A Model-Assisted Memetic Algorithm for Expensive Optimization Problems

Yoel Tenne

Abstract. This chapter proposes a new model-assisted memetic algorithm for expensive optimization problems. The algorithm follows successful optimization approaches such as a combined global–local, modelling and memetic optimization. However, compared to existing studies it offers three novelties: a statistically-sound framework for selecting optimal models during both the global and the local search, an improved trust-region framework and a procedure for improved exploration based on modifying previously found sites. The proposed algorithm uses a radial basis function neural network as a global model and performs a global search on this model. It then uses a local search with a trust-region framework to converge to a true optimum. The local search uses Kriging models and adapts them during the search to improve convergence. A rigorous performance analysis is given where the proposed algorithm is benchmarked against four reference algorithms using eight well-known mathematical test functions. The individual contribution of the components of the algorithm is also studied. Lastly, the proposed algorithm is also applied to a real-world application of airfoil shape optimization where it is also benchmarked against the four reference algorithms. Statistical analysis of all these tests highlights the beneficial combination of the proposed global and local search and shows that the proposed algorithm outperforms the reference algorithms.

1 Introduction

Modern engineering design optimization replaces expensive laboratory experiments with *computer experiments*. These are computationally-intensive simulations which accurately model real-world physics. Using computer experiments reduces the cost and time of the design process and so they are used in diverse areas ranging from the design of integrated circuits [105] to complete aircraft [31].

Yoel Tenne
School of Aerospace, Mechanical and Mechatronic Engineering,
The University of Sydney, Australia
e-mail: joel.tenne@aeromech.usyd.edu.au

R. Chiong (Ed.): Nature-Inspired Algorithms for Optimisation, SCI 193, pp. 133–169.
springerlink.com

With computer experiments, the engineering design process is cast as a nonlinear optimization problem having three distinct features:

- The objective function (or cost function) being minimized depends on the simulation outputs. However, such simulations are often legacy computer codes available only as executables. As such there is no analytic expression relating the simulation inputs to its outputs and so the simulation is treated as a *black-box function*. This means a suitable optimization algorithm must rely only on the observed responses and not require the function expression or derivatives.
- Each simulation run is *expensive*, that is it requires anywhere from minutes to hours of computer run time [86, 88]. This means only a small number of simulation runs can be made.
- The underlying real-world physics and possibly the numerical solution itself often give a complicated inputs–outputs response surface, for example which is multimodal and nonsmooth [16]. This means an elaborate optimization algorithm should be used.

Due to these issues common optimization algorithms often perform poorly in such problems. For example, nonlinear programming algorithms [23] either require the function derivatives or approximate them by finite-differences which is too expensive. Heuristics and nature-inspired algorithms [70] use only the observed responses but they often require far too many function evaluations to converge.

These difficulties have motivated new optimization approaches and we review two of these in the following two subsections.

1.1 Hybrid or Memetic Algorithms

One approach to improving the optimization search is by combining several algorithms. For example, the chances of locating a good optimum are improved when the search combines both a global and local search. The global search explores the function landscape to identify promising regions. A local search then exploits local information to identify a better solution. As such, finding a good solution requires an *exploration–exploitation balance*. This approach originated in the global optimization community in the mid 1970s and has been applied in various algorithms [96, 116].

In the late 1980s researchers in the evolutionary algorithm community proposed similar approaches. Goldberg [32, p.202–204] described *hybrid schemes* which combine an evolutionary algorithm (EA) with a local search. Norman and Moscato [77] proposed a similar approach for combinatorial optimization. Moscato [74] termed such combined approaches as *memetic algorithms* following the concept of a ‘meme’ which is a unit of imitation in cultural transmission or an abstract unit of information [21]. In all these cases a population-based algorithm explores the function landscape and efficiently adapts to it [49]. A local search is also used to improve individuals (candidate solutions) by focusing on a small region.

Later studies have focused on hybridization with gradient-based algorithms such as finite-difference quasi-Newton [37, 90, 94, 103]. Others have studied hybridization

with heuristics such as the simplex or hill-climbing algorithms [93, 122]. Some have replaced the EA by a simulated annealing algorithm (SA) [43].

Recent studies have focused on improving the search by analyzing the exploration–exploitation interaction [44]. If the cost of the local search is low it may be beneficial to apply it to all individuals [55]. Another approach uses fuzzy logic to balance exploration–exploitation [123]. It is also possible to adaptively select the 'best' type of local search to use [78].

Hybrid and memetic algorithms are an active research field and recent reviews are given in [54, 81]. A book [39] and three special issues [38, 82, 83] have focused on these algorithms which indicates their significance as an emerging research field.

1.2 Reducing the Number of Expensive Evaluations

One main difficulty in expensive optimization is the tight limit on function evaluations. To illustrate the problem we consider a population-based algorithm with a population size s and which is run for a total of g generations or iterations. The total run time of the algorithm (T) depends on the time required by the optimization algorithm alone (t_a) for actions such as sorting strings or performing mutations but excluding function evaluations, and the time required by a function evaluation (t_f). As such the total run time is

$$T = g(s \times t_f + t_a). \tag{1}$$

In expensive problems each function evaluation is a simulation run which requires minutes to hours of computer time. As such we can assume that the algorithm time is negligible, that is $t_f \gg t_a$ and so

$$T \simeq g(s \times t_f). \tag{2}$$

To get an estimate of the required time we use the EA settings recommended in [50], that is a population size of $s = 50$ and $g = 1000$ generations. This means the EA requires 50,000 function evaluations for its run. If a single simulation requires an hour (but often much longer) that is $t_f = 1\,\text{hr}$ then a full run of the EA would require about 2083 days or 5.5 years. This is unacceptable in practice and shows that population-based algorithms such as EAs cannot be directly applied to expensive problems. As such several approaches have been studied to solve this difficulty.

In *fitness inheritance* only a fraction of the population is evaluated with the expensive function while the remaining candidate solutions inherit their fitness from their 'parents' [95, 107]. An extension of this approach uses clustering to assign a variable fitness to offspring [52].

A second approach uses *hierarchical* or *variable-fidelity* simulations. Here the algorithm uses several simulations which differ by their accuracy (or fidelity) and so by their computational cost. Promising solutions migrate from low- to high-fidelity simulations and vice versa [27, 102]. The approach was also used with deterministic nonlinear programming algorithms [2].

A third approach is that of *parallelization*. It does not reduce the number of expensive evaluations but only the wall-clock time needed to obtain a solution. Population-based algorithms such as EA and SA operate in a decentralized manner and so they are easily implemented on parallel machines [80, 86].

In this study we take the approach of *Modelling*. Models are computationally-cheaper approximations of the expensive black-box function. They are based on function approximation theory, that is, they interpolate the unknown function based on the observed responses [62, 69, 117]. Since they are cheaper to evaluate, a *model-assisted* algorithm uses the model instead of the expensive function during most of the search [4, 31, 106].

The framework of *model-assisted optimization*, also called *design and analysis with computer experiments* [98], involves three main components [28, 99]:

- Selecting the sites where the expensive function will be evaluated (design of experiment).
- Generating a model based on the sample and
- Assessing the model accuracy.

The early approach of Response Surface Methodology was developed for noisy real-world experiments and used least-squares quadratic models and designs which aim to counter the noise, such as full and fractional factorial designs [8, 76].

However computer experiments are deterministic and so are noiseless (the same inputs repeatedly give the same outputs). Therefore more suitable methods have been studied. Designs tailored for computer experiment are *space-filling*, meaning they spread the sample sites over the search space (instead of resampling at the same location to counter noise). These include Latin hypercube designs [67], orthogonal arrays [84] and maximin designs [47]. Also, the lack of noise motivates using more flexible models which interpolate more accurately than the least-squares quadratics. Such models include neural-networks [6, 40] (also discussed in Sect. 3.3), Kriging [20] (also discussed in Sect. 3.5.2) and radial basis functions (RBFs) [11].

Whatever model is used, it is likely to be inaccurate due to the small sample size. It is then necessary to estimate the degree of inaccuracy since a poor model can drive the optimization search to a *false optimum*, that is an optimum of the model but not of the true function [46]. One approach to estimate the model accuracy is with statistics of goodness-of-fit [76, p.28–36], [120]. Another is with resampling methods which train a model using part of the available sample and test the model using the remaining part [61, Ch.2]. Recent studies have compared various methods for model accuracy assessment [68, 112].

1.3 Model-Assisted Algorithms

The modelling approach has proven to be efficient and effective and as such several classes of model-assisted algorithms have emerged.

One class of algorithms uses Kriging models in a Bayesian statistics framework. Kriging models are a statistical-oriented approach to interpolation and are discussed in length in Sect. 3.5.2. Briefly, a Kriging model treats the black-box function as a

combination of a deterministic function and a Gaussian random process. Statistical methods select the parameters of these functions to improve the model accuracy. After generating the Kriging model the algorithm seeks the site which is expected to either improve the best value found so far or to improve the overall model accuracy. These two objectives are combined to give a merit value known as the 'expected improvement' (EI). Sites already evaluated have an EI = 0 while all others have an EI which reflects both the predicted model value and the uncertainty about this prediction. At each iteration the algorithm samples the site having the highest EI value and it updates the model, which then results in new EI values for all sites. As such the EI approach balances between global and local search. The approach originated in 1960s with Kushner's univariate method [57]. Later studies extended it to multivariate functions [34, 110] and incorporated the Bayesian framework [72, 119], including the recent paper by Jones et. al. [48].

Another class of algorithms uses quadratic interpolants as models, motivated by a Taylor series function approximation. Quadratics both account for function curvature which assists the optimization and are simple to optimize. Algorithms in this class combine quadratics with a trust-region framework to ensure convergence to an optimum of the true expensive function [17]. Winfield studied in 1970s an early approach of a model-assisted algorithm using quadratics [121]. Powell [89] and Conn et. al. [14, 15] have later improved the approach based on recent advances in interpolation theory.

A third class is the *surrogate-management framework*. It uses a variant of Torczon's pattern search algorithm [115] as the search algorithm. The pattern search seeks the optimum of the current model (termed 'Search Step'). If it fails to find a new optimum then the model is refined (termed 'Poll Step') [7]. No restriction is made on the model type.

A fourth class is that of *model-assisted memetic algorithm*. The idea is to generate a model and seek its optimum with a memetic algorithm. A number of candidate solutions are then evaluated with the expensive function, the model is updated and the process repeats. One algorithm combines an EA, a neural network and a local search [30, 88]. Other algorithms combine an EA with global and local radial basis function models [79, 80, 114, 126, 127]. An algorithm which combines an EA with quadratic models and a local search was studied in [60, 113].

The latter class has proven to be both efficient and effective and following its success we propose a new model-assisted memetic algorithm for expensive optimization problems. Briefly, the algorithm first trains and selects a global model which is an artificial neural network and seeks the optimum of this model. It then uses a local search to improve this predicted optimum. This sequence is repeated until the number of function evaluations reaches the prescribed limit. Compared to existing studies the proposed algorithm contains three main novelties:

- Model selection and model management using statistical model selection: typically there will be a family of candidate models. Due to lack of domain knowledge the user often chooses a non-optimal model which degrades the optimization search. To address this and to improve the search the proposed algorithm selects all models under a unified and statistically-sound framework

of model selection. This yields optimal models which are adapted during the search.
- Improved trust-region framework: to converge to a true optimum the proposed algorithm uses a trust-region approach. We describe several improvements to this approach, such as selecting sites to improve the model and more efficient stopping criteria. Further, we replace the quadratic models (in the classical approach) with Kriging model and adapt the models during the search.
- Improved global exploration: a model can lead the optimization search to converge repeatedly to the same optimum without promoting exploration of the search space, a condition termed 'model stall'. To address this we propose a modification of sampled sites which promotes exploration and discovery of new optima. The method is computationally-efficient and applicable to any type of model.

Rigorous performance analysis shows the proposed algorithm outperforms several reference algorithms.

This chapter is organized as follows: Sect. 2 reviews concepts of model selection theory relevant to the proposed algorithm. Section 3 then describes in detail the proposed algorithm and Sect. 4 provides a rigorous performance analysis. It is followed by Sect. 5 which summarizes this chapter.

2 Model Selection and Complexity Control

A major aspect of the proposed algorithm is the selection of optimal models. This section briefly explains the basics of statistical model selection theory and focuses on the approach used in the proposed algorithm.

In a model selection problem we are given a set of sites and responses generated by an unknown function and we wish to select a model which best describes this function [12, 61]. The model is selected from a family of candidate models. The statistical theory of model selection uses a *discrepancy* (Δ), which is the mismatch between model predictions and the true responses, to measure the goodness-of-fit of a candidate model to the given data. The discrepancy is calculated for each candidate model and the model chosen is the one having the smallest discrepancy.

