and interpolation, to achieve faster convergence to the optima. In there technique, at every generation, a fixed percentage of the population is evaluated with exact objective function. The fitness of the rest of the population is estimated. The individuals which are evaluated accurately are determined based on their estimated fitness and uncertainty. Savings in accurate function evaluations up to fifty percent are reported.

Farina [15] has also used radial basis neural network for objective function approximation. The algorithm has been successfully tested on test problem ZDT3 [4] which has a typical discrete Pareto-optimal front.

The attempt to reduce the number of function evaluations using fitness inheritance technique [30, 3] is also reported. Sastry et al. [30] have used inheritance combined with population sizing models and have reported a saving of 20% in function evaluations. In case of fixed population size, the study has reported a saving of 70% by employing a simple inheritance technique. Chen et al. [3] have extended the fitness inheritance concept for multi-objective optimization. The study has reported a 40% saving in terms of function evaluations for the case of fitness inheritance without fitness sharing, while in the case of fitness inheritance with fitness sharing, a saving of 25% is claimed.

Rasheed and Hirsh [26] have used informed-operators for speeding up the genetic algorithm procedure. They have modified the genetic operators (mutation and crossover) and have made them more knowledge-based (or informed) using reduced models. Instead of making a random choice of parents, they generate a number of candidates and rank them using inexpensive reduced models, and then take the best of the result. Naturally, it provides a speed-up in the genetic algorithm procedure. They have also successfully tested their method on a complex engineering design problem. In another study by Rasheed et al. [27], a comparison between two methods for using reduced models to speed up the search in genetic algorithm based engineering design optimization is presented. They have reported that the informed-operators approach is better than the genetic engineering approach. It is also found that least square approximation with any of the above two speed-up approach produces better results than neural network approximations.

Applications using surrogate models (or meta-models) which can be substituted for exact and costly evaluation tools is also discussed by Giannakoglu [16]. In this work, radial basis function neural networks are used for generating a surrogate model. The prediction capability of the surrogate model is dependent on the shape of the real response surface, availability of sufficient training data, number of hidden units and the activation function used. Studies [16, 17] have also suggested a genetic algorithm which is based on inexact pre-evaluations (GA-IPE). This method starts with exact evaluations of all solutions in the first generation. Later, when a surrogate model is constructed, the entire population is evaluated approximately using the model. Only the best individuals identified

based on the inexact evaluation are exactly evaluated. This process reduces the computational burden of the overall procedure. Hence, although the GA-IPE takes more generations, it has a low overall evaluation cost. It is successfully applied to solve a turbomachinery problem of airfoil design.

El-Beltagy et al. [11] have suggested a Kriging approach, in which local meta-models are employed instead of global meta-models. This is because of the computational cost associated with building a global meta-model. In another study, Nair et al. [24] have combined the function approximation concept with genetic algorithms for structural optimization applications. Investigators have tried to reduce the number of exact function evaluations while ensuring a convergence to the optima of the original problem. They have employed a dynamic optimization technique, wherein the fitness function changes over successive generations. They have controlled the generational delay before the approximation model is updated along with an adaptive selection operator. This technique is applied to solve a 10-bar truss design problem. They have also used a strategy by which the fitness function is changed during the run but the granularity of the optimization model is kept unchanged.

A reconstruction algorithm is proposed by Ratle [28] which uses a Kriging meta-model for fitness landscape approximation. The algorithm shows an overall reduction in the number of fitness calls. The algorithm is tested successfully for a two-dimensional problem and for a difficult 20-dimensional multi-modal optimization problem.

Recently, Emmerich et al. [12] have proposed a meta-model assisted evolution strategy. A local Kriging meta-model is built in which a fixed number of nearest neighbors are used. The method is tested on test problems and on an airfoil shape optimization problem. The results are quite encouraging for single objective optimization. In another study by Emmerich and Naujokes [13], the Kriging meta-models are used in multi-objective optimization. The local meta-models are used to decide the potential of a new population member, i.e. to decide whether it should be evaluated precisely or rejected. Three different rejection principles are also tested and compared with the original algorithm.

Although different approaches are suggested, not many studies are attempted to solve multi-objective optimization problems for finding a set of Pareto-optimal solutions. If the target is to find multiple solutions in the objective space, certain portion of the Pareto-optimal front may be easier to approximate and certain portion of the trade-off frontier may be difficult to approximate. Before we discuss the issues related to meta-modeling in multi-objective optimization, we briefly describe different procedures which can be used to reduce the computational cost of evaluating a solution.

3 Procedures for Reducing Computational Cost

It is needless to say that the lion's share of the overall computational cost of applying a GA (or for that matter any optimization procedure) comes from the

evaluation of solutions. In order to reduce the computational time required to execute one function evaluation, the following strategies can be used:

1. Use a partial evaluation of a solution,
2. Use a parallel computer, and
3. Use an approximation of the optimization problem

Certain search and optimization problems may be functionally decomposable into a number of subproblems. In such problems, the most important subproblems can only be evaluated in the initial GA generations. Although this procedure will introduce some error in evaluating a solution in early generations (since all subproblems are not evaluated), the computations can be performed quickly. In the early generations the task of an optimizer is to determine correct search directions towards the optimum, such errors may not cause a large deviation from the true search direction. However, as generations progress, less important subproblems can be included and more accurate function evaluations can be expected. In such a procedure, the computational advantage would come from the savings in the computational time in the early generations. However, it is obvious that the partial evaluation of a solution cannot be performed in problems where a functional decomposition is impossible.

Because of the availability of parallel computers, it may be plausible to take advantage of parallel computing of different tasks involved in a function evaluation. For example, to evaluate a solution involving FFT or finite element computations, the solution can be sent to multiple processors for a faster computation. Since GAs use a population of solutions in each generation, most parallel GA applications perform a distributed computing of allocating one complete solution to each available processor, thereby reducing the overall computational time to complete one generation. In such applications, although each solution can be evaluated with the help of multiple processors, usually this is not followed. Although faster solution procedures are developed using parallel migration and island models, most such studies have found a lower bound on the computational complexity achievable in terms of resorting to an optimum number of processors. Beyond the optimum number of processors, the computational advantage is overshadowed by the communication time involved among the processors.

The focus of this study is to use a successive approximation of the optimization problem, which we describe next.

4 Proposed Meta-Modeling Procedure

Starting with a coarsely approximated model of the problem, GAs use successively fine-grained models with generations. Figure 1 depicts this procedure. The figure shows a hypothetical one-dimensional objective function for minimization in a solid line. Since this problem has a number of local minimum solutions (which is one of the difficulties often exist in a real-world optimization problem), it would be a difficult problem for any optimization technique. It is concluded elsewhere [18] that to find the global optimum in such a problem

using a GA, a population of size $O(\gamma^2)$, where γ is the inverse of the signal-to-noise of the function, is needed. The signal being the difference between the global and the next-best local optimal values and the noise being equal to the variance of the function values. Thus, the objective function shown in the figure demands a large population size, if the GA has to start from an initial random population. If each function evaluation is expensive, the application of a GA in such problems becomes difficult. Although a remedy in such problems is to not use a random initial population, but to use problem information to create the initial population in the vicinity of the global optimum, in many problems such information is simply not available. However, there exist generic solutions to the problem which we discuss in the following paragraph.

