INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2003; **43**:1149–1166 (DOI: 10.1002/fld.575)

Inexact information aided, low-cost, distributed genetic algorithms for aerodynamic shape optimization

Marios K. Karakasis, Alexios P. Giotis and Kyriakos C. Giannakoglou*,†

National Technical University of Athens, Lab. of Thermal Turbomachines, P.O. Box 64069, Athens 157 10, Greece

SUMMARY

Despite its robustness, the design and optimization of aerodynamic shapes using genetic algorithms suffers from high computing cost requirements, due to excessive calls to Computational Fluid Dynamics tools for the evaluation of candidate solutions. To alleviate this problem, either the use of distributed genetic algorithms or the implementation of surrogate evaluation models have separately been proposed in the past. A distributed genetic algorithm relies on the handling of population subsets that evolve in a semi-isolated manner by regularly exchanging their best individuals. It is known that distributed schemes generally outperform single-population ones. On the other hand, the implementation of less costly surrogate evaluation tools, such as the autocatalytic radial basis function networks developed by the authors for the purpose of getting rid of most of the 'useless' exact evaluations, reduces considerably the computational cost. The aim of the present paper is to employ a surrogate evaluation model in the context of a distributed genetic algorithm and to demonstrate that the combination of both results in maximum economy in CPU cost. In addition, whenever a multiprocessor system is available, the gain is much more pronounced, since the new optimization method maximizes parallel efficiency. The proposed method is used to solve inverse design and optimization problems in aeronautics and turbomachinery.

KEY WORDS: optimization; distributed evolutionary methods; genetic algorithms; radial basis function networks; metamodels; parallelization

1. INTRODUCTION

The design of optimal aerodynamic shapes is actually based on desirable pressure or velocity distributions over their contours—often resulting from inverse boundary layer computations—or the maximization of a fitness function measuring their aerodynamic performance (drag, lift, losses, etc.). For design purposes, evolutionary methods, among them genetic algorithms, GAs [1, 2], continue to gain ground against conventional steepest descent methods, since

*Correspondence to: K. C. Giannakoglou, Laboratory of Thermal Turbomachines, National Technical University of Athens, P.O. Box 64069, Athens 15710, Greece.
†E-mail: kgianna@central.ntua.gr

Contract/grant sponsor: A. S. Onassis Public Benefit Foundation Scholarship

Received 3 March 2002
Revised 11 March 2003

they are robust and capable to accommodate any commercial or in-house evaluation software (a Computational Fluid Dynamics (CFD) solver for aerodynamic analyses, a finite-element solver for structural analyses, etc.) without the slightest modification. Provided that the design parameters and the associated search spaces have been chosen properly, GAs are effective search algorithms and reach the optimal solution without getting stuck into local minima, even in complex multi-modal problems. However, since a major feature of GAs is the handling of populations of candidate solutions, excessive evaluations are usually required to locate the optimal solution. In aeronautics or turbomachinery the evaluations rely on CFD codes with high computing cost, so a GA-based design is time consuming and nowadays more effort is put to devise GA variants requiring fewer evaluations than their conventional counterparts.

In their previous works [3–8], the authors proposed a new way of simultaneously using exact (for the most important candidate solutions) and inexact (for the rest of the population) evaluation tools in the course of a GA-based optimization method. The concept of the aforementioned algorithm was quite simple: during the genetic evolution, store all the candidate solutions which have been evaluated through the costly CFD tool in a database and then use them systematically to forecast the merit of any new candidate solution. For this purpose, a surrogate model (often referred to as metamodel), which is less demanding in CPU cost than the exact evaluation tool, is dynamically built and used during a preliminary phase which will be referred to as the inexact pre-evaluation (IPE) phase. Among the various possible surrogate models, the authors preferably employ the radial basis function networks (RBFNs). In contrast to other rival techniques, distinct 'local' RBFNs, are trained for each individual, avoiding thus the modelling of the entire search space through a 'global' metamodel. In addition, the standard RBFNs are modified by taking into account the sensitivity of the cost or fitness function with respect to each one of the design parameters (sensitivity derivatives or importance factors, IFs, [7]). In the sake of completeness, this algorithm will be described briefly in a subsequent paragraph.

On the other hand, distributed genetic algorithms (DGAs) constitute an improved variant of their single-populated counterparts [9]. DGAs handle semi-isolated populations (demes), intercommunicating regularly by exchanging the most promising among their population members. Diversity in population is inversely proportional to the number of exchange cycles used. Different DGA variants can be devised by changing the connectivity of demes or the criteria for selecting the individuals to be replaced by immigrants.

The scope of this paper is to employ the IPE technique into a DGA. By analysing a number of inverse design or optimization problems in aeronautics and turbomachinery, as well as a typical function minimization problem with multimodal landscape, it will be demonstrated that the DGA enhanced by the surrogate models performs much better than the DGA or the GA with the same surrogate model; needless to say that all of them are much faster than the conventional GA. Therefore, the reduction in the number of required evaluations and, consequently, in the overall CPU cost is a major advantage of the proposed optimization method, which will be referred to as D(GA–IPE–IF) since it employs all the aforementioned features.