Based on information theory we consider the *Kulback-Leibler discrepancy* [56]. It uses the *likelihood* of a candidate model given the data, that is the conditional probability of observing the sample of sites and responses

$$\mathcal{X} = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i)\}, \quad i = 1 \ldots n, \tag{3}$$

under the model $\mathcal{S}(\boldsymbol{x})$, or $\mathcal{L}(\mathcal{S}|\mathcal{X})$ [87]. The discrepancy is then

$$\Delta = -\log \mathcal{L}(\mathcal{S}|\mathcal{X}). \tag{4}$$

With the Kulback-Leibler discrepancy the optimal model is the one having the *maximum likelihood*, that is

$$\Delta_{\min} = -\log \mathcal{L}_{\max}(\mathcal{S}|\mathcal{X}). \tag{5}$$

For some models a closed-form expression exists for the likelihood and the discrepancy. Otherwise, an empirical discrepancy [61, Ch.5] is used where

$$\Delta = \frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|}(\mathcal{S}(\boldsymbol{x}) - f(\boldsymbol{x}))^2, \tag{6}$$

which is the mean of sum of squared error between the true responses and the model predictions over the sample. Since the likelihood is a function of the model (and hence of the model parameters), maximizing the likelihood by solving (4) gives the optimal model parameters [65].

The family of feasible models may contain models of different *complexity*, that is the number of model parameters. More complex models may fit the sample better than simpler ones but their prediction at new sites (or *generalization*) may be poor, a condition termed *over-fitting* [6, Ch.9]. Often a simpler model may be a better approximation of the true function.

This motivates the selection of models based not only on their discrepancy but also on their complexity. As such we consider the *Akaike information criterion* (AIC) for complexity control [1], [61, p.243–245]. The criterion uses the Kulback-Leibler discrepancy but adds a penalty which increases with model complexity where

$$\text{AIC} = -\log \mathcal{L}(\mathcal{S}|\mathcal{X}) + 2m, \tag{7}$$

and m is the number of model parameters. As such a more complex model is preferred over a simpler one only if it is significantly more accurate. The optimal model is the one having the lowest AIC value.

The AIC was derived under asymptotic assumptions of a large sample. However in expensive optimization problems the sample is small and the AIC becomes biased [3, 59]. As such we use the *corrected Akaike information criterion* (AIC_c) which is unbiased for small samples where

$$\text{AIC}_\text{c} = \text{AIC} + \frac{2(m+1)(m+2)}{n-m-2}, \tag{8}$$

and n is the sample size [42]. The AIC_c has performed well against other complexity control criteria [3, 42].

3 The Proposed Algorithm

3.1 Initialization and Main Loop

The proposed algorithm begins by generating a Latin hypercube design (LHD) consisting of $k = 0.2fe_{\max}$ sites, where $fe_{\max}$ is the prescribed limit of expensive evaluations. As mentioned in Sect. 1.2, this design improves the accuracy of the resultant model by spreading the sites in the search space.

To generate a LHD of k sites the range of each variable is split into k equally sized intervals and one point is sampled at random in each interval. To create a LHD site a sampled point is selected at random (without replacement) for each variable and these samples are combined to give a site (a vector). The process is repeated until k sites have been created. Algorithm 1 gives a pseudocode for generating a LHD sample.

Algorithm 1. Generating a LHD sample

```
inputs
    number of variables (d);
    sample size (k);
    bounds on variables;

for each variable i = 1...d do
    divide the variable range into k equal intervals;
    sample one point (a scalar) at random in each interval;
for each LHD site x_j, j = 1...k do
    for each variable i = 1...d do
        select at random and without replacement a sample point;
        set ith component of site j (that is x_{j,i}) to selected sample point;
Output: a Latin hypercube design of size k
```

The expensive function is then evaluated at the LHD sites. During the search the algorithm caches all sites evaluated with the expensive function and their responses to reuse them later in the search and to reduce new evaluations.

The main loop then begins and the algorithm generates a global model of the objective function, as follows. It first uses a modified copy of the cache where sites found during previous local searchs have been 'masked' (as described in Sect. 3.2). It then uses this modified copy to train and select an artificial neural network which serves as the global model (as described in Sect. 3.3). Next, it seeks the optimum of the global model (as described in Sect. 3.4) and improves this predicted optimum with a local search (as described in Sect. 3.5). This sequence is

Algorithm 2. Main loop

```
generate an initial LHD;
evaluate and cache sites;
while fe ⩽ fe_max do
    if local searchs have been made then
        create a copy of the cache and 'mask' sites found during local searchs;
    using the (modified) cache train and select a global model;
    select initial site for the local search (the model's predicted optimum);
    improve predicted optimum with a local search;
Output: best solution and response found
```

repeated until the number of function evaluations reaches the prescribed limit $fe_{\max}$ ($fe_{\max} = 100, 150$ and 200 were used for performance analysis). A pseudocode of the main loop is given in Algorithm 2.

3.2 Modifying the Cache to Assist Exploration

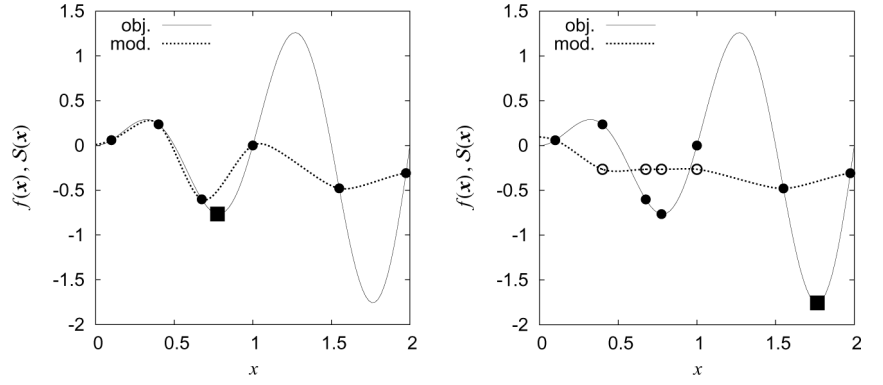
The global model is trained using the cached sites and responses. Elite sites (having a low response), which are typically found during the local searches, can 'dominate' the model so other minima will not be represented. To avoid this and to promote exploration such elite sites (and nearby sites) are identified and 'masked' by setting their response to the mean response of all cached sites. To identify sites nearby elites the proposed algorithm clusters the cached sites. All sites in clusters with sites found in previous local searchs have their responses set to the mean of responses in the cache ($\bar{f}$). When the global model is trained using this modified cache it will not be dominated by the elite sites and will promote exploration. The algorithm uses the mean response to avoid adding artificial multimodality to the global model.

For clustering the algorithm uses the k-harmonic means algorithm [124, 125] which is efficient and has outperformed competing algorithms such as the popular k-means [35]. At each iteration the algorithm finds the location of the k cluster centres as the harmonic mean of the distance to all sites, so all sites are accounted for. This improves the clustering quality and differs from the popular k-means algorithm where only the nearest sites to a centre are accounted for. Algorithm 3 gives a pseudocode of the k-harmonic means algorithm.

The k-harmonic means requires the number of clusters (k) as an input and so to find the optimal k the proposed algorithm uses a model selection approach. Following

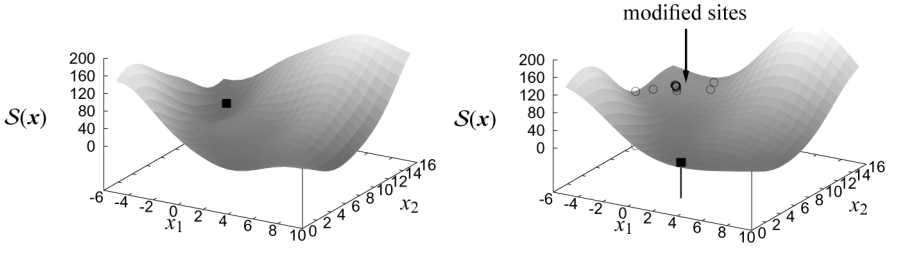
Algorithm 3. k-harmonic means clustering

Input: sites to cluster $\boldsymbol{x}_j$, $j = 1 \ldots n$
set $t = 1$; /* iterations counter */
initialize centres $\boldsymbol{c}_i$, $i = 1 \ldots k$ at random;
repeat
  for $i = 1 \ldots k$ **do** scan over clusters
    for $j = 1 \ldots n$ **do** scan over sites
      $d_{i,j} = \|\boldsymbol{c}_i - \boldsymbol{x}_j\|_2$; /* distance of centre i to site j */
      $q_{i,j} = {d_{i,j}}^3 \left(\sum_{p=1}^{k} \frac{1}{{d_{p,j}}^2} \right)^2$;
    $\boldsymbol{c}_i = \frac{\sum_{j=1}^{n} \frac{1}{q_{i,j}} \boldsymbol{x}_j}{\sum_{j=1}^{n} \frac{1}{q_{i,j}}}$; /* new centre i is harmonic mean */
  t = t + 1;
until *change in centres is small or max. iterations* ;



(a) First local search (b) Second local search

Fig. 1 A 1-D example of the proposed cache modification. The objective function is $f(x) = x\sin(x)$. The sampled sites, objective function and model are shown. (a) shows the first local search finds the local optimum at $x = 0.75$ (■). Sites are then clustered and those in the cluster containing the found optimum have their responses modified (⊙). (b) shows this leads to a model which identifies the second optimum such that the second local search now finds to the optimum at $x = 1.75$



(a) First local search (b) Second local search

Fig. 2 A 2-D example of the proposed cache modification. The objective function is Branin. (a) shows the first local search finds the local optimum at $(-3.1, 12.2)$ (■). Next, cached sites are clustered and those in the cluster containing the found optimum have their responses modified (⊙). (b) shows the second local search used with the model based on the modified sites now converges to a different optimum at $(3.1, 2.2)$ (■)

Sect. 2, the optimal k is found by minimizing the corrected Akaike information criterion ($\mathrm{AIC_c}$), where the discrepancy function is the inter-cluster error

$$\Delta = \sum_{i=1}^{k} \sum_{j=1}^{n_i} \|\boldsymbol{c}_i - \boldsymbol{x}_j^{(i)}\|_2 , \tag{9}$$

where n_i is the number of sites in cluster i, $\boldsymbol{c}_i$ is the ith cluster centre and $\boldsymbol{x}_j^{(i)}$ is the jth site in cluster i. This is a univariate minimization problem of minimizing $\mathrm{AIC_c}(k)$. It is solved with Brent's golden-search algorithm [9].

Figures 1 and 2 give a 1D and 2D example, respectively, of the proposed method and Algorithm 4 gives a pseudocode of the proposed method.

Algorithm 4. Modifying the cache to assist exploration

Input: cache of sites and responses;
create a copy of cache;
find mean response in cache $\bar{f}$;
cluster cached sites using k-harmonic means, find optimal k with $\mathrm{AIC_c}$ criterion;
for clusters containing a site found in previous local searchs set all responses to $\bar{f}$;
Output: modified copy of the cache

3.3 Generating the Global Model

Next, the proposed algorithm uses the modified copy of the cache to train a global model of the expensive function. Such a model can be a Lagrangian interpolant so it interpolates exactly at all available sites, that is

$$\mathcal{S}(\boldsymbol{x}_i) = f(\boldsymbol{x}_i),\ i = 1 \ldots n\ (= \text{cache size}), \tag{10}$$

where $\mathcal{S}(\boldsymbol{x}_i)$ is the model response at the ith site and $f(\boldsymbol{x}_i)$ is the true response there. However, there are two difficulties with such models:

a) they can generalize poorly due to over-fitting to the given data (as mentioned in Sect. 2) so their prediction is likely to be poor at new sites and
b) they become computationally-expensive to handle (since they account for all sites) and numerically unstable (due to ill-conditioning of the interpolation matrix) [22, 51].

To avoid these issues the proposed algorithm uses a *radial basis function network* (RBFN) for the global model which is an artificial neural network with radial basis functions processing units. Artificial neural networks are a form of nonparametric regression [6, 40] and can approximate a continuous function with high accuracy (given sufficient sites) [6, Ch.4]. RBFNs have the advantage of approximating well complicated function landscapes while their structure is simpler compared to other networks and so they are faster to train [6, Ch.5], [10, 73].

Figure 3 shows a typical RBFN. It contains three layers: the input layer, the processing layer containing the processing units (or neurons) and the output layer which is a weighted sum of the units responses. The proposed algorithm uses an RBFN with Gaussian processing units which is equivalent to approximating the objective function by a superposition of Gaussians [69]. The response of this RBFN is

$$\mathcal{S}(\boldsymbol{x}) = \sum_{j=1}^{N} \lambda_j \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{t}_j\|_2^2}{{c_j}^2}\right), \tag{11}$$

where N is the number of processing units, λ_j is a coefficient, $\boldsymbol{t}_j$ is the centre of the jth Gaussian and c_j is the shape parameter (or hyperparameter) which determines the width of the jth Gaussian.

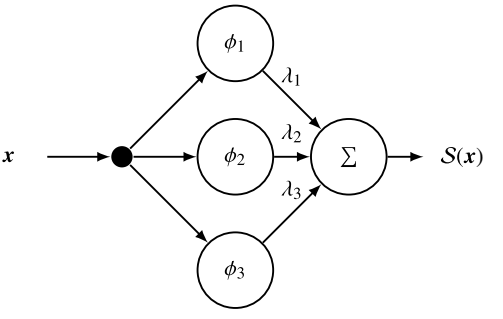


Fig. 3 An RBFN with three neurons (processing units)

An RBFN generalizes well and avoids over-fitting by abstracting the data. This is achieved by using fewer processing units than sample sites ($N < n$) so the centres ($\boldsymbol{t}_j$) typically do not coincide with the sample sites. Training an RBFN requires selecting the number of processing unit (N), the centres ($\boldsymbol{t}_j$), the shape parameters (c_j) and the coefficients (λ_j) . To efficiently select all these the proposed algorithm uses the following two steps.