Figure 1 also shows a coarsely approximated function in the entire range of the function with a dashed line. There could be a variety of ways such an approximated function can be obtained:

1. Linear or quadratic (or a response surface methodology) approximation of the true function
2. Approximation through a set of basis functions
3. Approximation through a chosen set of solutions

Classical methods often linearize non-linear optimization problems at suitable solutions and use a linear programming technique successively, such as the Frank-Wolfe method or successive linearization methods [29]. Besides, linearization techniques, non-linear problems can be approximated by quadratic or higher-order polynomial functions. Another way to approximate a function is to use a set of basis functions and find a weighted sum of such basis functions (finite or infinite numbers of them) as an approximation. Fourier approximation and Walsh function approximations are two such examples. Once such an approximation is known, the individual properties of the optimum of the basis functions may be analyzed to make a guess of the optimum of the approximating function. Although such techniques are used to decouple linkages among decision variables to convert the problem into a class of separable programming [29], such techniques are usually not used in optimization studies. However, such approximations can also be used nicely with the following approximation procedure.

The optimization problem can be evaluated exactly at a few finite number of pre-specified solutions in the entire range of the decision variables. Thereafter, an approximating function can be fitted through these function values using regression or some other sophisticated techniques such as an artificial neural network methodology. It is clear that if a large number of solutions are chosen, the approximating function will take a shape closer to the original function. On the other hand, if only a few solutions are chosen, the approximated function will ignore the local details and represent a coarse trend in variation of the function values. If this approximating function is optimized, it is likely that a GA will proceed in the right direction. However, as a GA tends to converge to the optimum of the approximating function, the approximating function needs to be modified to make a better approximated function from before. Since the population diversity will be reduced while approximating the first approximated

function, the second approximating function need not be defined over the whole range of the decision variables, as shown in Figure 1. This process may continue

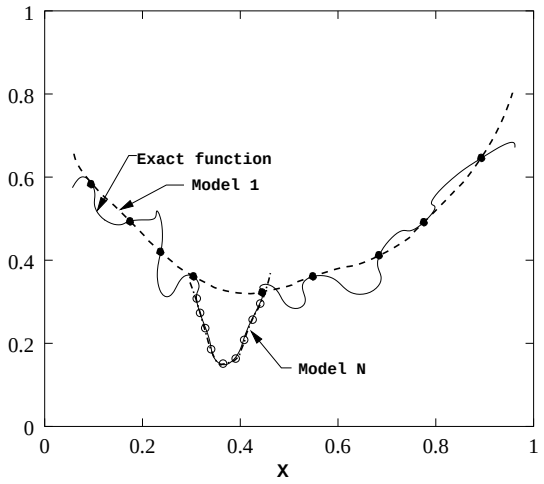


Fig. 1. Proposed successive meta-modeling technique.

till no further approximation results in an improvement in the function value. Although this successive approximation technique seems a reasonable plan, care must be taken to ensure that adequate diversity is retained while switching from one approximating function to a better one.

4.1 Specific ANN-Based Procedure

Figure 2 outlines a schematic of a plausible plan for the proposed procedure. The combined procedure begins with a set of randomly created N solutions, where N is the population size. Since an adequate size of solutions are required to arrive at an approximated problem, we execute a GA with exact function evaluations for n generations, thereby collecting a total of $N' = nN$ solutions for approximation. At the end of n generations, the approximation technique is invoked with N' solutions and the first approximated problem is created. The GA is then performed for the next $(Q - n)$ generations with this approximated problem. Thereafter, the GA is performed with the exact function for the next n generations and a new approximated problem is created. This procedure is continued till the termination criterion is met. Thus, this procedure requires a fraction n/Q of total evaluations in evaluating the problem exactly. A better approach will be to decrease N' solutions towards the later generations of GA run as during this phase approximate model will not change significantly.

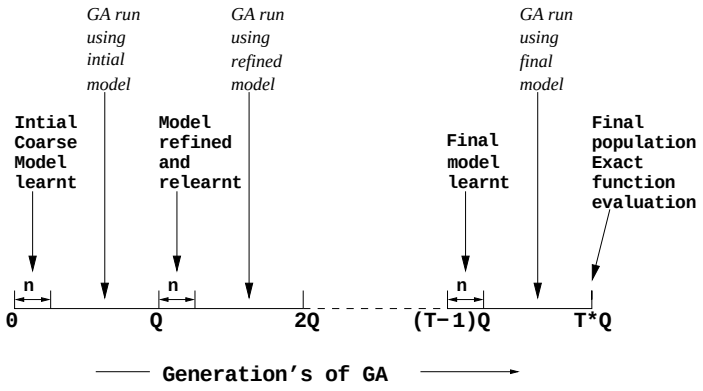


Fig. 2. A sketch of the proposed NSGA-II-ANN technique.

If a problem cannot be evaluated exactly, instead some approximations (such as involving FFT or finite element techniques) are needed to evaluate, the parameter n is set to zero and GAs are run for Q generations with the most coarse model (in the case of a finite element method only a few elements can be chosen to start with). Thereafter, the model is modified and GAs are run for another Q generations with the modified (and hopefully a better) model. This procedure will continue till a termination criterion is met. It is interesting to note that a set of basis functions with varying importance to local variations can be used as approximating functions here. In such cases, a GA may be started with an approximating function involving only a few basis functions, thereby allowing a quicker computation of the optimal solution. With generations, more and more basis functions can be added to make the model more similar to the actual optimization problem. In this study we concentrate on solving problems for which an exact evaluation method exists but is computationally expensive. However, similar methodology can also be used to solve problem for which no exact evaluation method exists.

We propose combining a GA with the artificial neural networks (ANN) as the basic approximating technique for fitness computation. The primary reason for using ANN as the basic approximating tool is its proven capabilities as function approximation tool from a given data set. The multilayer perceptron trained with the back-propagation algorithm may be viewed as a practical vehicle for performing a non-linear input output mapping of general nature [19]. It is important to note that first function evaluation should always be performed using the exact problem, as we do not have any database to train the neural network. Secondly, whenever we are performing exact function evaluations, we have to build or update the training database for training or retraining of the ANN. The frequency of training is exclusively controlled by the user. The parameters Q and n would be so chosen that even if a previous approximation function does

not adequately approximate the true problem, subsequent approximations have a chance to correct the population and push it in the right direction.

Another advantage of the proposed technique is its adaptability. Initially the GA population will be randomly spread in the entire search space for the problem undertaken. Since the same information is available in the ANN training database the approximated model generated using this database, will also be very general in nature and hence may miss some finer details about the search space. However as the generations proceed, the GA population will start drifting and focusing on the important regions which it identifies based on the current model. So when the proposed technique updates its model using exact function evaluation for the current generation, it will have more information about the local regions of interest as more population members will now be available in those regions than earlier. The ANN will now retrain and update its weights making it learn and adapt to the new smaller regions in the search space. Hence it will give finer refined approximated model to direct the search of GA in the subsequent generations. Thus the proposed technique will adaptively refine the approximated model from coarse to fine approximated model of the optimization problem.