A second interesting advantage of D(GA–IPE–IF) is in relation to the optimal use of a multiprocessor system. The IPE technique reduces the number of evaluations per generation; thus, a parallel platform with a higher number of processors than the, generally small, number of exactly evaluated individuals per generation remains partially unexploited (assuming that the evaluation of a single individual is not parallelized). However, with D(GA–IPE–IF), where

many GAs with IPE are running simultaneously, a perfect exploitation of a multiprocessor system occurs. Parallelization issues or other practicalities, such as sharing to avoid premature convergence in demes, will be discussed as the paper develops.

2. INEXACT INFORMATION AIDED GENETIC OPTIMIZATION

As already mentioned in the introduction, the screening of the IPE phase relies on the low-cost estimations provided by the RBFNs. During the genetic evolution, for each new individual a RBFN is trained. For its training, a number of database entries which are closest to it should be located and used. The database is dynamically updated by storing any previously evaluated individual, along with its cost function value. This first order approximation of the fitness, obtained in the IPE phase, guides the GA to evaluate accurately only the most promising individuals. Thus, the role of RBFNs as metamodels is primordial and it is worth presenting, at least in brief, their architecture.

2.1. Radial basis function networks

A RBFN for single-output approximations is a three-layer, fully connected, feedforward network [10, 11] (Figure 1), which performs a non-linear mapping $\mathrm{H}: \mathbb{R}^N \to \mathbb{R}^M$ from the N input units to a layer of M hidden nodes, followed by a linear one $\Psi: \mathbb{R}^M \to \mathbb{R}$ to the output unit $y \in \mathbb{R}$

$$y = \Psi \mathbf{H} x \tag{1}$$

where $x \in \mathbb{R}^N$ is the input, i.e. the vector of design variables.

Each hidden node is associated with a point $c_m,\ m = 1, M$ in the design variables' space $\mathbb{R}^N$, which is referred to as the RBF centre. The non-linear mapping H is performed by the application of an activation function $F_m,\ m = 1, M$ to the deviation of the input $x \in \mathbb{R}^N$ from

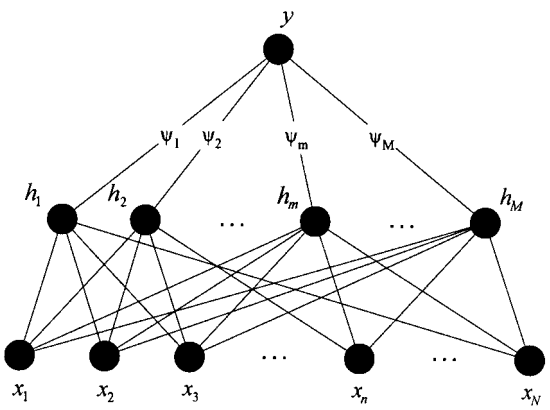

Figure 1. Radial basis function network.

the corresponding center on every hidden node. Hence, the response of each hidden node is

$$h_m(x) = F_m(\|x - c_m\|_2) = F(\|x - c_m\|_2, r_m) \tag{2}$$

A typical activation function can be $F(u,r) = \exp(-u/r)$. Here, we consider $r_m = r = 1$, $m = 1, M$.

The network training is equivalent to the computation of the M synaptic weights $(\psi_1, \psi_2, \ldots, \psi_M) = \psi$, which are the training parameters. It is performed by presenting the network with a number T of input–output pairs—i.e. airfoil shapes paired with cost function values. In the context of IPE, the number of training patterns is moderate, so it can be set $M = T$ and this yields a quite smooth input–output mapping. If $\hat{x}^{(t)}, t = 1, T$ is the t-th input pattern and $\hat{y}^{(t)}$ the corresponding output, the network training reduces to the solution of the linear system

$$\hat{H}\psi = \hat{\mathbf{y}} \tag{3}$$

where $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T)$ are the known outputs of the training patterns and the matrix $\hat{H}$ contains the responses of the hidden nodes to these patterns:

$$\hat{H}_{m,t} = h_m(\hat{x}^{(t)}), \quad m = 1, M \text{ and } t = 1, T$$

For $M = T, \hat{H}$ is a real symmetric positive-definite matrix.

A noticeable improvement over standard RBFNs was proposed by the authors [7], based on the self-adaptation of the network model to the problem itself. So, in Equation (2), a weighted norm was introduced that takes into account the importance of each one of the design variables to the fitness function. Practically, instead of the standard norm-2, its weighted variant is used,

$$u_m = \|x - c^{(m)}\|_{2,w} = \left(\sum_{i=1}^{N} I_i (x_i - c_i^{(m)})^2 \right)^{1/2} \tag{4}$$

where I_i—the so-called Importance Factors, IFs—quantify the relative importance of the design variables and can be calculated as follows:

$$I_i = \frac{|\partial y^* / \partial x_i|}{\|\nabla y^*\|_1} \tag{5}$$

Superscript $*$ denotes computations at a characteristic point, such as the current optimal point of the genetic evolution. It is interesting to stress that all partial derivatives are computed by the network itself,

$$y = \sum_{m=1}^{M} \psi_m h_m(u_m) \tag{6}$$

$$\frac{\partial y}{\partial x_i} = \sum_{m=1}^{M} \frac{I_i \psi_m}{u_m} (x_i - c_i^{(m)}) \frac{\mathrm{d}h_m(u_m)}{\mathrm{d}u_m} \tag{7}$$

The so modified RBFNs are autocatalytic, since IFs are computed from the RBFN and used for better training the network itself.