First, it identifies the optimal number of processing units (N). For a candidate number of processing units the cached sites are clustered using the k-harmonic means algorithm (described in Sect. 3.2) and the resulting centres are taken as the Gaussians' centres. The shape parameters (c_j) are taken as the radii of the corresponding clusters. The coefficients λ_j are obtained from the least-squares solution of the interpolation equations as

$$\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\lambda} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{f}, \tag{12}$$

where $\boldsymbol{\Phi}$ is the interpolation matrix such that

$$\boldsymbol{\Phi} : \Phi_{i,j} = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{t}_j\|_2^2}{{c_j}^2}\right), \tag{13}$$

and $\boldsymbol{f}$ is the vector of responses. This linear system is solved by the truncated singular value decomposition method (TSVD) since $\boldsymbol{\Phi}$ may be ill-condtioned [6, p.170–171].

All these parameters define an RBFN with N processing units. Similar to Sect. 3.2, finding the optimal number of processing units is treated as a model selection problem and is solved using the corrected Akaike information criterion ($\mathrm{AIC_c}$). For a network with N units the algorithm finds the empirical discrepancy (6) calculated over all cached sites. Each value of N defines a specific network and a corresponding $\mathrm{AIC_c}$. As such the algorithm finds the optimal N by minimizing $\mathrm{AIC_c}(N)$ using Brent's algorithm [9].

The proposed algorithm then optimizes the shape parameters. Taking the shape parameters found in the previous step ($\boldsymbol{c} = (c_1, \ldots c_N)$) as baseline values it finds the factor l which minimizes the discrepancy (6) of a network with shape parameters $l\boldsymbol{c}$. The number of centres (N) and their location ($\boldsymbol{t}_j$) are fixed to those found in the previous step, but the coefficients λ_j are recalculated for each candidate l value. Similarly to the previous stage, each l value defines a specific network and a corresponding discrepancy and so the algorithm finds the optimal l by minimizing $\Delta(l)$ using Brent's algorithm. Algorithm 5 gives a pseudocode of the proposed method for generating the RBFN global model.

Algorithm 5. Generating the global model

```
Input: modified copy of cache;
optimize number of units (N) by minimizing AIC_c(N):
begin
    for each candidate N cluster sites using k-harmonic means;
    set Gaussian centres (t_j) to cluster centres;
    set Gaussian widths (c_j) to cluster radii;
    find coefficients (λ_j) by least-squares;
    find discrepancy of candidate network and its AIC_c;
end
set c as shape parameters for optimal N;
optimize shape parameters by minimizing the discrepancy Δ(l):
begin
    for each candidate l set Gaussian widths to lc;
    generate network (find coefficients by least-squares);
    find discrepancy of candidate network;
end
Output: an RBFN global model with optimized parameters
```

3.4 Selecting the Starting Site for the Local Search

Next, the proposed algorithm seeks the global optimum of the global model. It uses a real-coded EA [13] followed by a gradient-based finite-differences quasi-Newton BFGS algorithm [23, Ch.5–6]. The EA uses a population size $s_{\text{pop}} = 50$, linear ranking, stochastic universal sampling (SUS), intermediate recombination, elitism with a generation gap $g_{\text{gap}} = 0.9$ and the breeder-genetic-algorithm mutation operator with probability $p_{\text{m}} = 0.05$ [75]. The evolutionary search is stopped when no improvement is observed after $g_{\text{n.i.}} = 10$ generations. The gradient-search then improves the best site found by the EA and this gives the predicted optimum ($\boldsymbol{x}_{\text{p}}$) of the global model.

The predicted optimum is then evaluated with the expensive objective function to give the true response $f(\boldsymbol{x}_{\text{p}})$. If $f(\boldsymbol{x}_{\text{p}})$ is better than the best value found so far then the local search is started from $\boldsymbol{x}_{\text{p}}$. Otherwise, this indicates the model is inaccurate and so the algorithm improves the model by adding a new site to it.

The accuracy of interpolants depends on the spread of the interpolation sites, measured by the *maximin distance* [63, 100]. For a set of sites $\boldsymbol{x}_i$, $i = 1 \ldots k$ the maximin distance is the maximum of all nearest-neighbour distances in the set. The model accuracy improves as the maximin distance increases, that is as the sites are more space-filling. As such, the proposed algorithm improves the global model by seeking the site which maximizes the maximin distance for the cached sites. It finds this site ($\boldsymbol{x}_\mathrm{n}$) by solving the nonlinear optimization problem

$$\boldsymbol{x}_\mathrm{n} : \max_{\boldsymbol{x} \in \mathcal{F}} \min_{\boldsymbol{x}_i \in \mathcal{X}} \{ \|\boldsymbol{x}_i - \boldsymbol{x}\|_2 \} \tag{14}$$

where $\mathcal{X}$ is the set of cached sites and $\mathcal{F}$ is the search space. This approach generates sites similarly to the *maximin design of experiments* [47].

The new site is evaluated with the expensive function and is cached. The global model is then updated and the process repeats until either a better optimum is found or 10 attempts have been made. In the latter case the best cached site is taken as the starting site for the local search. Algorithm 6 gives a pseudocode for the proposed method for selecting the starting site.

Algorithm 6. Selecting the starting site for the local search

```
Input: cache and modified copy of cache;
set i = 1 ;                                          /* number of attempts */
find best site in cache (x_b);
repeat
    generate global model;
    seek optimum of model using an EA followed by a gradient search (SQP);
    evaluate the predicted optimum (x_p) with expensive function and cache;
    if optimum is better then current best in cache then
        set x_0 = x_p ;                               /* set starting site */
    else
        improve model by searching for a site (x_n) using maximin distance;
        evaluate x_n with expensive function and cache;
        i = i + 1;
    if i = 10 then set x_0 = x_b ;                    /* set starting site */
until f(x_p) < f(x_b) or i = 10 ;
Output: initial site for local search (x_0)
```

3.5 Improving the Optimum with the Local Search

Next, the proposed algorithm improves the starting site ($\boldsymbol{x}_0$) by using a local search. The proposed local search has three distinct features: a) since it concentrates on a small region, it uses local models to better model the objective function b) it uses an improved trust-region framework to converge to a true optimum of the expensive function but it replaces the classical quadratic models with more flexible ones to improve the search and c) to further assist convergence it continuously adapts the type of model used.

3.5.1 Using a Trust-Region Framework to Converge to a True Optimum

As mentioned in Sect. 1.3, due to model inaccuracy a model-assisted optimization search can converge to a false optimum. To avoid this the proposed algorithm uses a trust-region framework.

The classical trust-region framework generates at each iteration a quadratic model and obtains its constrained optimum (a truncated Newton step) as a quadratic programming problem [17]. The framework guarantees asymptotic convergence to an optimum of the true objective function (a critical site satisfying the first order Karush-Kuhn-Tucker optimality conditions).

For quadratic models the constrained optimum is easily found but such models cannot model a complicated landscape well. As such, the proposed local search replaces them with the more flexible Kriging models which are described in Sect. 3.5.2 which follows. When compared to quadratics, Kriging models can approximate the objective function better over a larger trust-region and so convergence will be faster and require less function evaluations.

The proposed trust-region local search begins with an initial cuboid trust-region centred at $\boldsymbol{x}_{\mathrm{c}} = \boldsymbol{x}_0$ (the starting site) and of size $\Delta = 0.1$, that is

$$\mathcal{T} = \{\boldsymbol{x} : \|\boldsymbol{x}_{\mathrm{c}} - \boldsymbol{x}\|_\infty \leqslant \Delta\}. \tag{15}$$

To emphasize local function behaviour all cached sites which are in $\mathcal{T}$ are used to generate the local model. If the trust-region contains less than $m = \min\{d + 1, 10\}$ sites then the nearest exterior sites are also used. The algorithm selects the optimal type of Kriging model (as described in Sect. 3.5.2) and finds its constrained optimum in the trust-region ($\boldsymbol{x}_{\mathrm{m}}$). However, this is no longer a simple quadratic programming problem (as in the classical framework) and so to find the constrained optimum the algorithm uses an EA followed by a gradient-search, similarly to Sect. 3.4.

Following the classical trust-region approach the objective function is evaluated at the predicted optimum and a merit value is calculated

$$\rho = \frac{f(\boldsymbol{x}_{\mathrm{m}}) - f(\boldsymbol{x}_{\mathrm{c}})}{\mathcal{S}(\boldsymbol{x}_{\mathrm{m}}) - \mathcal{S}(\boldsymbol{x}_{\mathrm{c}})}, \tag{16}$$

where $\mathcal{S}(\boldsymbol{x})$ is the current Kriging local model. A value of $\rho \simeq 1$ indicates a good fit of the model to the objective function in the trust-region.

The classical trust-region framework assumes exact derivatives are available so a poor model fit is only due to a trust-region which is too large and so it decreases the trust-region. However, in model-assisted search the model may be inaccurate due to an insufficient number of interpolation sites in the trust-region. This needs to be accounted for to avoid a quick reduction of the trust-region and premature convergence [15].

As discussed in Sect. 3.4, the model's accuracy depends on the maximin distance of the interpolated sites. As such, the proposed algorithm determines if a model is sufficiently accurate (to justify reducing the trust-region) based on the number of space-filling sites in the trust-region. The maximum separation distance for a cuboid

trust-region is the diagonal length $\sqrt{d(2\Delta)^2}$ (d is the function dimension). A site is considered space-filling if its nearest-neighbour distance is at least 5% of the diagonal length. The model is considered accurate when the number of space-filling sites in the trust-region (s) is larger than a threshold value ($s^\star = d+1$). This value is based on the number of sites required to model the gradient by well-established methods like quasi-Newton finite-differences [15]. However, as d increases the required number of sites becomes comparable to the total number of function evaluations ($fe_{\max}$). As such, the algorithm uses $s^\star = \min\{d+1\,,\, 0.1fe_{\max}\}$.

Based on ρ, s and $s^\star$ the algorithm performs as follows:

- if $\rho > 0$: then the local model is accurate since a better solution has been found. Following the classical trust-region framework the algorithm centres the trust-region at the new optimum ($\boldsymbol{x}_{\mathrm{m}}$) and the trust-region is enlarged by a factor δ_+.
- if $\rho \leqslant 0$ and $s < s^\star$: the optimum predicted by the model is a false one, but the poor model accuracy is attributed to an insufficient number of space-filling sites in the trust-region. As such, the algorithm improves the local model by adding a new site $\boldsymbol{x}_{\mathrm{n}}$. Similar to Sect. 3.4, this site ($\boldsymbol{x}_{\mathrm{n}}$) is chosen to give the largest maximin distance with respect to all sites in the trust-region. $\boldsymbol{x}_{\mathrm{n}}$ is evaluated with the expensive function and is cached. If $f(\boldsymbol{x}_{\mathrm{n}}) < f(\boldsymbol{x}_{\mathrm{c}})$ than $\boldsymbol{x}_{\mathrm{n}}$ becomes the new trust-region centre.
- if $\rho \leqslant 0$ and $s \geqslant s^\star$: the local model fails to predict an improvement but its poor accuracy is attributed to the trust-region being too large (the model is considered to be accurate). Following the classical trust-region framework the algorithm decreases the trust-region by a factor δ_-.

As such the local search uses at most two expensive evaluations at each iteration, one for $\boldsymbol{x}_{\mathrm{c}}$ and possibly another for $\boldsymbol{x}_{\mathrm{n}}$. All new sites evaluated with the expensive function are cached.

Next, the local search stops if the trust-region is small enough $\Delta < \Delta_{\min}$ (we use $\Delta_{\min} = \Delta_0 \cdot \delta_-^2$) or if the limit of expensive evaluations has been reached. Otherwise, a new local search iteration begins. Algorithm 7 gives a pseudocode of the proposed trust-region local search.

3.5.2 Selecting Optimal Local Models

To assist the local search the proposed algorithm selects at each iteration an optimal local model. It selects models from a family of Kriging (or spatial-correlation) models as they have performed well compared to other models [45, 58].

Kriging models originated in geostatistics with the work of Krige and Matheron [19, 66]. Such a model has two components: a 'drift' function which models global variations in the objective function and a stochastic function (a stationary Gaussian process) which locally improves the prediction [20].