The reasons for choosing ANN for function approximation are (i) its ability to learn a multi-variable functional relationship on arbitrary functions, and (ii) its fast evaluation of a solution with the trained ANN. It is worth mentioning here that the computational time taken to train an ANN must be added to the overall computational time taken to solve the problem. However, using an incremental learning ANN, the ANN training phase and n generations of a GA run with exact evaluations can be continued parallely by using a dual processor machine. However, it is assumed in this study that the computational time for each function evaluation is so large that the time taken for ANN training is comparatively small. It is also worth mentioning that the proposed technique is equally applicable to single and multi-objective optimization and to constrained optimization, as an additional objective or constraint in the optimization problem is equivalent to addition of one more neuron in the output layer of ANN.

5 Proof-of-Principle Results

In all case studies performed here, we have used the NSGA-II procedure [5] as the basic multi-objective optimizer. The standard error back-propagation algorithm with sigmoidal activation function [19] is used for generating approximate model of the problem successively. For the ANN, input neurons represent problem variables and output neurons represent different objective functions and constraint functions. The combined procedure will be referred as NSGA-II-ANN simulation. The ANN is trained using two different models, namely, batch training and incremental training. If a simulation is performed with batch training in which after every Q generation the approximation model is refined and the training database is collected over n generations, It is called as the B-Q-n model. If the training method is incremental, then it is called as the I-Q-n model.

Here, we choose a set of available two and three-objective test problems, namely TNK, ZDT4, DTLZ2 [4, 6]. These problems are selected as they test different aspects of the optimizer, which will be highlighted in the subsequent sections. The learning rate of 0.3, a momentum factor of 0.1, permissible rms error of 0.001, SBX crossover probability of 0.9 and distribution index of 10 and the polynomial mutation (probability equal to the inverse of number of variables) with a distribution index of 50 are used for all test problems presented here.

5.1 Test Problem TNK

It is a two variable constrained test problem. The problem variables are real valued. The Pareto-optimal front is disconnected with three parts and falls on the constraint boundray. Since it is a difficult problem, 200 population members and a maximum of 2,500 generations are chosen for all simulations including the meta-modeled NSGA-II-ANN (Figures 3 and 4). The number of exact function

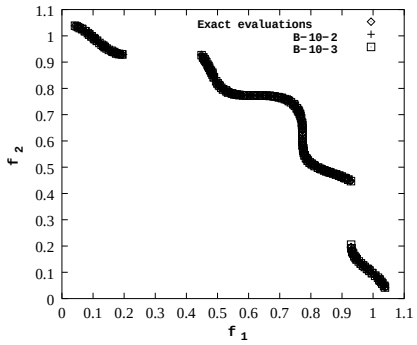


Fig. 3. Batch model simulation results for the TNK test problem.

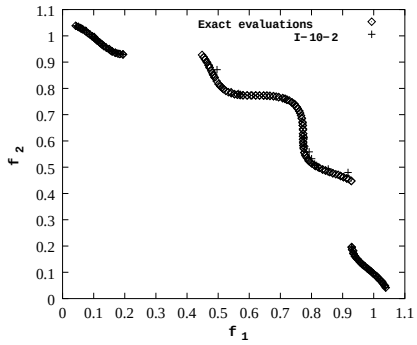


Fig. 4. Incremental model simulation results for the TNK test problem.

evaluations taken by simulations of various models in order to reach the Pareto-optimal front are given in Table 1. Figure 3 shows that B-10-2 and B-10-3

Table 1. Spread metric results of NSGA-II-ANN simulations on the TNK test problem.

Model Name	Hidden Neurons	Part-I	Part-II	Part-III	Exact Function Evaluations
NSGA-II	NA	0.644	0.872	0.704	5,00,400
B-10-2	9	0.597	0.810	0.613	2,50,183
B-10-3	11	0.588	0.833	0.724	2,50,256
I-10-2	9	0.540	1.047	0.507	4,99,933

NSGA-II-ANN simulations reach the Pareto-optimal front with a saving of about 50% in exact function evaluations. Figure 4 shows that I-10-2 NSGA-II-ANN simulations reach to Pareto-optimal front without any saving in exact function evaluations.

The spread metric calculation method is as suggested in Deb et al. [4]. In this metric, the Euclidean distance between two extreme ends of Pareto-optimal front as well as the uniformity of distribution for intermediate solutions is considered. A smaller value of the spread metric means a better spread. Batch model simulations show better spread results as well as savings in exact function evaluation. If we concentrate on the spread metric value, B-10-2 simulation is the best and is closely followed by the B-10-3 simulation. However, incremental mode simulations have difficulty in capturing the middle portion of the Pareto-optimal front which is also evident from the worse spread metric value obtained for the I-10-2 simulation.

5.2 Test Problem ZDT4

It is a ten-variable, real valued problem having a convex global Pareto-optimal front. This is a difficult test problem as it exhibits 100 distinct local Pareto-optimal fronts, out of which only one is global. Figures 5 and 6 show that the NSGA-II-ANN in batch and incremental mode, respectively, are able to reach the global Pareto-optimal front in all simulations with a population size of 100 run for 300 generations. However, due to presence of local sub-optimal Pareto-optimal fronts, the saving in exact function evaluations is somewhat smaller. The

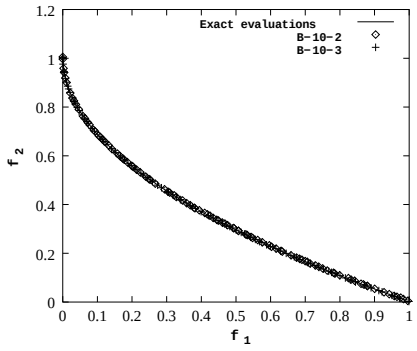


Fig. 5. Batch model simulation results for the ZDT4 test problem.

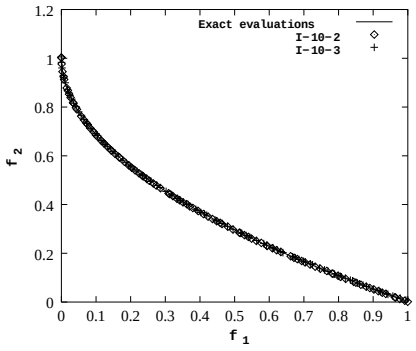


Fig. 6. Incremental model simulation results for the ZDT4 test problem.

number of exact function evaluations taken by simulations of various models in order to reach the Pareto-optimal front are given in Table 2. The saving in both batch models namely, B-10-2 and B-10-3 models are approximately 25%.

The best spread metric value is achieved by NSGA-II-ANN simulation in batch mode of training, namely, for B-10-3 model. The incremental models,

Table 2. Spread metric results of NSGA-II-ANN simulations on the ZDT4 test problem.

Model Name	Hidden Neurons	Spread metric value	Exact Function Evaluations
NSGA-II	NA	0.386	30,200
B-10-2	17	0.422	22,675
B-10-3	15	0.332	22,730
I-10-2	13	0.344	22,675
I-10-3	17	0.388	21,781

namely, I-10-2 and I-10-3 also show better spread metric values. Only one simulation, namely B-10-2 gives a poor spread when compared to the NSGA-II simulation performed with exact function evaluations.