2.2. The algorithm

The population members of each generations, after being pre-evaluated through locally trained RBFNs, are rank sorted and the top σN_{pop} among them (N_{pop} is the population size and $0<\sigma<1$ is a small percentage defined by the user, usually $\sigma \approx 0.1$) undergo exact re-evaluations through the CFD tool. Only the latter are stored in the database along with their updated—exact—cost value. Genetic operations are then carried out regardless of whether individuals were exactly or inexactly evaluated, creating thus the next generation. In this manner, more generations are likely due for the same final solution quality. However, the number of exact evaluations required in total is much lower.

The small number of adjacent patterns used to train each RBFN makes the training procedure of almost zero computing cost. Therefore, the total CPU cost of the IPE task could safely be neglected, when compared to the cost for exactly evaluating even a small percentage of the current population.

Having discussed the IPE concept and described briefly the surrogate model, the inexact information based optimization algorithm (abbreviated to GA–IPE–IF, as mentioned in Introduction) is outlined below:

Phase 1: The starting population keeps evolving for a few generations, using exact evaluations; all of the evaluated individuals are stored in the database along with their cost values. The IFs are given equal initial values.

Phase 2: During the subsequent generations and for each individual, a local RBFN is trained using a small number of adjacent entries from the database. The trained RBFN provides an inexact cost or fitness value for the corresponding individual.

Phase 3: The σN_{pop} top individuals in the population undergo exact evaluations using the exact evaluation tool; so, the database is further enriched. Finally, if a new global optimum is found, a RBFN is trained in its region only for the purpose of updating the IF values.

Phase 4: N_{pop} new offspring are created through genetic operations using mixed up exact and inexact cost values. Phases 2–4 are repeated up to the final convergence.

3. DISTRIBUTED GENETIC ALGORITHMS

DGAs operate with multiple sub-populations, referred to as demes, which evolve independently and exchange information regularly through migration cycles. The regular exchange of individuals acts as a synchronization barrier to the genetic evolution of demes that participate in it. In various works, [9, 12, 13], the increased exploration ability of DGAs compared to their single population counterpart was demonstrated.

The two main advantages of DGAs are their persistent diversity in the populations thanks to the semi-isolation of demes as well as their straightforward implementation on parallel hardware. Thus the search algorithm becomes resistant to premature convergence and, in parallelized DGAs, the elapsed computing time is kept low. Homogeneous (with all demes undergoing identical genetic operations, through the same values for the control parameters), non-homogeneous (allowing different operators and parametric values among the demes) but also hybrid DGAs can be found in the literature. Coupled with a hierarchical structure, DGA's

efficiency can be enhanced further [12]. By itself, DGA may constitute the higher hierarchical level, allowing for different operators or evaluation tools to be used at the lower levels of this hierarchy. From a certain viewpoint, the implementation of the IPE phase within a DGA can be considered as a three-level hierarchical scheme where the separation into demes, the use of the RBFN for the IPE and the use of the costly exact evaluation tool constitute the three levels.

According to the authors' experience, which is in accordance to conclusions drawn in other papers on DGAs, medium or small sized populations with a small number of demes generally outperform any other configuration. Therefore, in this work, no complex migration graphs have been tried. However some migration scenarios have been evaluated. While the emigrants are always the top individuals in a deme, the replaced individuals in the host deme could be either the worst or some randomly chosen individuals. In the latter case, it is reasonable that the few top individuals in the host deme are excluded from being replaced. Elitism can be applied after the arrival of immigrants, though with the risk of increasing selective pressure; an effective counterpoise to this drawback is presented in Section 3.2.

3.1. Inexact information aided DGA-implementation

The scope of this paper was to modify DGAs by incorporating a low-cost surrogate model. The IPE phase, carried out separately within each deme, is expected to reduce further the computing cost. Autocatalytic RBFNs are used and the training patterns are selected from a single database, equally accessible by all demes; in this manner, the data collected during the semi-isolated genetic evolutions is made available to each and every deme. The previously described GA–IPE–IF algorithm is employed separately in each deme, giving rise to a new algorithmic variant abbreviated to *D*(GA–IPE–IF).