A common approach is to use a constant drift function (for example set to 1) [53] so the Kriging model is

$$\mathcal{S}(\boldsymbol{x}) = \beta + Z(\boldsymbol{x}), \tag{17}$$

Algorithm 7. A trust-region framework for the local search

inputs
  $\mathcal{X}$; /* cache of sites and responses */
  $\boldsymbol{x}_0$; /* initial site for local search */

$s^\star = \min\{d+1, 0.1fe_{\max}\}$; /* model accuracy threshold */
$\boldsymbol{x}_c = \boldsymbol{x}_0$; /* centre trust-region at $\boldsymbol{x}_0$ */
repeat
  define a cuboid trust-region centred at $\boldsymbol{x}_c$ and of size Δ;
  find the cached sites inside the trust-region (if insufficient use exterior sites);
  select an optimal local model based on these sites;
  find model optimum in trust-region ($\boldsymbol{x}_m$) with EA and gradient-search;
  calculate a trust-region merit value ρ;
  update trust-region:
    if $\rho > 0$ set $\boldsymbol{x}_c = \boldsymbol{x}_m$, increase Δ
    if $\rho \leqslant 0 \bigcap s < s^\star$ improve local model by adding a site $\boldsymbol{x}_n$, if better set $\boldsymbol{x}_c = \boldsymbol{x}_n$
    if $\rho \leqslant 0 \bigcap s \geqslant s^\star$ decrease Δ
until $\Delta < \Delta_{\min}$ *or* $fe \geqslant fe_{\max}$;
Output: optimum found in local search

where β is the drift function coefficient and $Z(\boldsymbol{x})$ is the stochastic function [98]. The latter is taken as a Gaussian process with a zero mean and variance σ . The response of the Kriging model at any site is correlated with that of other sites. The correlation between two sites ($\boldsymbol{x}_1$ and $\boldsymbol{x}_2$) is defined by a covariance function

$$C(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sigma^2 R(\boldsymbol{x}_1, \boldsymbol{x}_2), \tag{18}$$

where $R(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is a spatial correlation function (SCF). The model is defined by adjusting the free coefficient (β) and the SCF parameters to fit the available data.

Different spatial correlation functions have been studied [53]. Each SCF results in a different model and so the optimal SCF is problem dependant. In practice the SCF is prescribed and fixed throughout the optimization search [71]. To improve this, the proposed algorithm uses the model selection framework (described in Sect. 2) to select the optimal SCF based on maximum likelihood. The likelihood of a Kriging model is given by the closed-form expression [98]

$$\mathcal{L} = -\frac{d}{2}\log(2\pi\sigma^2) - \frac{1}{2}\log(|\boldsymbol{R}|) - \frac{1}{2\sigma^2}(\boldsymbol{f} - \boldsymbol{1}\beta)^{\mathrm{T}}\boldsymbol{R}^{-1}(\boldsymbol{f} - \boldsymbol{1}\beta), \tag{19}$$

where

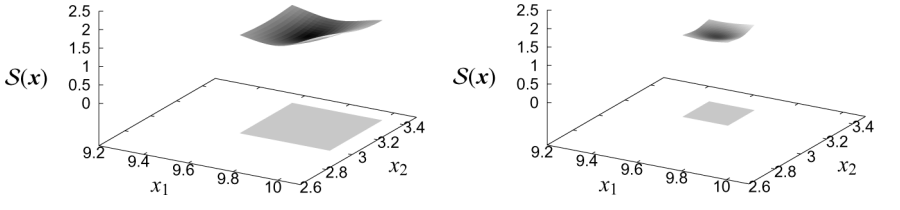
$$\boldsymbol{R} : R_{i,j} = R(\boldsymbol{x}_i, \boldsymbol{x}_j, \theta) \tag{20}$$

is the correlation matrix of all sites in the sample, θ is a correlation parameter and the spatial correlation function is given by

Table 1 Candidate spatial correlation functions (SCFs)

Name	$\mathcal{R}(\theta, l_k)$
Exponential	$\exp(-\theta\lvert l_k\rvert)$
Gaussian	$\exp(-\theta l_k^2)$
linear	$\max\{0, 1-\theta\lvert l_k\rvert\}$
spherical	$1-1.5\xi_k+0.5\xi_k^3, \quad \xi_k=\min\{1, \theta\lvert l_k\rvert\}$
spline	$\zeta(\xi_k)=\begin{cases}1-15\xi_k^2+30\xi_k^3 & 0\leqslant\xi_k\leqslant 0.2\\ 1.25(1-\xi_k)^3 & 0.2<\xi_k<1\\ 0 & \xi_k\geqslant 1\end{cases}$ $\xi_k=\theta\lvert l_k\rvert$

$l_k = \boldsymbol{x}_{i,k} - \boldsymbol{x}_{j,k}$ is the difference between the kth component of two sites $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ [109].



(a) Iteration 5 (b) Iteration 14

Fig. 4 An example of models used during the local search with the Branin function. (a) shows iteration 2 where the local model used a Gaussian SCF. (b) shows iteration 14 where the local model used a spline SCF. The trust-region is also shown.

Algorithm 8. Selecting optimal local models

Input: sites and responses used for the local model
for *SCF = exponential, Gaussian, linear, spherical, spline* **do**
  generate Kriging model using candidate SCF;
  find the model's maximum likelihood;
select the Kriging model having the largest maximum likelihood;
Output: optimal Kriging local model

$$R(\boldsymbol{x}_i, \boldsymbol{x}_j, \theta) = \prod_{k=1}^{d} \mathcal{R}(\theta, l_k), \quad l_k = \boldsymbol{x}_{i,k} - \boldsymbol{x}_{j,k}, \tag{21}$$

where the functions $\mathcal{R}(\theta, l_k)$ are defined in Table 1. The model parameters (β, σ and θ) are found by maximizing its likelihood (19) [65]. The numerical procedures for generating the Kriging models are given in [109].

As such, at each local search iteration the proposed algorithm builds a Kriging model, one for each of the SCFs in Table 1, and it selects the one giving the largest maximum likelihood. Complexity control is unnecessary since all models have equal complexity, that is the three parameters σ, β and θ. This approach results in a model-adaptive local search. Figure 4 shows an example and Algorithm 8 gives a pseudocode of the proposed method.

3.6 Caching New Sites

During the optimization search new sites and responses are cached for later use. If cached sites are nearly collocated the interpolation matrices used to generate the global and local models will be severely ill-condtioned. To avoid this a new site is added to the cache if it is sufficiently spaced from cached sites (a minimum l_2 distance of $\Delta_{\min}/2$, where $\Delta_{\min}$ is the prescribed minimum trust-region radius, as described in Sect. 3.5.1). Otherwise the new site replaces the cached site nearest to it (in the l_2 norm) if the new response is better than the cached one. Algorithm 9 gives the pseudocode for caching new sites.

Algorithm 9. Caching new sites

Input: $\boldsymbol{x}_{\text{new}}, f(\boldsymbol{x}_{\text{new}})$; /* new site and response */
find the cached site $\boldsymbol{x}_{\text{cac}}$ nearest to $\boldsymbol{x}_{\text{new}}$;
if $\|\boldsymbol{x}_{\text{cac}} - \boldsymbol{x}_{\text{new}}\|_2 \geqslant \Delta_{\min}/2$ **then**
  add $\boldsymbol{x}_{\text{new}}$ and $f(\boldsymbol{x}_{\text{new}})$ to the cache;
else if $f(\boldsymbol{x}_{\text{new}}) < f(\boldsymbol{x}_{\text{cac}})$ **then**
  replace $\boldsymbol{x}_{\text{cac}}$ and $f(\boldsymbol{x}_{\text{cac}})$ with $\boldsymbol{x}_{\text{new}}$ and $f(\boldsymbol{x}_{\text{new}})$;

4 Performance Analysis

This section gives a detailed performance analysis of the proposed algorithm in three parts. First, we test the proposed algorithm on eight well-known mathematical test functions. In these tests it is also benchmarked against four reference algorithms.

Next, we study the individual contribution of the global search and of the local search components of the proposed algorithm. This is done by comparing the full proposed algorithm to the two cases where its global search is disabled and where its local search is disabled.

Lastly, we apply the proposed algorithm to a real-world application of airfoil shape optimization and we also benchmark it against the four reference algorithms.

4.1 Reference Algorithms and Test Procedure

To obtain a reference of performance we benchmarked the proposed algorithm against four representative model-assisted EAs [91, 92]. These algorithms build a

Algorithm 10. Reference algorithm

```
generate initial sites with LHD and evaluate them;
cache sites and responses;
while fe ⩽ fe_max do
    generate a Kriging model based on cache;
    search for model optimum with an EA for 10 generations;
    evaluate candidate solutions from population with true function;
    cache evaluated candidate solutions and their responses;
Output: best solution and response found
```

Kriging model, seek its optimum and evaluate a certain percentage of the population with the true function. The model is then updated and the process repeats until the limit of function evaluations is reached. Algorithm 10 gives a pseudocode of the reference algorithms.

The four algorithms differ by their Kriging spatial correlation function (SCF) and the percentage of elites and non-elites that they evaluate at each iteration. Table 2 compares the reference algorithms.

Table 3 gives the parameter settings used by the proposed algorithm during the tests. Parameters which define the EA operation were identical in the proposed algorithm and in the four reference algorithms.

To obtain statistically-significant results 30 runs were repeated for each test with the proposed algorithm and the reference algorithms. For each function and each algorithm we provide the statistics mean, standard deviation, median, best and worst result. Also, to determine in a rigorous manner which algorithm performs better we used the Mann–Whitney (or Wilcoxon) significance test [64], which is a nonparametric version of the *t*-test [18, Ch.5], [104, p.513–576]. The Mann–Whitney test is preferable since it is more widely applicable: it is valid for non-normal data while applying the *t*-test on non-normal data can give incorrect inferences [18, Ch.2].

We used the one-tailed Mann–Whitney test which provides a test statistic U . The null and alternative hypothesis are:

$$H_0 : P(r_i \leqslant r_p) \geqslant 0.5 \tag{22a}$$

$$H_1 : P(r_i > r_p) < 0.5\,, \tag{22b}$$

where $P(r_i < r_p)$ is the probability that a result of the proposed algorithm is larger (worse) than a result of the ith reference algorithm ($i = 1 \ldots 4$). As such we test if the proposed algorithm is more likely to give a better result than the reference algorithms. The null hypothesis is rejected at the $\alpha = 0.05$ significance level if $U > 1.644$ and at the $\alpha = 0.01$ significance level if $U > 2.326$. For each test function we applied the Mann–Whitney test between results of the proposed algorithm and each of the four reference algorithms and provide the resultant U statistics. As such, for each reference algorithm if $U > 1.644$ or $U > 2.326$ we reject the null hypothesis at the 0.05 and 0.01 significance level, respectively, and accept the proposed algorithm outperformed the reference algorithm.

Table 2 Settings for the reference algorithms

			Evaluated per generation	
Number	Designation	SCF [1]	% elites	% non-elites
1	(G,10,0)	Gaussian	10	0
2	(S,10,0)	spline	10	0
3	(G,5,5)	Gaussian	5	5
4	(S,5,5)	spline	5	5

[1] spatial correlation function.

Table 3 Parameter settings for the proposed algorithm

	General Parameters	
$fe_{\max}$	max. (true) objective function evaluations	100 , 150 , 200[1]
	EA Search	
s_{pop}	population size	50
g_{gap}	generation gap	0.9
$g_{\max}$	maximum generations	20
p_{m}	mutation probability	0.05
$g_{\text{n.i.}}$	no-improvement generations to stop	10
	Trust-region Search	
Δ_0	initial trust-region radius	0.1
δ_+	trust-region size increase factor	2
δ_-	trust-region size decrease factor	0.5
$\Delta_{\min}$	minimum trust-region radius	$\Delta_0 \cdot \delta_-^2$
$\Delta_{\max}$	maximum trust-region radius	$\Delta_0 \cdot \delta_+^2$

[1] 100 and 200 for test functions, 150 for airfoil optimization.

4.2 Mathematical Test Functions

In this section we used eight mathematical test functions which are widely used and well-known: Branin, Hartman 3 and Hartman 6 [24], Greiwank [33], Rastrigin [117, p.185–192], Rosenbrock [97], Schwefel 2.13 [101, p.301–302] and Weierstrass [36]. For the Greiwank, Rastrigin, Rosenbrock, Schwefel and Weierstrass we used the function definitions from [111] along with the supporting files available online. The Branin, Hartman 3 and Hartman 6 have a fixed dimension (2, 3 and 6, respectively) while those from [111] were tested in dimension 10 and 30 to evaluate the 'curse of dimensionality' [5] on the algorithms performance. All functions were tested with a limit on function evaluations ($fe_{\max} = 100$ for Branin, Hartman 3 and Hartman 6 and $fe_{\max} = 200$ for all other functions). These are realistic settings for expensive problems and they test the algorithms under a constraint of resources, as

Table 4 Mathematical test functions

Function	Dimension (d)	Definition	Search space	$f(\boldsymbol{x}_\mathbf{g})$[1]
Reference: Dixon and Szegö [24]				
Branin	2	$(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$	$[-5, 10] \times [0, 15]$	0
Hartman 3	3	$\sum_{i=1}^{4} c_i \exp[\sum_{j=1}^{4} a_{i,j}(x_i - p_{i,j})^2]$	$[0, 1]^d$	−3.86
Hartman 6	6	$\sum_{i=1}^{4} c_i \exp[\sum_{j=1}^{6} a_{i,j}(x_i - p_{i,j})^2]$	$[0, 1]^d$	−3.32
Reference: Suganthan et. al. [111]				
Griewank	10, 30	$\sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^d$	0
Rastrigin	10, 30	$\sum_{i=1}^{d} \left\{x_i^2 - 10 \cdot \cos(2\pi x_i) + 10\right\}$	$[-5, 5]^d$	0
Rosenbrock	10, 30	$\sum_{i=1}^{d} \left\{(2x_{i-1} - x_i^2)^2 + (1 - x_i)^2\right\}$	$[-2, 2]^d$	0
Schwefel 2.13	10, 30	$\sum_{i=1}^{d} (\sum_{j=1}^{d} a_{i,j} \sin(\alpha_j) + b_{i,j} \cos(\alpha_j) - \sum_{j=1}^{d} a_{i,j} \sin(x_j) + b_{i,j} \cos(x_j))^2$	$[-\pi, \pi]^d$	0
Weierstrass	10, 30	$\sum_{i=1}^{d} \sum_{k=0}^{20} a^k \cos(2\pi b^k (x_i + 0.5))$	$[-0.5, 0.5]^d$	$-d$

[1] Value at global optimum.