5.3 Test Problem DTLZ2

This 12-variable test problem has three objectives with a concave Pareto-optimal front. It is a problem where most of the multi-objective optimizers find difficulty in reaching the Pareto surface and maintaining a uniform distribution of the solutions on the Pareto-optimal surface. Figures 7 and 8 show the Pareto-optimal fronts for batch and incremental NSGA-II-ANN with 100 population members run for 300 generations. The number of exact function evaluations taken by both simulations are tabulated in Table 3. The batch model B-10-3 shows a saving

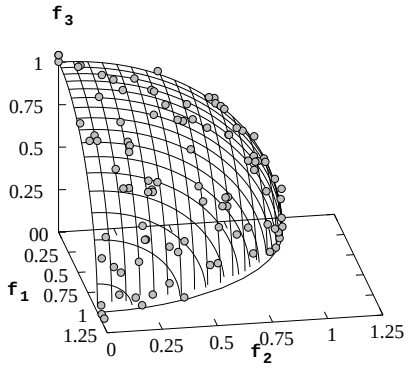


Fig. 7. Batch model simulation results for the DTLZ2 test problem.

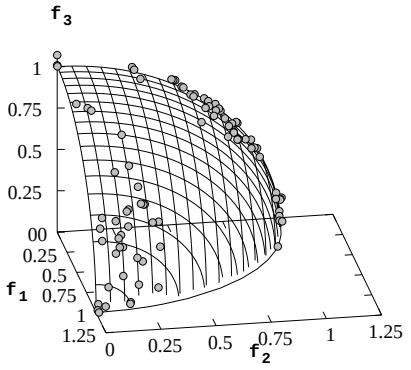


Fig. 8. Incremental model simulation results for the DTLZ2 test problem.

of 62% in exact function evaluations. The incremental model I-10-3 provides a saving of 44% in exact function evaluations. The distribution of the solutions on

Table 3. Sparsity measure results of NSGA-II-ANN simulations on the DTLZ2 test problem

Model Name	Hidden Neurons	Sparsity measure value	Exact Function Evaluations
NSGA-II	NA	0.951	30,200
B-10-3	17	0.964	11,496
I-10-3	7	0.908	17,120

the Pareto-optimal surface is evaluated by using a sparsity measure [7]. It is a method to quantify the distribution of solutions similar to an entropy measure or the grid diversity measure. A higher value of the sparsity measure means a better diversity among the solutions. Table 3 shows the sparsity measure for NSGA-II-ANN simulation. The B-10-3 batch model shows the best diversity among Pareto-optimal solutions followed by NSGA-II run with only exact function evaluations. I-10-3 incremental model depicts the poorest distribution of solutions in this problem.

6 Case Study 1: A Curve Fitting Problem

The proposed technique is now applied to a B-spline curve fitting problem. A saw-tooth function with two teeth is taken as the basic curve to be fitted using B-splines (Figure 9). The tooth root height of 0.1, tooth peak height of 0.6, and tooth span of 0.5 are chosen here.

A B-spline with the parameter $k = 3$ produces a polynomial curve of degree two, with C^1 continuity for all curve segments and guarantees to pass through the starting and end control points and make tangents at the corresponding line segments.

The following two conflicting objectives are considered:

1. Minimize the error between the saw-tooth curve and the B-spline fitted curve, and
2. Minimize the maximum curvature of the B-spline fitted curve.

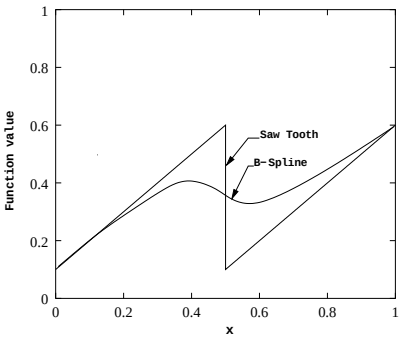


Fig. 9. The B-spline curve fitting problem.

In the current problem, the number of control points are taken to be 41, thus dividing total x range in 40 equal divisions. However, in order to create meaningful solutions, the first and the last control points for the B-spline is fixed at

tooth root height and tooth peak height, respectively, thereby leaving only 39 control points to be treated as decision variables. For any given B-spline curve $\mathcal{S}$, the exact evaluation of the first objective can be achieved in the following manner:

$$F_1(\mathcal{S}) = \int_{x=0}^{x=1} |f_{\text{saw-tooth}} - \mathcal{S}| dx. \tag{1}$$

Since such a computation is difficult to achieve exactly (mainly because of the non-differential modulus function used in the operand), we compute the above integral numerically by using the Trapezoidal rule. We have used 400 divisions in the entire range of x for the computation of two objectives. The second objective can be written as follows:

$$F_2(\mathcal{S}) = \max_{x=0}^{x=1} \frac{\frac{d^2\mathcal{S}}{dx^2}}{\left[1 + \left(\frac{d\mathcal{S}}{dx}\right)^2\right]^{3/2}}. \tag{2}$$

Since the B-spline curve $\mathcal{S}$ is defined piece-wise, the term for the curvature can be derived exactly for each segment. The term can then be optimized exactly using the first and second-order optimization criteria and the following location of the optimum is found in each B-spline segment:

$$u^* = 0, \text{ if } u_c \leq 0; \quad u^* = 1, \text{ if } u_c \geq 1; \quad u^* = u_c, \text{ otherwise}, \tag{3}$$

where the parameter u_c is calculated as follows:

$$u_c = \frac{x_{uu}(x_0 - x_1) + y_{uu}(y_0 - y_1)}{x_{uu}^2 + y_{uu}^2}, \quad x_{uu} = x_0 - 2x_1 + x_2, \quad y_{uu} = y_0 - 2y_1 + y_2.$$

Here, (x_0, y_0), (x_1, y_1) and (x_2, y_2) are three control points of each segment. Once the optimal u^* is calculated, the corresponding curvature can be calculated as follows:

$$R = \frac{x_u y_{uu} - x_{uu} y_u}{\left(x_u^2 + y_u^2\right)^{3/2}}, \tag{4}$$

where the first derivatives x_u and y_u are calculated as follows:

$$x_u = (u^* - 1)x_0 + (1 - 2u^*)x_1 + u^* x_2,$$
$$y_u = (u^* - 1)y_0 + (1 - 2u^*)y_1 + u^* y_2.$$

Such computations can be performed for all segments and the maximum curvature of the entire B-spline curve can be determined. For a large number of B-spline segments, many such computations are required, thereby involving a large computation time to evaluate the second objective. If such computations are extended to 3-D curve or surface fitting, the computations become even more expensive.

Here, we use a population of size 200, a crossover probability of 0.9 with a distribution index of 10, and the polynomial mutation with a distribution index of 50 [4]. The database size reduces linearly to 25% of its initial size in a span

of 50% middle generations of EA. For initial and last 25% generations of EA, database size is kept constant. The permissible normalized rms error for ANN model is taken 0.005.