Aiming not only at the reduction of the number of evaluations needed to attain a given solution quality but also at increased overall performance, certain parallelization issues arise. The master–slave parallel model is an evident choice, with the DGA being the master who dispatches evaluation tasks to worker processes. In such a scheme, DGA involves two levels of synchronization: (a) inside demes, prior to the application of genetic operators (all the dispatched evaluation tasks should have been accomplished) and (b) between demes, prior to the migration of individuals (all the demes participating in it should have evolved for the necessary number of generations). This gives rise to the following requirements:

1. In-deme synchronization should be self-ruled: the synchronization inside demes, at the barrier of genetic operators, should be accomplished regardless of the situation in other demes.
2. Worker processes should not be bound to demes: when a deme is waiting for others to reach the migration barrier, all slaves should be disposable to the rest of demes that are still evaluating their individuals.

In order to meet these requirements, the *D*(GA–IPE–IF) algorithm was implemented as multithreaded and parallel application (Figure 2). Each deme evolves in its own thread, coordinated by the main process thread. Requests for evaluations are posted to an 'evaluation' server. The latter, using its own control threads, handles the worker processes and com-

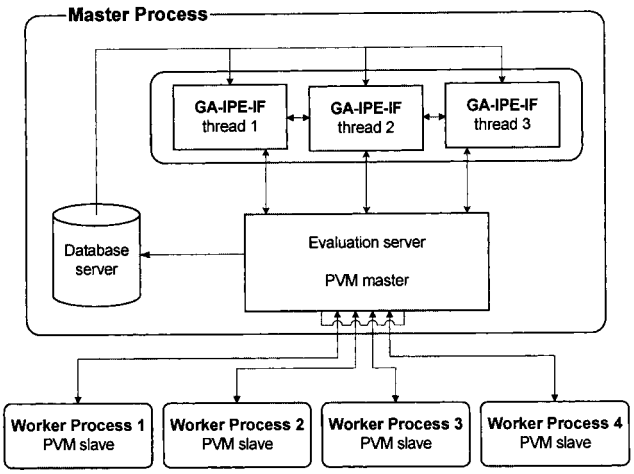

Figure 2. Schematic representation of the *D*(GA–IPE–IF) algorithm, on a multiprocessing system.

municates with them via PVM or MPI programming standards. The worker processes are completely hidden from the demes and each of them can evaluate individuals from anyone of the demes. The results of evaluations are stored in a commonly accessible database, from which they can be retrieved accordingly by the demes for the training of RBFNs. Despite the complexity of threads synchronization, the scheme described before fulfils completely the above stated specifications. Worker processes may remain idle only when some or all of the demes have reached the inevitable synchronization barrier of migration.

3.2. Preserving demes' diversity

Features such as high migration rates or immigrants including elitism may increase the selective pressure and the convergence rate at the early stages, but may also deteriorate convergence at its latest stages. In order to circumvent this drawback, without damaging the high starting convergence rate, sharing needs to be applied in some demes, whose population has become too uniform. This means that individuals that are too close to the current optimum are penalized, so as to foster exploration in more remote regions in the search space.

The application of sharing is conditioned by the main process thread. A deme is allowed to apply sharing to its individuals if its diversity has been reduced less than a threshold value. Sharing is applied at a small number of distinct generations (4–5 for DGA or 15–20 for the *D*(GA–IPE–IF) scheme) and is combined with slightly increased mutation probability. Only a small subset of the total number of demes is allowed to undergo sharing at a time. In addition, between two successive applications of sharing, a minimum number of generations of normal evolution is indispensable.

The following sharing function, which has zero derivative at zero distance, is proposed

$$s_i(z) = 4A \frac{\exp(-z_i)}{(1+\exp(-z_i))^2} \tag{8}$$

$$z = \frac{\|x_i - x_{\text{opt}}\|}{w} = \frac{d_i}{w}, \quad i = 1,\ N_{\text{pop}} \tag{9}$$

where x_i denotes the ith individual of the current population, whose fitness f_i becomes $f_i s_i$. In Equation (9) x_{opt} is the current optimal solution, whereas parameters A and w control the shape of the niche, represented by Equation (8).

Throughout the genetic evolution, A remains constant (an appropriate value could be $A=5$), the width w of the niche, however, is dynamically adjusted, taking into consideration the dispersion of the population at that generation. If $\bar{d}$ is the mean value of d_i and σ their standard deviation, then the width of the niche should be such that individuals whose distance from the current optimum is greater than $\beta\sigma$ to be penalized by a factor less than a αA, that is

$$s\left(\frac{\beta\sigma}{w}\right) = \alpha A, \quad \beta > 0 \text{ and } 0 < \alpha < 1 \tag{10}$$

This leads to

$$w = -\frac{\beta\sigma}{\ln[2/\alpha(1+\sqrt{1-a})-1]} \tag{11}$$

Suitable values are $\beta=2$ and $a=0.01$, which practically means that individuals, whose distance from the current optimum is greater than 2σ, will not be penalized.

Sharing, as described above, is applied only to a some of the demes and especially to those that tend to become too uniform, while the rest of them keep evolving normally. The impact of sharing on convergence speed will be demonstrated in the results section.