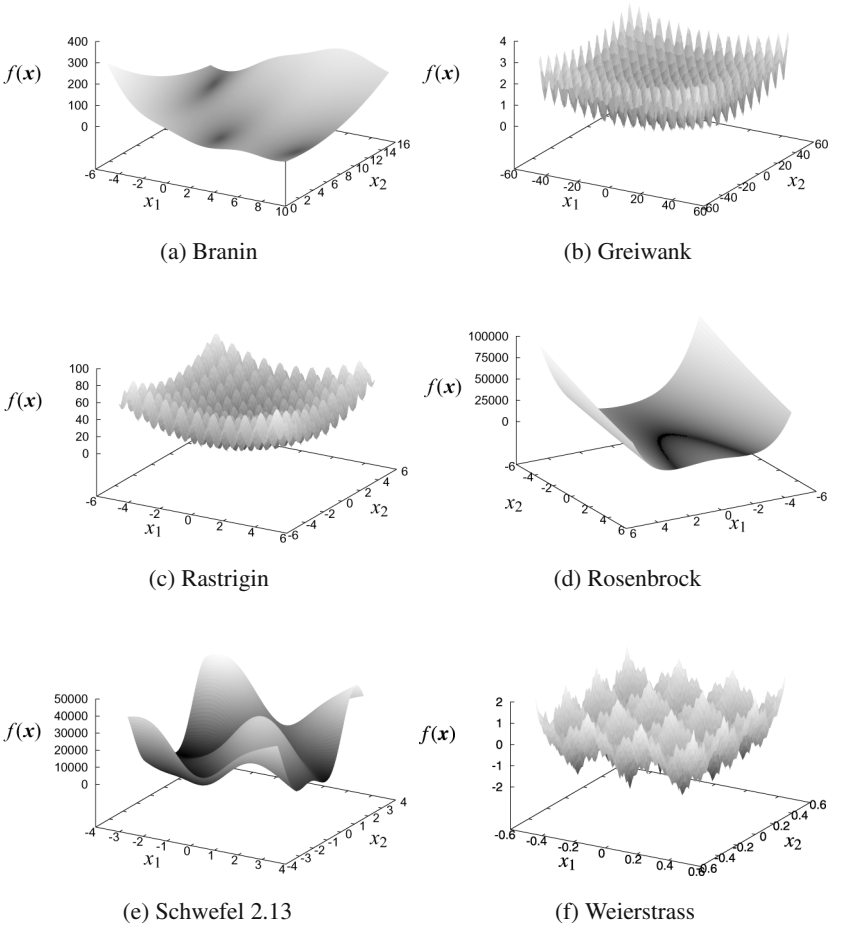
suggested in [118]. Table 4 gives the test functions' details and Fig. 5 shows their bivariate version (excluding Hartman 3 and Hartman 6 which are not bivariate).

Tables 5–7 provide the resultant test statistics for the comparisons with the four reference algorithms over the eight test functions. Results for the Branin, Hartman 3 and Hartman 6 for all algorithms are similar since they all obtained a good approximation of the global optimum. This indicates that these functions were not challenging to all five algorithms and there is no clear winner.

A significant difference in performance between the proposed algorithm and the reference algorithms is seen with the more complicated functions (Greiwank, Rastrigin, Rosenbrock, Schwefel and Weierstrass). The mean and median statistics indicate that the proposed algorithm found a better solution than the reference algorithms. This is attributed to the combined global and local search and the careful selection of models in these searches. Also, the standard deviation of results for the proposed algorithm is typically lower than that of the reference algorithms which indicates its performance is more stable. Overall, based on the Mann–Whitney test in all cases we reject the null hypothesis in (22) at both significance levels $\alpha = 0.05$ and 0.01 and accept that the proposed algorithm outperformed the four reference algorithms.

4.3 Individual Component Contribution

We also study the individual contribution of the global search and the local search to the overall performance of the proposed algorithm. For this, we used two reference algorithms obtained from the complete proposed algorithm. One algorithm



(a) Branin (b) Greiwank

(c) Rastrigin (d) Rosenbrock

(e) Schwefel 2.13 (f) Weierstrass

Fig. 5 Bivariate versions of the mathematical test functions

uses only the proposed global search (local search is disabled) and the other uses only the proposed local search (global search is disabled). The two reference algorithms were tested with five test functions: Greiwank and Rastrigin in dimension 10 and Rosenbrock, Schwefel and Weierstrass in dimension 30. A similar analysis to that of the Sect. 4.2 was used, that is we provide the statistics mean, standard deviation, median, best and worst and the Mann–Whitney U statistic.

Table 8 shows test results over the five test functions. First, for the highly multimodal functions (Greiwank, Rastrigin, Weierstrass) the global search reference algorithm performed better than the local search one. In such complicated landscapes an extensive global search finds a better optimum while a local search converges to an inferior optimum typically close to the starting site. An opposite trend exists for the simpler Rosenbrock and Schwefel functions where an extensive local search

Table 5 Results for mathematical tests functions

			Reference algorithms			
Function	Statistic [1,2]	Proposed	(G,10,0)	(S,10,0)	(G,5,5)	(S,5,5)
Branin	Mean	$3.979e-01$	$4.091e-01$	$4.767e-01$	$4.088e-01$	$4.566e-01$
	S.D.	$1.881e-06$	$5.158e-03$	$2.441e-01$	$1.754e-03$	$1.941e-01$
	Median	$3.979e-01$	$4.079e-01$	$4.015e-01$	$4.079e-01$	$3.982e-01$
	Best	$3.979e-01$	$4.079e-01$	$3.979e-01$	$4.079e-01$	$3.979e-01$
	Worst	$3.979e-01$	$4.362e-01$	$1.527e+00$	$4.139e-01$	$1.409e+00$
	U		$5.941e+00$	$5.426e+00$	$5.941e+00$	$4.376e+00$
Hartman 3	Mean	$-3.862e+00$	$-3.860e+00$	$-3.716e+00$	$-3.759e+00$	$-3.818e+00$
	S.D.	$3.510e-04$	$2.112e-03$	$2.242e-01$	$5.526e-03$	$5.532e-02$
	Median	$-3.863e+00$	$-3.861e+00$	$-3.806e+00$	$-3.763e+00$	$-3.841e+00$
	Best	$-3.863e+00$	$-3.862e+00$	$-3.863e+00$	$-3.763e+00$	$-3.863e+00$
	Worst	$-3.862e+00$	$-3.857e+00$	$-3.087e+00$	$-3.751e+00$	$-3.628e+00$
	U		$2.928e+00$	$4.082e+00$	$2.928e+00$	$3.581e+00$
Hartman 6	Mean	$-3.322e+00$	$-3.307e+00$	$-3.284e+00$	$-3.281e+00$	$-3.288e+00$
	S.D.	$5.245e-04$	$4.535e-02$	$5.982e-02$	$6.484e-02$	$7.002e-02$
	Median	$-3.322e+00$	$-3.321e+00$	$-3.321e+00$	$-3.321e+00$	$-3.321e+00$
	Best	$-3.322e+00$	$-3.321e+00$	$-3.321e+00$	$-3.321e+00$	$-3.321e+00$
	Worst	$-3.321e+00$	$-3.178e+00$	$-3.192e+00$	$-3.165e+00$	$-3.127e+00$
	U		$3.254e+00$	$2.712e+00$	$2.495e+00$	$2.712e+00$

[1] S.D. :standard deviation
[2] Reject H_0 at $\alpha = 0.05$ if $U \geqslant 1.644$.
Reject H_0 at $\alpha = 0.01$ if $U \geqslant 2.326$.

finds a better optimum. Second, the Mann–Whitney statistic indicates the full proposed algorithm outperformed the reference algorithms, which shows the benefit of the proposed global–local approach. Lastly, the standard deviation of results of the reference algorithms was typically much larger than for the proposed algorithm which indicates their performance is much less stable. Overall, results show that both the proposed global search and the proposed local search contribute to the optimization search. However, individually they perform well on some functions but worse on others. The proposed algorithm combines both approaches and so it achieves an effective and efficient search over a wide range of functions.

4.4 Real-World Application

As a final test we have also applied the proposed algorithm to the real-world application of airfoil shape optimization. Here we are given an aircraft's flight conditions (speed and altitude) and the goal is to find an airfoil shape which generates the required lift force (L) with a minimum of aerodynamic friction (or drag) force (D). In practice these requirements are not expressed as forces but as aerodynamic coefficients:

$$c_{\mathrm{L}} = \frac{L}{\frac{1}{2}\rho U^2} \quad \text{(lift coefficient)} \tag{23a}$$

$$c_{\mathrm{D}} = \frac{D}{\frac{1}{2}\rho U^2} \quad \text{(lift coefficient)} \tag{23b}$$

where ρ is the air density at the prescribed flight altitude and U is the prescribed flight speed. Figure 6 shows an example.

Table 6 Results for mathematical tests functions–10D

			Reference algorithms			
Function	Statistic [1,2]	Proposed	(G,10,0)	(S,10,0)	(G,5,5)	(S,5,5)
Greiwank	Mean	$1.001e+00$	$1.663e+00$	$1.125e+00$	$1.319e+00$	$1.137e+00$
	S.D.	$1.511e-01$	$9.735e-01$	$1.659e-01$	$4.055e-01$	$1.735e-01$
	Median	$9.955e-01$	$1.247e+00$	$1.101e+00$	$1.253e+00$	$1.113e+00$
	Best	$7.996e-01$	$9.890e-01$	$8.281e-01$	$8.418e-01$	$7.318e-01$
	Worst	$1.373e+00$	$4.466e+00$	$1.811e+00$	$2.873e+00$	$1.567e+00$
	U		$3.654e+00$	$2.967e+00$	$2.936e+00$	$2.905e+00$
Rastrigin	Mean	$4.089e+00$	$7.546e+01$	$2.703e+01$	$5.730e+01$	$2.243e+01$
	S.D.	$3.145e+00$	$2.249e+01$	$1.317e+01$	$2.634e+01$	$1.368e+01$
	Median	$3.980e+00$	$7.950e+01$	$2.686e+01$	$5.933e+01$	$1.940e+01$
	Best	$1.488e-06$	$1.393e+01$	$1.991e+00$	$1.375e+01$	$3.980e+00$
	Worst	$1.393e+01$	$1.238e+02$	$5.771e+01$	$1.120e+02$	$5.970e+01$
	U		$6.653e+00$	$6.209e+00$	$6.623e+00$	$6.165e+00$
Rosenbrock	Mean	$4.292e-01$	$9.697e-01$	$2.902e+00$	$2.078e+00$	$3.531e+00$
	S.D.	$7.117e-01$	$8.911e-01$	$1.284e+00$	$1.529e+00$	$1.546e+00$
	Median	$2.064e-01$	$7.126e-01$	$2.965e+00$	$1.707e+00$	$3.433e+00$
	Best	$3.928e-04$	$4.606e-03$	$6.685e-01$	$4.903e-02$	$2.855e-01$
	Worst	$3.734e+00$	$2.841e+00$	$6.009e+00$	$5.515e+00$	$6.729e+00$
	U		$2.587e+00$	$6.195e+00$	$4.923e+00$	$6.150e+00$
Schwefel 2.13	Mean	$1.082e+04$	$5.461e+04$	$2.776e+04$	$4.682e+04$	$4.102e+04$
	S.D.	$1.461e+04$	$7.234e+04$	$2.465e+04$	$4.876e+04$	$4.042e+04$
	Median	$5.082e+03$	$2.723e+04$	$1.722e+04$	$2.191e+04$	$2.977e+04$
	Best	$2.266e+01$	$4.433e+02$	$3.050e+03$	$2.723e+02$	$1.492e+03$
	Worst	$4.116e+04$	$3.470e+05$	$1.098e+05$	$1.674e+05$	$1.713e+05$
	U		$2.467e+00$	$2.567e+00$	$2.467e+00$	$2.867e+00$
Weierstrass	Mean	$-7.366e+00$	$-3.229e+00$	$-3.752e+00$	$-3.404e+00$	$-4.384e+00$
	S.D.	$1.011e+00$	$5.841e-01$	$7.929e-01$	$6.456e-01$	$1.073e+00$
	Median	$-7.408e+00$	$-3.144e+00$	$-3.821e+00$	$-3.422e+00$	$-4.450e+00$
	Best	$-9.119e+00$	$-4.835e+00$	$-5.208e+00$	$-4.498e+00$	$-7.164e+00$
	Worst	$-5.353e+00$	$-2.111e+00$	$-2.403e+00$	$-2.105e+00$	$-2.113e+00$
	U		$6.653e+00$	$6.653e+00$	$6.653e+00$	$6.357e+00$

[1] S.D. :standard deviation

[2] Reject H_0 at $\alpha = 0.05$ if $U \geqslant 1.644$.
Reject H_0 at $\alpha = 0.01$ if $U \geqslant 2.326$.