An extensive parametric study is performed by varying four parameters, (i) ANN learning rate, (ii) number of hidden neurons, (iii) model updating frequency parameter Q and (iv) the database size parameter n. In all cases, the number of overall exact function evaluations is kept the same to 750×200 or 1,50,000. All simulations are compared with an NSGA-II simulation performed with exact evaluations (200 population size and 750 generations). The trade-off frontiers corresponding to the best performing batch and incremental learning NSGA-II-ANNs are shown against the exact NSGA-II performance in Figure 10. In all simulations with batch and incremental learning models with different pa-

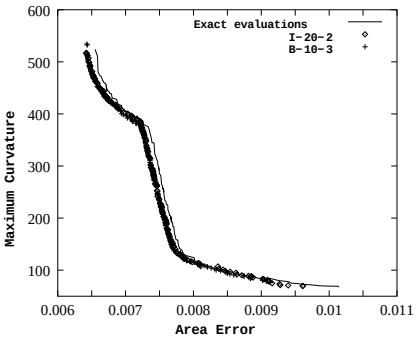


Fig. 10. Best of both incremental and batch models simulation results for curve fitting problem.

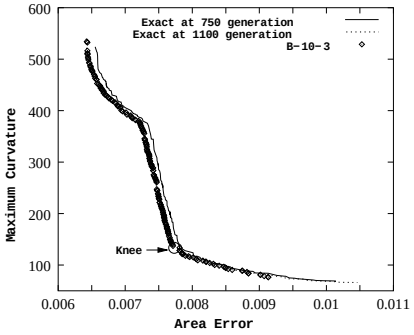


Fig. 11. Comparison of the overall best simulation result with exact solution at 1,100 generation for curve fitting problem.

rameter settings, the obtained non-dominated front is found to be better than that obtained using exactly-evaluated NSGA-II. The best result for incremental training is found with the I-20-2 model, while in the case of batch training slightly better results are found with the B-10-3 model. This demonstrates that although approximate models are used, the combined NSGA-II-ANN procedure proposed here is able to find a better non-dominated front than the exact model. In order to investigate how many generations it would take by the NSGA-II with exact evaluations to obtain a front similar to that obtained using the proposed approaches, we have continued the NSGA-II run with exact evaluations for more than 750 generations. Figure 11 shows that the B-10-3 model reaches a similar front in about 1,100 generations, thereby making a saving of around 32% by the meta-modeled NSGA-II.

In order to visualize the B-spline curves obtained using the NSGA-II-ANN simulations, the two extreme solutions and a *knee* solution from the final non-

dominated front of B-10-3 model are drawn in Figure 12. The original saw-tooth

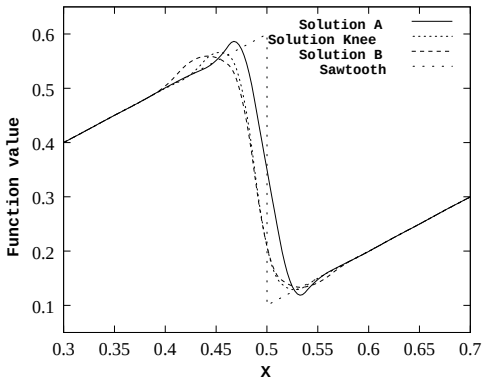


Fig. 12. Two extreme solutions and a knee solution for the curve fitting problem.

function is also shown in dots. For clearly showing the region near $x = 0.5$, we plot them for $x \in [0.3, 0.7]$. The figure shows that one solution (marked as A) is a better fit with the saw-tooth function, whereas the other solution (marked as B) not a good fit in the vicinity of $x = 0.5$, but produces a smaller curvature. The knee solution is an intermediate solution from which if we move in either direction for a small improvement in one objective, a large sacrifice in other objective is to be made [1].

6.1 Effect of Permissible RMS Error

The issue of choosing a proper permissible normalized rms error for ANN also plays an important role in the proposed NSGA-II-ANN procedure. A large value of permissible normalized rms error may not adequately approximate the true problem as it will propagate a large error between the exact and the obtained model. Similarly, a very low value of permissible normalized rms error will lead to over-approximation of the true problem, thereby causing ANN to lose its generalization capability. This discussion indicates that there should be a critical value of permissible normalized rms error at which the proposed NSGA-II-ANN procedure should produce a better performance.

Various values of permissible normalized rms error are also tried with the B-10-3 model in the range $[0.001, 0.010]$. It is found that a permissible normalized rms error value less then 0.003 leads to over-approximation of the true problem. Hence NSGA-II-ANN procedure fails to converge close to the true Pareto-optimal front. Table 4 shows that permissible normalized rms error values above 0.003 and up to 0.004 are able to find a non-dominated front which is better converged than that obtained using the exact function evaluations (for 750 generations), indicating a saving of exact evaluations. At the permissible normalized

rms error value of 0.005, the performance of proposed NSGA-II-ANN procedure is the best. However, a further increase in the permissible normalized rms error value above 0.006 shows poor convergence, with no savings in the exact evaluations. The obtained non-dominated front is inferior to that obtained with exact evaluations for 750 generations. The convergence metric computes the average

Table 4. Convergence metric for different modeling error in curve fitting problem.

Sl. No.	Model Name	Permissible Normalized RMS Error	Convergence Metric	Normalized Convergence Metric
1	*B*-10-3	0.003	0.012525	0.3779
2	*B*-10-3	0.004	0.012525	0.3779
3	*B*-10-3	0.005	0.000779	0.0235
4	*B*-10-3	0.006	0.033142	1.0000
5	*B*-10-3	0.007	0.033142	1.0000
6	*B*-10-3	0.008	0.033142	1.0000
7	*B*-10-3	0.009	0.033142	1.0000
8	*B*-10-3	0.010	0.033142	1.0000
9	*I*-20-2	0.005	0.003458	0.1043
10	*Exact*-750	N.A.	0.019351	0.5839
11	*Exact*-1100	N.A.	0.003795	0.1145

distance of each obtained solution from a reference set of points. As the current problem is a practical problem, the true Pareto-optimal front is not known. As suggested in [8], we choose a reference set P^* containing 274 data points obtained from a combined pool of 11 simulations (shown in Table 4). For calculating the convergence metric value, first the non-dominated set F of the final generation of each simulation is identified. Then for each point in F, smallest normalized Euclidean distance to P^* is calculated. Next the convergence metric value is calculated by averaging the normalized distance of all points in the F. Lastly, in order to keep the convergence metric within $[0, 1]$, we divide the convergence metric value by the maximum value found among all simulations.

6.2 Comparing Both Convergence and Spread

Next, we make a detailed study comparing different batch and incremental learning NSGA-II-ANNs with exactly evaluated NSGA-II based on two performance measures: (i) convergence and (ii) diversity preservation. Since the true Pareto-optimal front is not known in this problem, we combined the solutions found in all simulation runs and treat the resulting non-dominated front as the target frontier.

Figure 13 shows the normalized convergence metric values of the different simulations. However it should be noted that maximum value of convergence

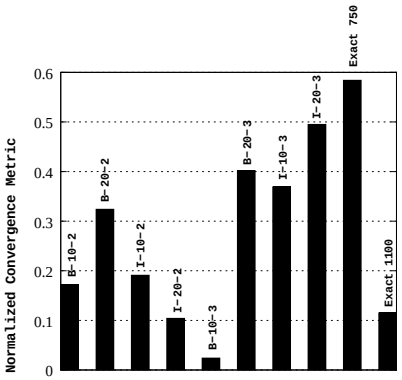


Fig. 13. Normalized convergence metric for different simulations for curve fitting problem.