4. METHOD APPLICATION—COMPARATIVE STUDIES

In this section, the proposed D(GA–IPE–IF) will be compared with the conventional or other enhanced GA variants. In the airfoil inverse design or optimization problems that follow, there will be a shift of emphasis from physics to computational cost. For the sake of fairness in comparisons, all convergence rates will be measured and plotted in terms of exact evaluations, unless otherwise stated. We recall that the cost for training and using RBFNs during the IPE phase is negligible.

For the airfoil design problems, three flow analysis-evaluation tools have been used, namely:

1. a panel method for incompressible, irrotational flows,
2. a time-marching, inviscid flow solver for compressible fluids [14], accompanied by the unstructured grid generation software and
3. an integral boundary layer method coupled with an external flow solver (MSES software kindly provided by Drela, MIT [15]), with a range of validity that includes low-Reynolds numbers and transonic Mach numbers.

Table I. Suggested values of the D(GA–IPE–IF) algorithm parameters, which were kept constant to all problems examined.

GA	
Uniform crossover probability	90%
Mutation probability	1%
Binary tournament probability	90%
Coding	Gray binary

IPE	
Minimum exact evaluations	120
Exactly evaluated	10%

DGA	
Migration rate (gener.)	4
Number of emigrants	1
Elitism after migration	Yes
Replaced individuals	Worst
Migration graph	Each to all

The airfoil shapes were parameterized using two Bezier curves, one for each side. The first and the last control points along each curve were fixed, designating the leading and trailing edges (LE/TE). The LE node and the first control points next to it over both airfoil sides were alligned, for a rounded edge to be formed.

Table I summarizes the major D(GA–IPE–IF) parametric values that were common to all of the examined problems. It is interesting to note that these values imply an increased selective pressure within each deme. This may affect differently the convergence characteristics of the algorithm during its first and last stages. In the first stages, the high selective pressure is desirable so as D(GA–IPE–IF) to be endowed with convergence properties equivalent to these of GA–IPE–IF. On the other hand, premature convergence is lurking and, for this reason, sharing (as previously described) should necessarily be applied.

4.1. Minimization of the Rastrigin function

Prior to presenting airfoil design problems, the minimization of a known mathematical function, i.e. the so-called *Rastrigin* function, will be demonstrated. The problem involves $N = 30$ variables, varying in $[-5.12, 5.12]$ and requires the minimization of

$$f(x) = 10 \cdot N + \sum_{i=1}^{N} (x_i^2 - 10\cos(2\pi x_i))$$

Since the solution landscape is multi-modal, this problem is well-suited for assessing the efficiency of optimization methods. For the GA variants (single deme) the size of population

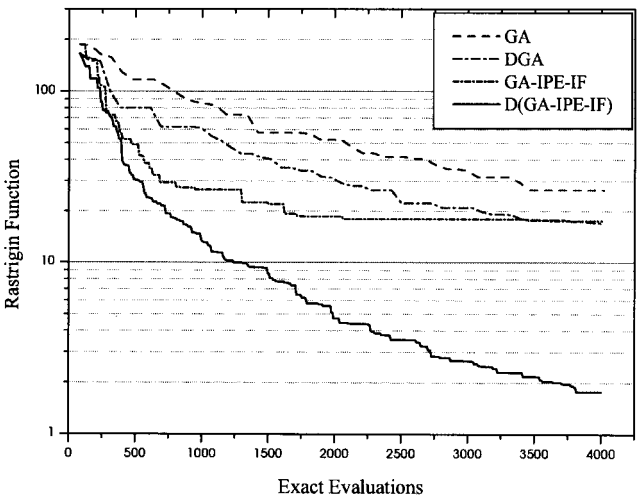

Figure 3. Design of a compressor cascade airfoil: convergence history.

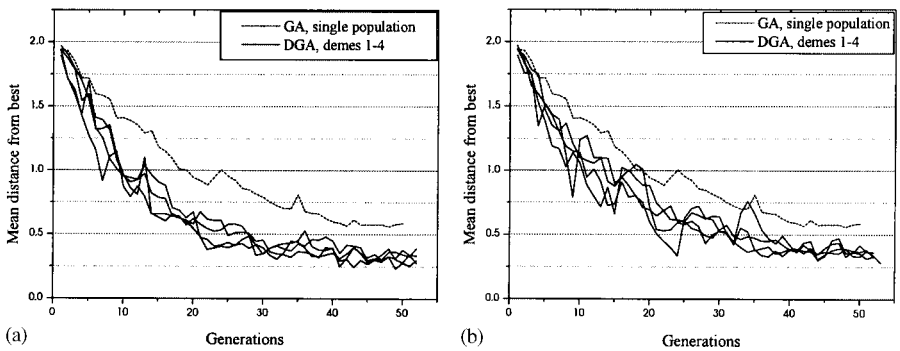

Figure 4. Design of a compressor cascade airfoil: the mean distance from the current optimal solution in each generation.

was 80, while for the DGA ones 4 demes of 20 individuals each were employed. In all the computations illustrated in Figure 3, 4000 evaluations at most were allowed. It is clear that *D*(GA–IPE–IF) prevails over any other scheme. During the early stages, its convergence rate is similar to that of GA–IPE–IF. However, as evolution goes on, *D*(GA–IPE–IF) clearly outperforms GA–IPE–IF.