Table 7 Results for mathematical tests functions–30D

Function	Statistic [1,2]	Proposed	Reference algorithms			
			(G,10,0)	(S,10,0)	(G,5,5)	(S,5,5)
Greiwank	Mean	$1.003e+00$	$1.098e+00$	$1.060e+00$	$1.999e+00$	$1.063e+00$
	S.D.	$3.421e-02$	$3.175e-02$	$3.597e-02$	$6.687e-02$	$3.833e-02$
	Median	$1.022e+00$	$1.104e+00$	$1.056e+00$	$2.014e+00$	$1.059e+00$
	Best	$9.384e-01$	$9.784e-01$	$9.969e-01$	$1.748e+00$	$9.846e-01$
	Worst	$1.034e+00$	$1.139e+00$	$1.146e+00$	$2.077e+00$	$1.201e+00$
	U		$3.800e+00$	$3.257e+00$	$4.072e+00$	$3.568e+00$
Rastrigin	Mean	$7.734e+00$	$7.776e+01$	$7.483e+01$	$8.859e+01$	$7.369e+01$
	S.D.	$1.112e+01$	$2.514e+01$	$2.669e+01$	$2.739e+01$	$2.159e+01$
	Median	$2.145e+00$	$7.190e+01$	$6.730e+01$	$8.463e+01$	$7.408e+01$
	Best	$9.834e-03$	$4.659e+01$	$3.738e+01$	$5.143e+01$	$2.313e+01$
	Worst	$3.139e+01$	$1.361e+02$	$1.379e+02$	$1.570e+02$	$1.435e+02$
	U		$4.685e+00$	$4.685e+00$	$4.685e+00$	$4.654e+00$
Rosenbrock	Mean	$1.402e+01$	$2.106e+01$	$2.041e+01$	$2.095e+01$	$2.070e+01$
	S.D.	$1.761e+00$	$2.823e+00$	$2.722e+00$	$3.039e+00$	$2.486e+00$
	Median	$1.391e+01$	$2.085e+01$	$2.050e+01$	$2.116e+01$	$2.068e+01$
	Best	$9.863e+00$	$1.505e+01$	$1.418e+01$	$1.389e+01$	$1.751e+01$
	Worst	$1.907e+01$	$2.651e+01$	$2.657e+01$	$3.010e+01$	$2.537e+01$
	U		$6.368e+00$	$6.141e+00$	$6.277e+00$	$6.429e+00$
Schwefel 2.13	Mean	$1.933e+05$	$2.994e+05$	$3.086e+05$	$3.176e+05$	$3.157e+05$
	S.D.	$7.435e+04$	$1.168e+05$	$1.245e+05$	$1.173e+05$	$9.693e+04$
	Median	$1.875e+05$	$2.906e+05$	$2.886e+05$	$2.800e+05$	$3.307e+05$
	Best	$9.325e+04$	$1.079e+05$	$1.021e+05$	$1.540e+05$	$7.321e+04$
	Worst	$2.902e+05$	$6.348e+05$	$5.886e+05$	$5.843e+05$	$4.981e+05$
	U		$2.655e+00$	$2.780e+00$	$2.905e+00$	$3.342e+00$
Weierstrass	Mean	$-2.078e+01$	$-1.512e+01$	$-1.256e+01$	$-1.440e+01$	$-1.573e+01$
	S.D.	$8.117e-01$	$4.363e+00$	$3.090e+00$	$3.532e+00$	$3.591e+00$
	Median	$-2.092e+01$	$-1.434e+01$	$-1.171e+01$	$-1.545e+01$	$-1.490e+01$
	Best	$-2.189e+01$	$-2.577e+01$	$-2.078e+01$	$-2.143e+01$	$-2.372e+01$
	Worst	$-1.973e+01$	$-7.481e+00$	$-8.318e+00$	$-8.356e+00$	$-9.391e+00$
	U		$3.223e+00$	$4.189e+00$	$3.867e+00$	$3.258e+00$

[1] S.D. :standard deviation
[2] Reject H_0 at $\alpha = 0.05$ if $U \geqslant 1.644$.
Reject H_0 at $\alpha = 0.01$ if $U \geqslant 2.326$.

The specific problem we study is that of optimizing the airfoil of a transport aircraft cruising at 35,000 ft and at a Mach number M = 0.8 (that is 80% of the speed of sound at this altitude) with an angle of attack $\alpha = 2^\circ$. The target lift coefficient is $c_{\mathrm{L}}^\star = 1$. Also, the airfoil thickness must be equal to or larger than a minimum value ($t^\star = 0.095$, normalized by the airfoil chord) to ensure the airfoil does not break during flight. The cruise conditions and thickness constraint are based on [29, p.484–487], [85]. The airfoil optimization problem is then

$$
\begin{aligned}
&\min\ c_{\mathrm{D}}\ \text{(drag coefficient)}\\
&\text{s.t. } c_{\mathrm{L}}^{\star} = 1\ \text{(required lift coefficient)}\\
&\quad t^{\star} = 0.095\ \text{(min. allowed thickness at 0.2–0.8 of chord)}\\
&\quad \alpha = 2^{\circ}\ \text{(cruise angle of attack)}\\
&\quad \mathrm{M} = 0.8\ \text{(cruise Mach number)}\\
&\quad \mathrm{h} = 35{,}000\,\mathrm{ft}\ \text{(cruise altitude)}
\end{aligned}
\tag{24}
$$

Table 8 Results for mathematical tests functions–Individual component contribution

Function	Statistic [1,2]	Proposed	Reference algorithms	
			Global Search	Local Search
Greiwank–10D	Mean	1.001e+00	1.765e+05	1.771e+08
	S.D.	1.511e−01	8.430e+04	3.753e+07
	Median	9.955e−01	1.572e+05	1.717e+08
	Best	7.996e−01	7.172e+04	1.262e+08
	Worst	1.373e+00	3.491e+05	2.398e+08
	U		3.780e+00	3.780e+00
Rastrigin–10D	Mean	4.089e+00	6.543e+00	5.199e+01
	S.D.	3.145e+00	2.567e+00	1.290e+01
	Median	3.980e+00	5.757e+00	5.077e+01
	Best	1.488e−06	3.642e+00	2.723e+01
	Worst	1.393e+01	1.083e+01	6.866e+01
	U		2.218e+00	4.685e+00
Rosenbrock–30D	Mean	1.402e+01	2.674e+01	1.993e+01
	S.D.	1.761e+00	2.013e+00	4.236e+00
	Median	1.391e+01	2.681e+01	2.031e+01
	Best	9.863e+00	2.269e+01	1.339e+01
	Worst	1.907e+01	2.931e+01	2.791e+01
	U		4.664e+00	3.924e+00
Schwefel 2.13–30D	Mean	1.933e+05	1.294e+06	3.453e+05
	S.D.	7.435e+04	2.314e+05	1.756e+05
	Median	1.875e+05	1.303e+06	3.131e+05
	Best	9.325e+04	7.783e+05	1.415e+05
	Worst	2.902e+05	1.569e+06	7.355e+05
	U		3.780e+00	2.419e+00
Weierstrass–30D	Mean	−2.078e+01	−1.857e+01	−1.459e+01
	S.D.	8.117e−01	1.421e+00	2.563e+00
	Median	−2.092e+01	−1.808e+01	−1.440e+01
	Best	−2.189e+01	−2.153e+01	−1.910e+01
	Worst	−1.973e+01	−1.721e+01	−1.024e+01
	U		2.843e+00	3.554e+00

[1] S.D. :standard deviation

[2] Reject H_0 at $\alpha = 0.05$ if $U \geqslant 1.644$.
Reject H_0 at $\alpha = 0.01$ if $U \geqslant 2.326$.

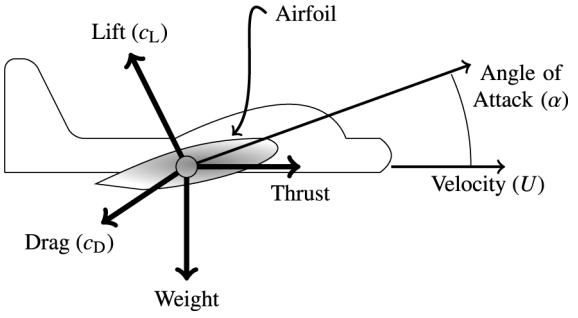


Fig. 6 The forces operating on an aircraft at flight. The angle-of-attack (α) is the measured approximately between the velocity and the airfoil chord.

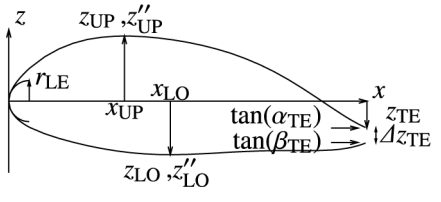


Fig. 7 PARSEC design variables

Accordingly, we used the objective function

$$f = \frac{|c_L - c_L^\star|}{\max\{c_{L,max} - c_L^\star, c_L^\star - c_{L,min}\}} + \frac{c_D}{c_{D,max}} + \frac{\max\{t^\star - t, 0\}}{t^\star} \tag{25}$$

where $c_{L,max} = 1.5$, $c_{L,min} = -0.5$ are the assumed extremal values for the lift coefficient and $c_{D,max} = 0.2$ is an assumed maximal drag coefficient. For $c_{L,min}$, $c_{L,max}$ and $c_{D,max}$ only rough estimates are needed since they only normalize the objectives.

To generate a candidate airfoil we used the PARSEC parameterization which uses 11 design variables [108]. Figure 7 shows an example of this. We set the bounds of these design variables based on previous studies [41, 86] and Table 9 gives their values. To ensure a closed airfoil shape we set the leading edge gap as $\Delta z_{TE} = 0$. Also, to avoid unrealistic shapes where the lower airfoil curve intersects the upper curve we set the trailing edge angles to satisfy $\beta_{TE} \geqslant \alpha_{TE}$ (effectively $\beta_{TE} = \alpha_{TE} + \theta$ where $\theta \geqslant 0$).

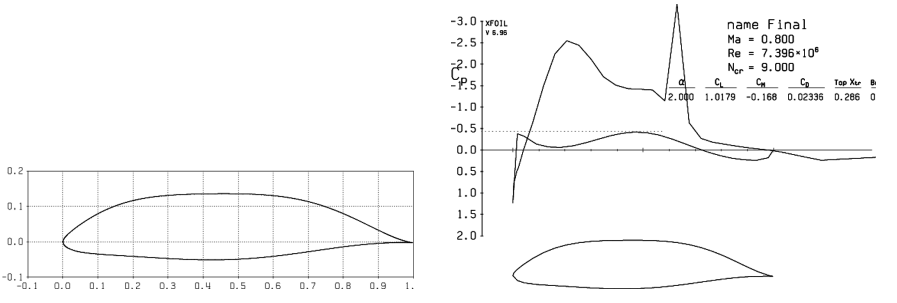
To obtain the lift coefficient and drag coefficient of candidate airfoils the optimization algorithm used XFoil, an analysis code for subsonic isolated airfoils based on the panel method [26]. Each airfoil evaluation required approximately 30 seconds on a desktop computer. We set the limit of function evaluations to $fe_{max} = 150$.

Figure 8 shows an airfoil found by the proposed algorithm and the variation of the pressure coefficient (c_P) along the upper and lower airfoil curves. The airfoil yields a lift coefficient of $c_L = 1.019$ and a drag coefficient $c_D = 0.023$ and satisfies

Table 9 PARSEC design variables and their bounds

Variable	Meaning	min.	max.
r_{LE}	leading-edge radius	0.002	0.030
x_{up}	max. upper thickness location	0.2	0.7
z_{up}	max. upper thickness	0.08	0.18
z''_{up}	max. upper curvature	−0.6	0.0
x_{lo}	max. lower thickness location	0.2	0.6
z_{lo}	max. upper thickness	−0.09	0.02
z''_{lo}	max. lower curvature	0.2	0.9
z_{TE}	trailing edge height	−0.01	0.01
Δz_{TE}	trailing edge thickness	0	0
α_{TE} [1]	upper trailing edge angle°	165	180
β_{TE} [1,2]	lower trailing edge angle°	165	190

[1] measured anti-clockwise from the x-axis.
[2] $\beta_{TE} \geqslant \alpha_{TE}$ to avoid intersecting curves.