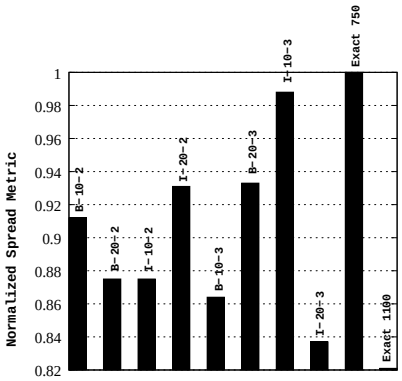


Fig. 14. Normalized spread metric for different simulations for curve fitting problem.

metric obtained for the curve fitting problem is 0.033142 (as shown in Table 4). Hence the maximum ordinate in the Figure 13 is not unity. But still this figure clearly communicates the comprehensive picture of the convergence obtained in various simulations. It is observed from the figure that for the batch model running with two generational database size ($2N$), B-10-2 model performed better than B-20-2 model. In the incremental model with two generational database, I-20-2 model performed better than I-10-2 model. In case of three generation ($3N$) database size with batch model (results not shown here), B-10-3 model has also performed better than B-20-3 model. In case of incremental model with three generational database size, I-10-3 model has also performed better than I-20-3 model. The best incremental model is found to be I-20-2 and the best batch model is B-10-3. Both of these models performed better than NSGA-II simulation run with exact function evaluations for $1,100$ generations. The overall best performance is obtained in the B-10-3 batch model. So all simulations working with approximate function evaluations performed better than the NSGA-II simulation working with exact function evaluations for 750 generations and hence show an encouraging trend for the use of the technique presented here.

Next, the distributing ability of these simulations is examined. The spread metric is calculated by identifying two extremes of the reference set P^*. The simulation for which the spread metric is to be calculated, is taken and the best non-dominated front F is identified. Next the two extremes of this best non-dominated front F are identified. In the spread metric calculation two aspects are covered:

1. The closeness of two extremes of the reference set P^* to the corresponding extremes of best non-dominated front of simulation F, and
2. The uniformity of distribution of intermediate non-dominated points of the simulation F.

The smaller is the spread metric value, better is the distribution of the simulation under investigation. In order to visualize a comprehensive picture of the spread metric value for all the ten simulations, once again the spread metric values are normalized in the range of $[0, 1]$ and is plotted in Figure 14. The worst distribution is obtained for the benchmark simulation labeled Exact-750. All NSGA-II-ANN simulations performed better than this particular simulation. But the best simulation with respect to spread metric is Exact-1100 simulation with normalized spread metric value of 0.821. The worst batch model NSGA-II-ANN simulation is B-20-3 with 194 non-dominated points and normalized convergence metric value of 0.933. The best batch model NSGA-II-ANN simulation is B-10-3 with 187 non-dominated points and normalized convergence metric value of 0.864. Even though the number of non-dominated points in case of B-10-3 simulation are less than that with B-20-3 simulation, the spread is better for B-10-3 simulation. The worst incremental simulation is I-10-3 with 193 non-dominated points and normalized convergence metric value of 0.988. The best incremental simulation is I-20-3 with 193 non-dominated points and normalized convergence metric value of 0.837. So it can be concluded that in view of the spread measure, better performance of meta-medeled NSGA-II-ANNs is observed.

6.3 Effect of Initial Population and Attainment Curve

Next, in order to study the effect of initial population, we perform 21 different simulations of the best-performing batch model (B-10-3) with different initial populations. Independent non-dominated fronts are combined and 0%, 50% and 100% attainment curves [14] are plotted in Figure 15. The 0% and 100% attain-

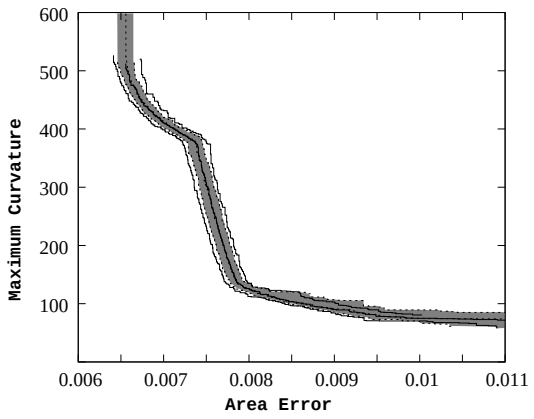


Fig. 15. Attainment curve plot for exact and approximate runs for the curve fitting problem.

ment curves divide the objective space into three main regions. The first region located to the left of the 0% attainment curve (for minimizing both objectives) is composed of the solution vectors which were never attained in any of the run. This region gives the idea about the best performance of the algorithm. The second region located to the right of the 100% attainment curve represents the solution vectors which were attained in all the simulation runs. This region gives the idea of the worst-case performance of the algorithm. The third region located within the 0% and 100% attainment curves is composed of solutions which are attained in some of the simulation runs but not in all simulations. This region can further be subdivided into sub-areas according to percentage of simulation runs in which the corresponding objective vectors are attained. A 50% attainment curve indicates the curve which is attained in half of the simulations.

It can be observed from the figure that the best performance (0% attainment curve) of NSGA-II simulations, working with exact evaluations for 750 generations, is better than the NSGA-II-ANN B-10-3 model. The 50% attainment curve for the NSGA-II working with exact function evaluation and is almost similar to that for the NSGA-II-ANN simulations working with B-10-3 model, suggesting that the median performance of the two approaches are more or less identical. The 100% attainment curves show that worst-case performance of NSGA-II-ANN approach is better than that of the exactly-evaluated NSGA-II. It is also observed from the figure that variation in the performance of NSGA-II with exact evaluations due to a change in initial population is slightly more than that in the batch learning NSGA-II-ANN.

7 Case Study 2: A Cantilever Plate Design for Optimal Shape

The design task is to find optimal shapes for two conflicting objectives: (i) minimize weight of the plate and (ii) minimize maximum deflection anywhere on the plate due to application of the load. A functional constraint restricts the maximum developed stress to be less than the specified yield strength of the plate material. The plate, loading, and initial meshes (of size 12×20) used in the study are shown in Figure 16. A load of 100 kN is applied at the free end of the cantilever. The plate material is taken to be steel. A binary variable represents presence or absence of the material at a grid element. Thus, a binary string of size 240 represents a shape. The continuity of materials from support to load is enforced and a finite element procedure is employed for exact function evaluations, thereby requiring a large computational overhead.

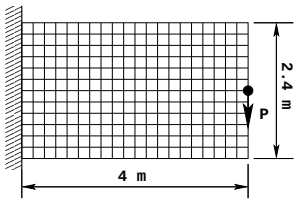


Fig. 16. A cantilever plate.

Since in the previous examples, the batch learning NSGA-II-ANN performed the best, we apply this method only to this shape optimization problem. Several batch models are applied and the best performing results are shown in Figure 17.