During the previously shown computations, sharing was applied to the *D*(GA–IPE–IF) algorithm, which helps maintaining high diversity in demes. As a consequence, the exploration of the search space is better. Due to its major effect, a quantitative analysis of sharing is necessary. So, Figure 4(a) plots the mean distance of the individuals in each generation from the current optimal solution for a DGA without sharing. This distance measures the population diversity and it is interesting to note that DGA yields lower mean distance values than the

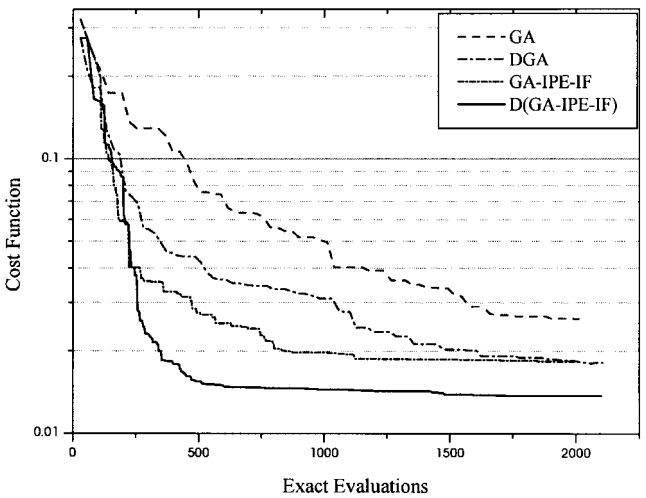

Figure 5. Design of a compressor cascade airfoil: convergence history.

single-population GA. The application of sharing to two of the demes at most each time increases the mean distance, i.e. the diversity in each deme, and yields better exploration capabilities, as shown in Figure 4(b).

4.2. *Inverse design of the NACA4412 profile*

This test-case deals with the redesign of the NACA4412 airfoil, the target being a known pressure distribution at zero incidence. This distribution was computed using the panel method, i.e. the same analysis tool used for the exact evaluations during optimization. For the parameterization of the airfoil, four control points per side were used, giving rise to 14 free variables (recall that at the LE geometrical constraints are applied).

This run was carried out using a population of 30 individuals; the D(GA–IPE–IF) was configured to employ 3 demes with 10 individuals each. The maximum number of evaluations was set equal to 2000. It is interesting that, even with small deme populations, D(GA–IPE–IF) outperforms any other scheme, as shown in Figure 5. In Figure 6, the best solution is compared to the target airfoil. Some slight discrepancies in the c_p distribution close to LE are due to the small number of Bezier control points used to parameterize the airfoil.

4.3. *Drag reduction of the RAE2822 airfoil*

The next case deals with the optimization of the well known RAE2822 airfoil, the target being the reduction of the airfoil drag, while maintaining the same lift. The flow conditions were chord-based $Re_\infty = 6.2 \times 10^6$, freestream Mach number $M_\infty = 0.75$ and flow angle $a_\infty = 2.734°$. Transition fixing on both airfoil sides was imposed, as in the original case. The integral boundary layer (MSES) software was used as evaluation tool. At these flow conditions, the original airfoil yields lift and drag coefficients equal to $c_{l,\mathrm{ref}} = 0.749$ and $c_{d,\mathrm{ref}} = 0.0235$. The cost function was defined as $f = (c_{l,\mathrm{ref}} - c_l)^2 + \beta \cdot c_d$, where $\beta = 2$.

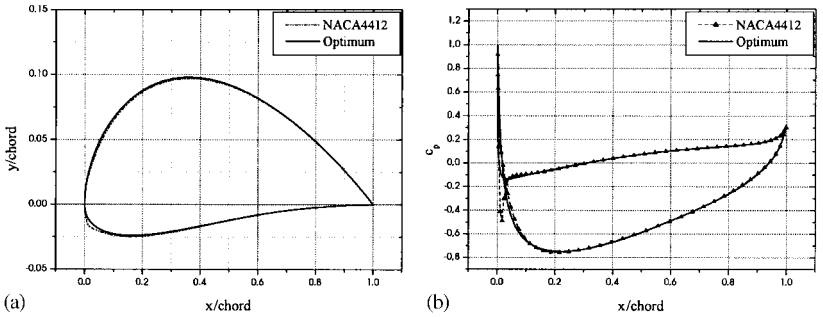

Figure 6. Design of a compressor cascade airfoil: optimal versus reference airfoil shapes and the corresponding pressure coefficient distributions.