(a) Airfoil geometry (b) Pressure coefficient change

Fig. 8 An airfoil obtained by the proposed algorithm. (a) shows the airfoil geometry. (b) shows the change of pressure coefficient along the upper and lower airfoil curves and the airfoil is shown below for reference

the minimum thickness requirement (minimum thickness at 0.2–0.8 of chord is t = 0.097). Figure 8(b) shows the pressure coefficient change along the upper and lower airfoil curves. A pressure jump on the upper curve around 0.7 of the chord indicates a shockwave, which is expected due to the high subsonic cruise speed (M = 0.8).

We have also benchmarked the proposed algorithm against the four reference algorithms from the Sect. 4.1 and have performed a statistical analysis as in Sects. 4.2–4.3. Table 10 shows the test statistics from which it follows that also in this real-world application the proposed algorithm outperformed the four reference algorithms.

Table 10 Results for the airfoil shape optimization

Function	Statistic [1,2]	Proposed	(G,10,0)	(S,10,0)	(G,5,5)	(S,5,5)
Airfoil	Mean	$5.674e-01$	$7.336e-01$	$7.757e-01$	$6.934e-01$	$7.571e-01$
	S.D.	$8.098e-02$	$1.336e-01$	$2.583e-01$	$8.974e-02$	$1.701e-01$
	Median	$6.029e-01$	$6.920e-01$	$7.460e-01$	$7.092e-01$	$7.666e-01$
	Best	$3.846e-01$	$4.763e-01$	$4.287e-01$	$5.009e-01$	$4.369e-01$
	Worst	$6.326e-01$	$9.159e-01$	$1.430e+00$	$8.003e-01$	$9.738e-01$
	U		$3.021e+00$	$2.858e+00$	$2.939e+00$	$2.613e+00$

[1] S.D. :standard deviation
[2] Reject H_0 at $\alpha = 0.05$ if $U \geqslant 1.644$.
Reject H_0 at $\alpha = 0.01$ if $U \geqslant 2.326$.

5 Summary

Modern engineering design optimization uses computer simulations and as such it is cast as a problem of optimizing an expensive black-box function. To efficiently and effectively solve such problems we have proposed a new model-assisted memetic algorithm. The proposed algorithm combines several optimization approaches such as: global–local optimization, modelling and memetic optimization. It first trains and selects a global model which is an artificial neural network and seeks the optimum of this model. It then uses a local search to improve this predicted optimum. This sequence is repeated until the number of function evaluations reaches the prescribed limit. Compared to existing studies the proposed algorithm contains three main novelties: a) it selects all models under a unified and statistically-sound framework of model selection and complexity control, and this gives optimal models which are adapted during the search b) it uses an improved trust-region framework to converge to a true optimum while replacing the classical quadratic models with Kriging models and adapting these models during the search and c) it improves global exploration by training the global model with modified sites.

An extensive performance analysis has been provided. Results show the proposed algorithm outperformed four model-assisted EAs on eight well-known mathematical test functions. The individual contribution of the proposed global search and local search component was also studied. While each component performs well on a certain class of problems it also performs poorly on another. This emphasizes the advantage of the global–local approach used. Lastly, the proposed algorithm was also applied to a real-world application of airfoil shape optimization where it also performed better than the reference algorithms.

References

1. Akaike, H.: Information theory and an extension of the maximum likelihood principle. In: Proceedings of the 2*nd* International Symposium on Information Theory, Akadémiai Kiadó, Budapest, pp. 267–281 (1973)

2. Alexandrov, N.M., Lewis, R.M., Gumbert, C.R., Green, L.L., Newma, P.A.: Optimization with variable-fidelity models applied to wing design. In: Proceedings of the 38*th* Aerospace Sciences Meeting and Exhibit, American Institute for Aeronautics and Astronautics, Reston, Virginia (2000)
3. Anderson, D.R., Burnham, K.P., White, G.C.: Comparison of Akaike information criterion and consistent Akaike information criterion for model selection and statistical inference from capture-recapture studies. Journal of Applied Statistics 25(2), 263–282 (1998)
4. Barthelemy, J.F.M., Haftka, R.T.: Approximation concepts for optimum structural design – a review. Structural optimization 5, 129–144 (1993)
5. Bellman, R.E.: Adaptive Control Processes: A Guided Tour. Princeton University Press, Princeton (1961)
6. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, New York (1995)
7. Booker, A.J., Dennis, J.E., Frank, P.D., Serafini, D.B., Torczon, V., Trosset, M.W.: A rigorous framework for optimization of expensive functions by surrogates. Structural Optimization 17(1), 1–13 (1999)
8. Box, G.E.P., Draper, N.R.: Empirical Model Building and Response Surface. John Wiley and Sons, New York (1987)
9. Brent, R.P.: Algorithms for Minimization Without Derivatives, 3rd edn. Dover Publications, New York (2002)
10. Broomhead, D., Lowe, D.: Multivariate functional interpolations and adaptive networks. Complex Systems 2, 321–355 (1988)
11. Buhman, M.D.: Radial Basis Functions Theory and Implementations. Cambridge Monographs on Applied and Computational Mathematics, vol. 12. Cambridge University Press, Cambridge (2003)
12. Burnham, K.P., Anderson, D.R.: Model selection and inference: A Practical Information-theoretic Approach. Springer, New York (1998)
13. Chipperfield, A., Fleming, P., Pohlheim, H., Fonseca, C.: Genetic Algorithm TOOLBOX For Use with MATLAB, Version 1.2. Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield (1994)
14. Conn, A.R., Scheinberg, K., Toint, P.L.: On the convergence of derivative-free methods for unconstrained optimization. In: Iserles, A., Buhmann, M.D. (eds.) Approximation Theory and Optimization: Tributes to M.J.D. Powell, pp. 83–108. Cambridge University Press, Cambridge (1997)
15. Conn, A.R., Scheinberg, K., Toint, P.L.: Recent progress in unconstrained nonlinear optimization without derivatives. Mathematical Programming 79, 397–414 (1997)
16. Conn, A.R., Scheinberg, K., Toint, P.L.: A derivative free optimization algorithm in practice. In: Proceedings of the Seventh AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, American Institute for Aeronautics and Astronautics, Reston, Virginia, AIAA Paper AIAA-1998-4718 (1998)
17. Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust Region Methods. SIAM, Philadelphia (2000)
18. Conover, W.: Practical Nonparametric Statistics, 2nd edn. John Wiley and Sons, New York (1980)
19. Cressie, N.A.C.: The origins of Kriging. Mathematical Geology 22(3), 239–252 (1990)
20. Cressie, N.A.C.: Statistics for Spatial Data. Wiley, New York (1993)
21. Dawkins, R.: The Selfish Gene. Oxford University Press, Oxford (1976)

22. Demmel, J.W.: The geometry of ill-conditioning. Computer Journal 3(2), 201–229 (1987)
23. Dennis, J.E., Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Classics in Applied Mathematics. SIAM Publishing, Philadelphia (1996)
24. Dixon, L.C.W., Szegö, G.P.: The global optimization problem: An introduction. In: [25], pp. 1–15 (1978)
25. Dixon, L.C.W., Szegö, G.P. (eds.): Towards Global Optimisation 2. North-Holland Publishing Company, Amsterdam (1978)
26. Drela, M., Youngren, H.: XFOIL 6.9 User Primer. Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA (2001)
27. Eby, D., Averill, R.C., Punch III, W.F., Goodman, E.D.: Evaluation of injection island GA performance on flywheel design optimization. In: Proceedings of the Third Conference on Adaptive Computing in Design and Manufacturing–ACDM 1998, pp. 121–136. Springer, London (1998)
28. Fang, K.T., Li, R., Sudjinato, A.: Design and Modeling for Computer Experiments. Chapman and Hall, Boca Raton (2006)
29. Filippone, A.: Flight Performance of Fixed and Rotary Wing Aircraft, 1st edn. Elsevier, Amsterdam (2006)
30. Gaspar-Cunha, A., Vieira, A.: A multi-objective evolutionary algorithm using neural networks to approximate fitness evaluations. International Journal of Computers, Systems and Signals 6(1), 18–36 (2005)
31. Giannakoglou, K.C.: Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. International Review Journal Progress in Aerospace Sciences 38(1), 43–76 (2002)
32. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
33. Griewank, A.O.: Generalized descent for global optimization. Journal of Optimization Theory and Applications 34, 11–39 (1981)
34. Groch, A., Vidigal, L.M., Director, S.W.: A new global optimization method for electronic circuit design. IEEE Transactions on Circuit and Systems 32(2), 160–170 (1985)
35. Hamerly, G., Elkan, C.: Alternatives to the *k*-means algorithm that find better clusterings. In: Munson, E.V., Furuta, R., Maletic, J.I. (eds.) Proceedings of the 2002 ACM Symposium on Document Engineering, in conjunction with the eleventh ACM International Conference on Information and Knowledge Management–CIKM 2002, New York, pp. 600–607 (2002)
36. Hardy, G.H.: Weierstrass's non-differentiable function. Transactions of the American Mathematical Society 17, 301–325 (1916)
37. Hart, W.E., Belew, R.K.: Optimization with genetic algorithm hybrids that use local search. In: Belew, R.K., Mitchell, M. (eds.) Adaptive Individuals in Evolving Populations: Models and Algorithms, Santa Fe Institute Studies in the Sciences of Complexity, ch. 27, pp. 483–496. Addison-Wesley, Reading (1995)
38. Hart, W.E., Krasnogor, N., Smith, J.E.: Special issue on memetic algorithms. Evolutionary Computation 12(3) (2004)
39. Hart, W.E., Krasnogor, N., Smith, J.E.: Recent Advances in Memetic Algorithms. Studies in Fuzziness and Soft Computing, vol. 166. Springer, Heidelberg (2005)
40. Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Prentice-Hall, Upper Saddle River (1999)

41. Holst, T.L., Pulliam, T.H.: Aerodynamic shape optimization using a real-number-encoded genetic algorithm. In: Proceedings of the 19th AIAA Applied Aerodynamics Conference, American Institute for Aeronautics and Astronautics, Reston, Virginia, AIAA Paper AIAA-2001-2473 (2001)
42. Hurvich, C.M., Tsai, C.L.: Regression and time series model selection in small samples. Biometrika 76(2), 297–307 (1989)
43. Ingber, L.A., Rosen, B.: Genetic algorithms and very fast simulated reannealing: A comparison. Mathematical and Computer Modelling 16(11), 87–100 (1992)
44. Ishibuchi, H., Yoshida, T., Murata, T.: Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Transactions on Evolutionary Computation 7, 204–223 (2003)
45. Jin, R., Chen, W., Simpson, T.W.: Comparative studies of metamodeling techniques under multiple modeling criteria. Structural Optimization 23(1), 1–13 (2001)
46. Jin, Y., Olhofer, M., Sendhoff, B.: A framework for evolutionary optimization with approximate fitness functions. IEEE Transactions on evolutionary computation 6(5), 481–494 (2002)
47. Johnson, M.E., Moore, L.M., Ylvisaker, D.: Minimax and maximin distance designs. Journal of Statistical Planning and Inference 26(2), 131–148 (1990)
48. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization 13, 455–492 (1998)
49. de Jong, K.A.: Genetic algorithms are NOT function optimizers. In: Whitley, D.L. (ed.) Foundations of Genetic Algorithms 2, pp. 5–17. Morgan Kaufmann, San Mateo (1993)
50. de Jong, K.A., Spears, W.M.: An analysis of the interacting roles of population size and crossover in genetic algorithms. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 38–47. Springer, Heidelberg (1991)
51. Kansa, E.J., Hon, Y.C.: Circumventing the ill-conditioning problem with multiquadric radial basis functions: Applications to elliptic partial differential equations. Computers and Mathematics with Applications 39(7), 123–137 (2000)
52. Kim, H.S., Cho, S.B.: An efficient genetic algorithm with less fitness evaluation by clustering. In: Proceedings of 2001 IEEE Conference on Evolutionary Computation, pp. 887–894. IEEE, Piscataway (2001)
53. Koehler, J.R., Owen, A.B.: Computer experiments. In: Ghosh, S., Rao, C.R., Krishnaiah, P.R. (eds.) Handbook of Statistics, pp. 261–308. Elsevier, Amsterdam (1996)
54. Krasnogor, N., Smith, J.E.: A tutorial for competent memetic algorithms: Model, taxonomy, and design issues. IEEE Transactions on Evolutionary Computation 9(5), 474–488 (2005)
55. Ku, K., Mak, M., Siu, W.: A study of the Lamarckian evolution of recurrent neural networks. IEEE Transactions on Evolutionary Computation 4, 31–42 (2000)
56. Kullback, S., Leibler, R.A.: On information and sufficiency. The Annals of Mathematical Statistics 22(1), 79–86 (1951)
57. Kushner, H.J.: A versatile stochastic model of a function of unknown and time varying form. Journal of Mathematical Analysis and Applications 5(1), 150–167 (1962)
58. Laslett, G.M.: Kriging and splines: An empirical comparison of their predictive performance in some applications. Journal of the American Statistical Association 89(426), 391–400 (1994)
59. Leontaritis, I.J., Billings, S.A.: Model selection and validation for non-linear systems. International Journal of Control 45(1), 311–341 (1986)