For a comparison, the trade-off frontier corresponding to the exactly-evaluated NSGA-II is also shown. In all simulations, 5,400 exact function evaluations are

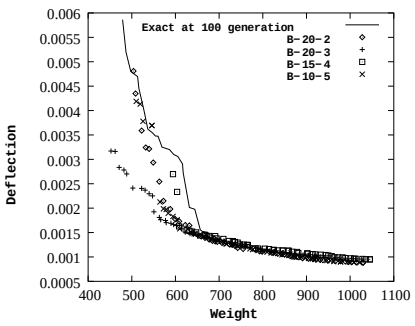


Fig. 17. Best of different batch models for the shape optimization problem.

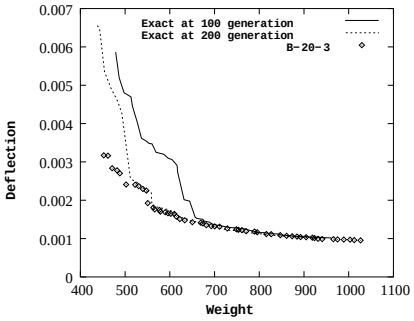


Fig. 18. Comparison of best batch simulation with exact NSGA-II runs for the shape optimization problem.

allowed. It can be seen that the B-20-3 NSGA-II-ANN frontier fully dominates that obtained using exactly-evaluated NSGA-II simulation (Figure 18). Interestingly, even when the exactly-evaluated NSGA-II is continued to run for 200 generations (totaling 10,800 exact function evaluations), the obtained frontier is not as good as that obtained with the meta-modeled NSGA-II using 50% evaluations. Hence, the B-20-3 model saves at least 50% function evaluations.

7.1 Convergence and Spread of Solutions

To quantify the level of convergence achieved in each simulation, we compute a convergence and spread measures used earlier. Like before, we choose a reference set P^* containing 94 data points obtained from a combined pool of 15 simulations. Figure 19 shows the normalized convergence metric value calculated for various simulations by the NSGA-II-ANN procedure and the NSGA-II working with exact function evaluations (labeled as exact). This figure shows that convergence metric value for all the NSGA-II-ANN simulations is either near or better than NSGA-II run with exact function evaluations for 200 generations. The convergence metric value of the best simulation, namely B-20-3, is somewhere in between 250 generation and 300 generation. Hence, a saving larger than 50% is achieved. Next, we compute the spread measure using the same reference set P^*. Figure. 20 shows that the B-20-3 model finds the best spread.

7.2 Effect of Initial Population and Attainment Curves

We now study the effect of initial population on the best-performing B-20-3 meta-model. We use 5 hidden neurons and a learning rate of 0.3, which performed

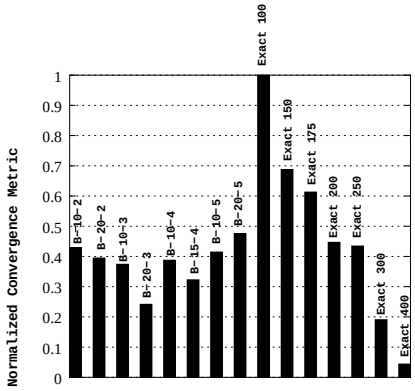


Fig. 19. Normalized convergence metric results for different simulations for shape optimization problem.

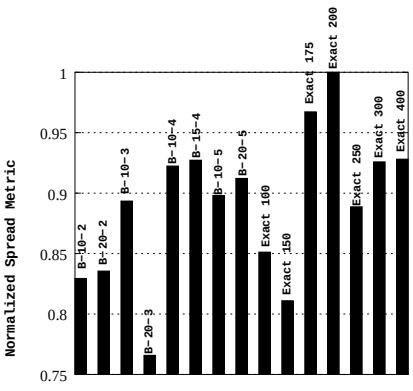


Fig. 20. Normalized spread metric results for different simulations for shape optimization problem.

the best in the above discussion. Once again, 21 different simulations are run for this meta-model and for the exactly-evaluated NSGA-II and different attainment curves [14] are plotted in Figure 21. In both cases, the runs are terminated when a total of 5,400 exact evaluations are done. The NSGA-II attainment curves are shown by solid lines and NSGA-II-ANN simulation in dashed lines. It can be observed that the B-20-3 NSGA-II-ANN model produces a better performance in the small-weight region (high-deflection region) compared to that obtained using the exactly-evaluated NSGA-II simulations. For larger-weight or smaller-deflection solutions, both NSGA-II and meta-modeled NSGA-II-ANN have achieved a similar level of performance. The median performance of the two algorithms are more or less same in the larger-weight or smaller-deflection region, while in smaller-weight or larger-deflection region, the performance of meta-modeled NSGA-II-ANN working with B-20-3 model is found to be better. Investigating the 100% attainment curve, it can be concluded that the worst-case performance of meta-modeled NSGA-II-ANN is better than that of the exactly-evaluated NSGA-II. To show the variation of performance due to a change in the initial population clearly, we shade the region between 0% and 100% attainment curves for meta-modeled NSGA-II-ANN simulations. It is observed from this figure that although variations in both approaches are substantial in the small-weight region of the trade-off curve, the performance of meta-modeled NSGA-II is somewhat better. Overall, the study also indicates that optimal solutions corresponding to smaller-weight (or larger-deflection) are difficult to achieve and more sensitive to GA parameter settings than the smaller-deflection (or larger-weight) solutions. Importantly, the reduced exact evaluations with the proposed scheme is found to be better or equivalent to the exactly-evaluated NSGA-II.

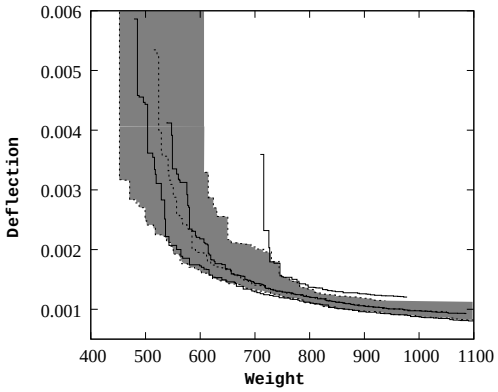


Fig. 21. Attainment curve plot for exact and approximate runs for the shape optimization problem.

8 Conclusions

Most real-world optimization problems require a computationally expensive procedure of evaluating objective functions and constraints. This computational load is no less in multi-objective optimization. Hence there is a need for a generic multi-objective optimization procedure which can work reliably with approximate models. In this paper, we have suggested a meta-modeled multi-objective optimizer, namely NSGA-II-ANN in which the well-known elitist multi-objective optimizer NSGA-II has been combined with ANN. The NSGA-II-ANN procedure has been tested on a number of test problems and a couple of practical problems, involving a curve fitting problem and an engineering shape optimization problem. Test problems and case studies involve typical features of practical optimization problems: (i) a combination of real and discrete variables, (ii) unconstrained problem and constrained problems, (iii) discontinuous and continuous Pareto-optimal fronts, and (iv) non-convex and convex Pareto-optimal fronts. Simulation results have been analyzed for two aspects, namely, convergence to the trade-off frontier and the spread of obtained solutions. It can be concluded that NSGA-II-ANN simulations generally has shown a saving in exact function evaluations of about 25% to 62% to achieve a similar performance compared to the exactly-evaluated NSGA-II.