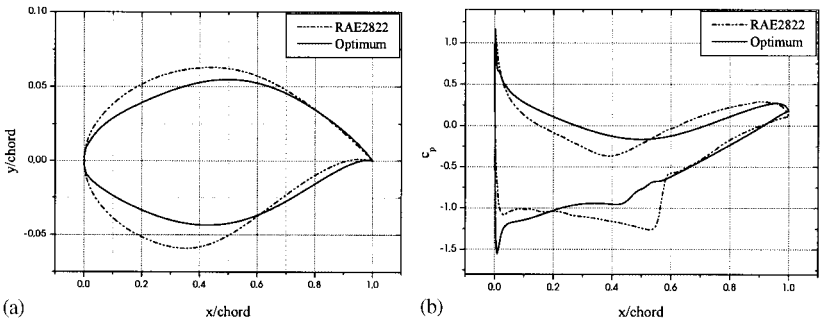

Figure 7. Design of a compressor cascade airfoil: comparison of optimal and reference airfoils.

The computed optimal airfoil, along with the corresponding pressure coefficient distribution at the aforementioned flow conditions are compared to the reference data (Figure 7). As it can be deduced from Figure 7(b), the drag inducing shock wave was fully eliminated. For the optimal design, the lift and drag coefficients were equal to $c_l = 0.744$ and $c_d = 0.00963$.

For GA and GA-IPE-IF, the population size was 80 while for DGA and D(GA-IPE-IF) 4 demes of 20 individuals each were used. The convergence history of these four algorithmic variants is plotted in Figure 8. It is evident that D(GA-IPE-IF) locates a better design much faster than any other algorithm. This is particularly important in this problem where, during the first generations, a noticeable percentage of the examined individuals (even close to 50%) fails to converge. It should be made clear that any candidate airfoil shape, for which the integral boundary layer method does not converge within the allowed number of iterations, is given a high cost value.

Here, also, the role of sharing is important, as illustrated in Figure 9. Figure 9(a) presents the mean distance of all population members from the current optimal solution within the first two demes; the same quantity for the single-populated GA-IPE-IF is shown too. The peaks that appear in the mean distance curve for the second deme indicate the regular application of sharing, either to the deme itself or to any other deme, with which individuals have been exchanged. According to the criterion used, sharing was first applied during the 52th

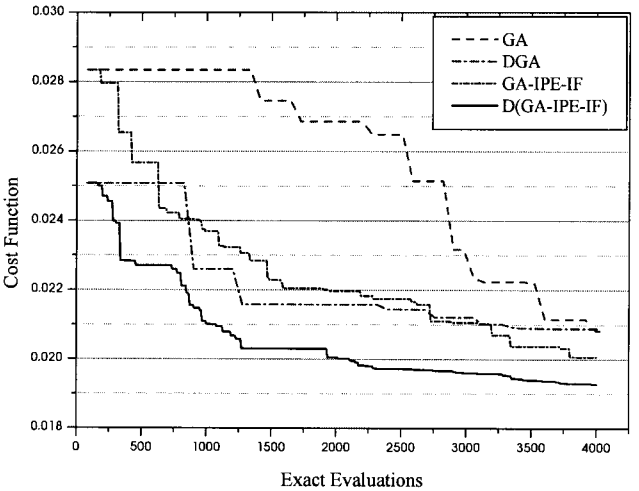

Figure 8. Design of a compressor cascade airfoil: convergence history.

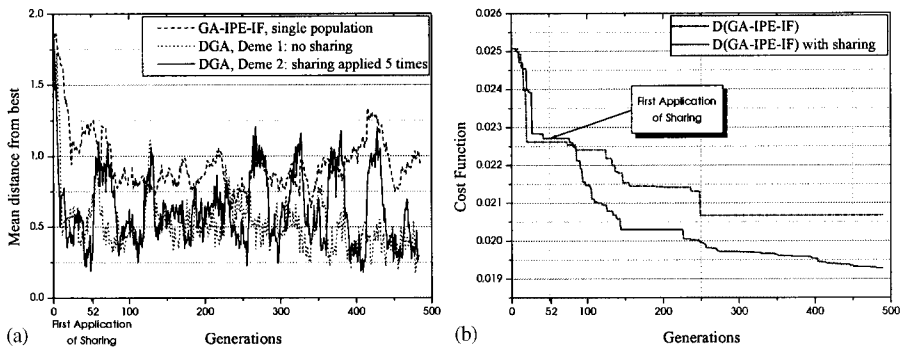

Figure 9. Design of a compressor cascade airfoil: effect of sharing in the *D*(GA-IPE-IF) scheme.

generation. A few generations later, the *D*(GA-IPE-IF) with sharing begins to outperform the variant without sharing, as shown in Figure 9(b).

4.4. Lift maximization of a three-element airfoil

In the present case, the aim was to maximize the lift produced by a three-element, configuration. The shape of its components—slat, main body, flap—was given and only their relative positions were allowed to vary within predefined bounds. Thus, the design variables are six: horizontal and vertical distances from the main body and the rotation angle for flap and slat. The cost function was defined to be equal to $f = -c_l$, where c_l denotes the lift coefficient of