60. Liang, K.H., Yao, X., Newton, C.: Evolutionary search of approximated N-dimensional landscapes. International Journal of Knowledge-Based Intelligent Engineering Systems 4(3), 172–183 (2000)
61. Linhart, H., Zucchini, W.: Model Selection. Wiley Series in Probability and Mathematical Statistics. Wiley-Interscience Publication, New York (1986)
62. Lorentz, G.G.: Approximation of Functions. Rinehart and Winston, New York (1966)
63. Madych, W.R.: Miscellaneous error bounds for multiquadric and related interpolators. Computers and Mathematics with Applications 24(12), 121–138 (1992)
64. Mann, H.B., Whitney, D.R.: On a test whether one of two variables is stochastically larger than the other. The Annals of Mathematical Statistics 18, 50–60 (1947)
65. Marida, K., Marshall, R.: Maximum likelihood estimation of models for residual covariance in spatial regression. Biometrika 71(1), 135–146 (1984)
66. Matheron, C.: Principles of geostatistics. Economic Geology 58, 1246–1266 (1963)
67. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21(2), 239–245 (1979)
68. Meckesheimer, M., Booker, A.J., Barton, R.R., Simpson, T.W.: Computationally inexpensive metamodel assessment strategies. AIAA Journal 40(10), 2053–2060 (2002)
69. Medgyessy, P.: Decomposition of Superpositions of Distribution Functions. Akadémiai Kiadó, Budapest (1961)
70. Michalewicz, Z., Fogel, D.B.: How to Solve It: Modern Heuristics. Springer, Berlin (2004)
71. Mitchell, T.J., Morris, M.D.: Bayesian design and analysis of computer experiments: Two examples. Statistica Sinica 2 (1992)
72. Mockus, J., Vitešis, V., Žilinskas, A.: The application of Bayesian methods for seeking the extremum. In: [25], pp. 117–130 (1978)
73. Moody, J., Darken, C.J.: Fast learning in networks of locally-tuned processing units. Neural Computation 1(2), 281–294 (1989)
74. Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: Toward memetic algorithms. Tech. Rep. 826, California Institute of Technology, Pasadena, California (1989)
75. Mühlenbein, H., Schlierkamp-Voosen, D.: Predictive models for the breeder genetic algorithm I: Continuous parameter optimization. Evolutionary Computations 1(1), 25–49 (1993)
76. Myers, R.H., Montgomery, D.C.: Response Surface Methodology: Process and Product Optimization Using Designed Experiments. John Wiley and Sons, New York (1995)
77. Norman, M., Moscato, P.: A competitive-cooperative approach to complex combinatorial search. Tech. Rep. 790, California Institute of Technology, Pasadena, California (1989)
78. Ong, Y.S., Keane, A.J.: Meta-Lamarckian learning in memetic algorithm. IEEE Transactions On Evolutionary Computation 8(2), 99–110 (2004)
79. Ong, Y.S., Nair, P.B., Keane, A.J.: Evolutionary optimization of computationally expensive problems via surrogate modeling. AIAA Journal 41(4), 687–696 (2003)
80. Ong, Y.S., Nair, P.B., Keane, A.J., Wong, K.W.: Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In: Knowledge Incorporation in Evolutionary Computation. Studies in Fuzziness and Soft Computing, vol. 167, pp. 307–332. Springer, Berlin (2005)

81. Ong, Y.S., Lim, M.H., Zhu, N., Wong, K.W.: Classification of adaptive memetic algorithms: A comparative study. IEEE Transactions on Systems, Man, and Cybernetics–Part B 36(1), 141–152 (2006)
82. Ong, Y.S., Krasnogor, N., Ishibuchi, H.: Special issue on memetic algorithms. IEEE Transactions on Evolutionary Computation 37(1) (2007)
83. Ong, Y.S., Lim, M.H., Neri, F., Ishibuchi, H.: Special issue on emerging trends in soft computing–memetic algorithms. Journal of Soft Computing (to appear)
84. Owen, A.B.: Orthogonal arrays for computer experiments, integration and visualization. Statistica Sinica 2, 439–452 (1992)
85. Oyama, A., Obayashi, S., Nakahashi, K.: Real-coded adaptive range genetic algorithm and its application to aerodynamic design. International Journal of the Japan Society of Mechanical Engineering 43(2), 124–129 (2000)
86. Oyama, A., Obayashi, S., Nakahashi, T.: Real-coded adaptive range genetic algorithm applied to transonic wing optimization. In: Schoenauer, M. (ed.) The 6*th* International Conference on Parallel Problem Solving from Nature–PPSN VI, pp. 712–721. Springer, Heidelberg (2000)
87. Pawitan, Y.: In All Likelihood: Statistical Modelling and Inference Using Likelihood. Oxford Scientific Publishing, Oxford (2001)
88. Poloni, C., Giurgevich, A., Onseti, L., Pediroda, V.: Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. Computer Methods in Applied Mechanics and Engineering 186(2-4), 403–420 (2000)
89. Powell, M.J.D.: UOBYQA: Unconstrained optimization by quadratic approximation. Mathematical Programming, Series B 92, 555–582 (2002)
90. Quagliarella, D., Vicini, A.: Coupling genetic algorithms and gradient based optimization techniques. In: Quagliarella, D., Périaux, J., Poloni, C., Winter, G. (eds.) Genetic Algorithms in Engineering and Computer Science, ch. 14, pp. 288–309. John Wiley and Sons, Chichester (1997)
91. Ratle, A.: Accelerating the convergence of evolutionary algorithms by fitness landscape approximations. In: Eiben, A.E., Bäck, T., Schwefel, H.P. (eds.) Proceedings of the 5*th* International Conference on Parallel Problem Solving from Nature–PPSN V, pp. 87–96. Springer, Berlin (1998)
92. Ratle, A.: Optimal sampling strategies for learning a fitness model. In: The 1999 IEEE Congress on Evolutionary Computation–CEC 1999, pp. 2078–2085. IEEE, Piscataway (1999)
93. Renderes, J.M., Bersini, H.: Hybridizing genetic algorithms with hill-climbing methods for global optimization: Two possible ways. In: Sebald, A., Fogel, L.J. (eds.) Proceedings of the Third Annual Conference on Evolutionary Programming, pp. 312–317. World Scientific, Singapore (1994)
94. Renderes, J.M., Flasse, S.P.: Hybrid methods using genetic algorithms for global optimization. IEEE Transactions on Systems, Man, and Cybernetics–Part B 26(2), 243–258 (1996)
95. Reyes-Sierra, M., Coelle Coello, C.A.: Dynamic fitness inheritance proportion for multi-objective particle swarm optimization. In: Keijzer, M. (ed.) Proceedings of the Genetic and Evolutionary Computation Conference–GECCO 2006, pp. 89–90. Association for Computing Machinery, New York (2006)
96. Rinnooy Kan, A.H.G., Timmer, G.T.: Stochastic global optimization methods part I: Clustering methods. Mathematical Programming 39, 27–56 (1987)

97. Rosenbrock, H.H.: An automated method for finding the greatest of least value of a function. The Computer Journal 3, 175–184 (1960)
98. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Statistical Science 4(4), 409–435 (1989)
99. Santner, T.J., Williams, B.J., Notz, W.: The Design and Analysis of Computer Experiments. Springer Series in Statistics. Springer, New York (2003)
100. Schaback, R.: Multivariate interpolation and approximation by translates of a basis function. In: Chui, C.K., Schumaker, L.L. (eds.) Approximation Theory VIII, pp. 491–514. World Scientific, Singapore (1995)
101. Schwefel, H.P.: Numerical Optimization of Computer Models, Interdisciplinary Systems Research, vol. 26. John Wiley and Sons, Chichester (1981)
102. Sefrioui, M., Périaux, J.: Aerodynamic shape optimization using a hierarchical genetic algorithm. In: European Conference on Computational Methods in Applied Sciences and Engineering–ECCOMAS 2000, European Committee on Computational Methods in Applied Sciences, pp. 1–18 (2000)
103. Seront, G., Bersini, H.: A new GA-local search hybrid for continuous optimization based on multi level single linkage clustering. In: Whitley, D.L., Beyer, H., Cantu-Paz, E., Goldberg, D.E., Parmee, I., Spector, L. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference–GECCO 2000, pp. 90–95. Morgan Kaufmann, San Francisco (2000)
104. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures, 4th edn. Chapman and Hall, Boca Raton (2007)
105. Simkin, J., Trowbridge, C.W.: Optimizing electromagnetic devices combining direct search methods with simulated annealing. IEEE Transactions on Magnetics 28(2), 1545–1548 (1992)
106. Simpson, T.W., Poplinski, J.D., Koch, P.N., Allen, J.K.: Metamodels for computer-based engineering design: Survey and recommendations. Engineering with Computers 17, 129–150 (2001)
107. Smith, R.E., Dike, B., Stegmann, S.: Fitness inheritance in genetic algorithms. In: George, K.M. (ed.) Proceedings of the 1995 ACM Symposium on Applied Computing–ACM 1995, pp. 345–350. Association for Computing Machinery, New York (1995)
108. Sobieckzy, H.: Parametric airfoils and wings. In: Fujii, K., Dulikravich, G.S., Takanashi, S. (eds.) Recent Development of Aerodynamic Design Methodologies: Inverse Design and Optimization, Notes on Numerical Fluid Mechanics, vol. 68, pp. 71–88. Vieweg, Braunschweig (1999)
109. Søren, L.N., Nielsen, H.B., Søndergaard, J.: DACE: A MATLAB Kriging toolbox. Technical Report IMM-TR-2002-12, Informatik and Mathematical Modelling, Technical University of Denmark, Lingby, Copenhagen (2002)
110. Stuckman, B.E.: A global search method for optimizing nonlinear systems. IEEE Transactions on Systems, Man, and Cybernetics 18(6), 965–977 (1988)
111. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical Report KanGAL 2005005, Nanyang Technological University, Singapore and Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology Kanpur, India (2005), http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/SharedDocuments/Forms/AllItems.aspx
112. Tenne, Y.: Metamodel accuracy assessment in evolutionary optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation–CEC 2008. IEEE World Congress on Computational Intelligence, pp. 1505–1512. IEEE, Piscataway (2008)

113. Tenne, Y., Armfield, S.W.: A memetic algorithm using a trust-region derivative-free optimization with quadratic modelling for optimization of expensive and noisy black-box functions. In: Yang, S., Ong, Y.S., Jin, Y. (eds.) Evolutionary Computation in Dynamic and Uncertain Environments. Studies in Computational Intelligence, vol. 51, pp. 389–415. Springer, Berlin (2007)
114. Tenne, Y., Armfield, S.W.: A versatile surrogate-assisted memetic algorithm for optimization of computationally expensive functions and its engineering applications. In: Yang, A., Shan, Y., Thu Bui, L. (eds.) Success in Evolutionary Computation. Studies in Computational Intelligence, vol. 92, pp. 43–72. Springer, Heidelberg (2008)
115. Torczon, V.: On the convergence of pattern search algorithms. SIAM Journal on Optimization 7(1), 1–25 (1997)
116. Törn, A.: Global optimization as a combination of global and local search. In: Proceedings of Computer Simulation Versus Analytical Solutions for Business and Economic Models, School of Business Administration, Göteborg, Göteborg, Sweden. Business Administration Studies–BAS, vol. 17, pp. 191–206 (1973)
117. Törn, A., Žilinskas, A.: Global Optimization. LNCS, vol. 350. Springer, Heidelberg (1989)
118. Törn, A., Ali, M.M., Viitanen, S.: Stochastic global optimization: Problems classes and solution techniques. Journal of Global Optimization 14, 437–447 (1999)
119. Žilinskas, A.: A review of statistical models for global optimization. Journal of Global Optimization 2, 145–153 (1992)
120. Waldorp, L.J., Raoul, P.P.P., Huizenga, H.M.: Goodness-of-fit and confidence intervals of approximate models. Journal of Mathematical Psychology 50, 203–213 (2006)
121. Winfield, D.: Function minimization by interpolation in a data table. Journal of the Institute of Mathematics and its Applications 12, 339–347 (1973)
122. Yen, J., Liao, J.C., Lee, B., Randolph, D.: A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method. IEEE Transactions on Systems, Man, and Cybernetics–Part B 28(2), 173–191 (1998)
123. Yun, Y., Gen, M., Seo, S.: Various hybrid methods based on genetic algorithm with fuzzy logic controller. Journal of Intelligent Manufacturing 14, 401–419 (2003)
124. Zhang, B.: Generalized K-harmonic means–dynamic weighting of data in unsupervised learning. Tech. Rep. HPL-2000-137, Hewlett-Packard Labs (2000)
125. Zhang, B., Hsu, M., Dayal, U.: K-harmonic means–a data clustering algorithm. Tech. Rep. HPL-1999-124, Hewlett-Packard Labs, Software Technology Laboratory, HP Laboratories Palo Alto (1999)
126. Zhou, Z., Ong, Y.S., Lim, M.H., Lee, B.: Memetic algorithms using multi-surrogates for computationally expensive optimization problems. Journal of Soft Computing 11(10), 957–971 (2007)
127. Zhou, Z., Ong, Y.S., Nair, P.B., Keane, A.J., Lum, K.Y.: Combining global and local surrogate models to accelerate evolutionary optimization. IEEE Transactions On Systems, Man and Cybernetics-Part C 37(1), 66–76 (2007)