An important finding of this study is that a single strategy (the batch learning NSGA-II-ANN with the B-10-3 model) has been found to perform consistently better in all problems. The use of three generations of exactly-evaluated solutions for training and the use of the trained ANN for the next seven generations is a winning plan found for all problems considered in this study. The problem difficulties are taken care of by the choice of an appropriate population size. The current study must now be extended and compared with other meta-modeling

techniques, such as RSM and Kriging, and the superiority of one method over the other can be established.

References

1. Branke, J., Deb, K., Dierolf, H., and Osswald M.: Finding Knees in Multi-objective Optimization. In *Proceedings of Parallel Problem Solving from Nature (PPSN 2004)*. Springer, 2004, pp.722–731.
2. Branke, J., and Schmidt, C.: Faster convergence by means of fitness estimation. In *Soft Computing Journal*. (in press).
3. Chen, J., Goldberg, D. E., Ho, S., and Sastry, K.: Fitness inheritance in multi-objective optimization. In *Proceedings, Genetic and Evolutionary computation Conference, 2002*. Morgan Kaufmann, 2002, pp. 319-326.
4. Deb, K.: *Multi-Objective Optimization Using Evolutionary Algorithms*. First Edition, Chichester, Uk: Wiley, 2001.
5. Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2):182–197, 2002.
6. Deb, K., Thiele, L., Laumanns, M., and Zitzler, E.: Scalable test problems for evolutionary multi-objective optimization. In Abraham, A., Jain, L., and Goldberg, R., editors, *Evolutionary Multiobjective Optimization, 2005*, pages 105–145. London: Springer-Verlag.
7. Deb, K., Mohan, M., and Mishra, S.: Towards a quick computation of well-spread Pareto-optimal solutions. In *Proceedings, Evolutionary Multi-Criterion Optimization, 2003*. Springer, 2003, pp. 226-236.
8. Deb, K., and Jain, S.: Running performance Metrics for evolutionary multi-objective optimization. In *Proceedings, Fourth Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*. (Singapore), 2002, pp. 13-20.
9. Deb, K. and Gupta, H.: Searching for robust Pareto-optimal solutions in multi-objective optimization. In *Proceedings of Evolutionary Multi-Criterion Optimization (EMO 2005)*, Springer-Verlag, 2005, pp. 150–164.
10. Eby, D., Averill, R. C., Punch III, W. F., and Goodman, E. D.: Evaluation of injection island GA performance on flywheel design optimization. In *Proceedings, Third Conference on Adaptive Computing in Design and Manufacturing*. Springer, 1998.
11. El-Beltagy, M. A., Nair, P. B., and Keane, A. J.: Metamodelling techniques for evolutionary optimization of computationally expensive problems: promises and limitations. In *Proceedings of the Genetic and Evolutionary Computation Conference, 1999*. Morgan Kaufman, 1999, pp. 196-203.
12. Emmerich, M., Giotis, A., Ozdenir, M., Back, T., and Giannakoglou, K.: Metamodel-assisted evolution strategies. In *Proceedings, Parallel Problem Solving from Nature, 2002*. Springer, 2002, pp. 371-380.
13. Emmerich, M., and Naujoks, B.: Metamodel assisted multi-objective optimization strategies and their application in airfoil design. In *Proceedings, Adaptive Computing in Design and Manufacture VI, 2004*. Springer, 2004, pp. 249–260.
14. Fonseca, C. M. and Fleming, P. J.: On the performance assessment and comparison of stochastic multiobjective optimizers. In *Proceedings of Parallel Problem Solving from Nature IV (PPSN-IV)*, Springer, 1996, pp. 584–593.

15. Farina, M.: A neural network based generalized response surface multi-objective evolutionary algorithms. In *Congress on Evolutionary Computation, 2002*. IEEE Press, 2002, pp. 956-961.
16. Giannakoglou, K.C.: Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. In *Progress in Aerospace Science,* Vol. 38, pp. 43-76, 2002.
17. Giotis, A. P., and Giannakoglou, K.C.: Chapter 23: low cost GAs assisted by ANNs - applications in turbomachinery. edited by *Periaux, J. et al.*, Jhon Wiley & Sons, (to appear).
18. Goldberg, D. E., Deb, K., and Clark, J. H.: Genetic algorithms, noise, and the sizing of populations. *Complex System*, 6, pp. 333-362, 1992.
19. Haykin, S.: *Neural networks a comprehensive foundation*. second edition, Singapore: Addison Wesley, 2001. pp. 208.
20. Jin, Y., Olhofer, M., and Sendhoff, B.: A framework for evolutionary optimization with approximate fitness functions. In *IEEE Transactions on Evolutionary Computation,* 6(5), pp. 481-494, 2002.
21. Jin, Y., and Sendhoff, B.: Fitness approximation in evolutionary computation - A survey. In *Proceedings, Genetic and Evolutionary Computation Conference, 2002*. Morgan Kaufmann, 2002, pp. 1105-1112.
22. Jin, Y., and Sendhoff, B.: Reducing fitness evaluations using clustering techniques and neural network ensembles. In *Proceedings, Genetic and Evolutionary Computation Conference, 2004*. Springer, 2004, pp. 688-699.
23. Nain, P.K.S., and Deb, K.: Computationally effective search and optimization procedure using coarse to fine approximation. In *Proceedings, Congress on Evolutionary Computation, 2003*. IEEE Computer Society Press, 2003, pp. 2081-2088.
24. Nair, P. B., Keane, A. J., and Shimpi, R. P.: Combining approximation concepts with genetic algorithm based structural optimization procedures. In *Proceedings of the 39th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 1998, pp. 1741-1751
25. Poloni, C., Giurgevich, A., Onesti, L., and Prdiroda, V.: Hybridization of a multiobjective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. In *Computer Methods in Applied Mechanics and Engineering*, volume 186, 2000, pp-403-420.
26. Rasheed, K., and Hirsh, H.: Informed operators: speeding up genetic-algorithm-based design optimization using reduced models. In *Proceedings, Genetic and Evolutionary Computation Conference, 2000*. Morgan Kaufmann, 2000, pp. 628-635.
27. Rasheed, K., Vattam, S., and Ni, X.: Comparison of methods for using reduced models to speed up design optimization. In *Proceedings, Genetic and Evolutionary Computation Conference, 2002*. Morgan Kaufmann, 2002, pp. 1180-1187.
28. Ratle, A.: Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In *Proceedings, Parallel Problem Solving from Nature, 1998*. volume V, 1998, pp. 87–96.
29. Reklaitis, G. V., Ravindran, A. and Ragsdell, K. M. (1983). *Engineering Optimization Methods and Applications*. New York : Wiley.
30. Sastry, K., Goldberg, D. E., and Pelikan, M.: Don't evaluate, inherit. In *Proceedings, Genetic and Evolutionary computation Conference, 2001*. Morgan Kaufmann, 2001, pp. 551-558.
31. Sefrioui, M., and Périaux, J.: A hierarchical genetic algorithm using multiple models for optimization. In *Proceedings, 6th International Conference on Parallel Problem Solving from Nature - PPSN VI* . Lecture Notes in Computer Science 1917, Springer 2000.