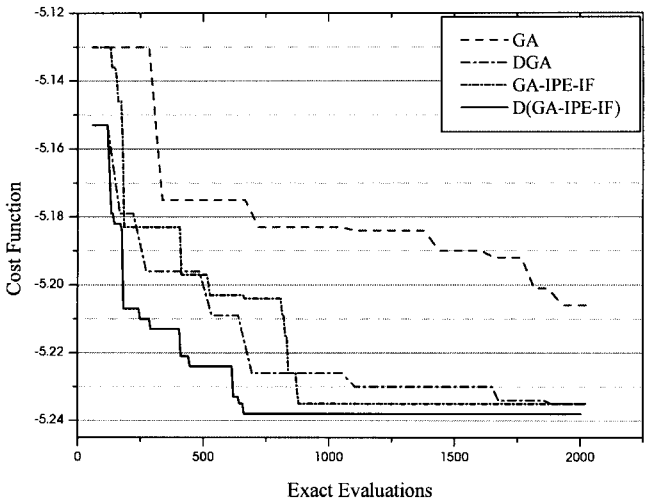

Figure 10. Design of a compressor cascade airfoil: convergence history.

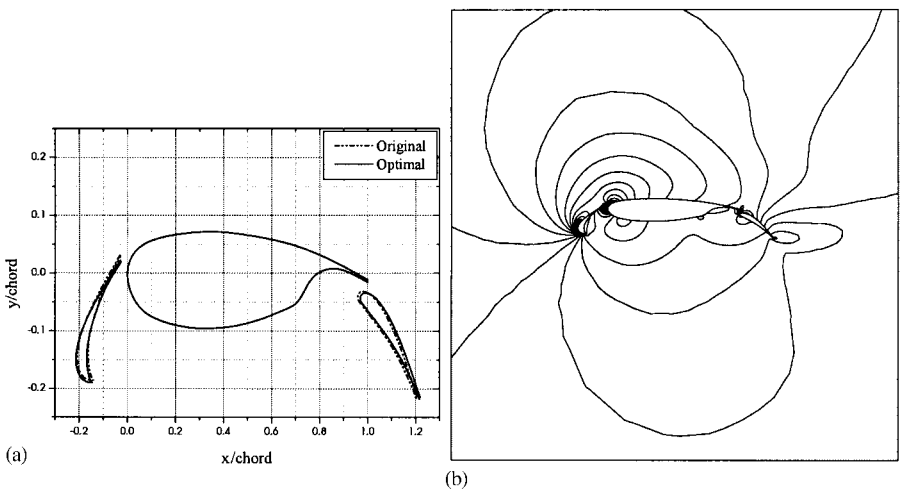

Figure 11. Design of a compressor cascade airfoil: analysis of the optimal configuration.

the entire configuration. This case was also analyzed in previous publications by the authors [7, 8] but, in this paper. emphasis is given to the use of distributed algorithms.

The convergence rates of the tested algorithms are presented in Figure 10. Once more, the superiority of *D*(GA-IPE-IF) is evident. The runs were carried out with population size equal to 60 for GA and GA-IPE-IF or with 2 demes of 30 individuals each for the distributed-algorithms. *D*(GA-IPE-IF) employed the sharing technique presented in previous sections.

The optimal slat/flap positions are shown in Figure 11, along with the iso-Mach contours of the flow field around the high-lift configuration, as computed by an Euler equations solver.

4.5. Design of a compressor cascade airfoil

The last case aims at the design of an optimal compressor cascade airfoil, operating at inlet Mach number $M_1 = 0.7$, inlet flow angle $a_1 = 45°$ and chord-based $Re = 4 \cdot 10^5$.

The airfoil was parameterized using 8 control point Bezier curve for the suction side and 7 control point one for the pressure side, combined with a circular arc at LE [7]. In addition, the stagger angle was allowed to vary between -20 and $-10°$ and the pitch (with unit chord) between 0.55 and 0.65.

The target was the minimization of the total pressure loss coefficient of the cascade with an exit flow angle as close as possible to 0°. Moreover, the maximum thickness of the airfoil had to be greater than 7% of the chord length. To deal with these two constraints the cost function had the general form of

$$f = \text{losses} \cdot p_{\text{turning}} \cdot p_{\text{thickness}}$$

where p_{turning} and $p_{\text{thickness}}$ represent exponential penalty functions for unacceptable designs.

The scope of tuning this case was slightly different from the previous. As developed in previous sections, GA–IPE–IF reduces the individuals, which need to be exactly evaluated per generation, to a small subset of the entire population. Since the application of genetic operators acts as a synchronization barrier, only σN_{pop} individuals can be evaluated simultaneously. If more processors are available, they remain idle (assuming that no parallelization of the evaluation software is done). This bottleneck can successfully be circumvented by *D*(GA-IPE-IF), which employs multiple GA-IPE-IF schemes at the same time. Therefore, the present test-case aims at demonstrating that *D*(GA-IPE-IF), besides its undisputed exploration ability, alleviates this inherent drawback of a GA-IPE-IF working on a multiprocessor platform.

For this reason, 12 identical processors have been used. Each GA or DGA variant was distinctly configured so as to optimally exploit the available processors; the termination criterion was defined by the maximum number of 2000 exact evaluations (Figure 12). The configurations that were used are listed below:

GA : the population size was 84 $(= 7 \times 12)$,

GA-IPE-IF : the population size was 80, with only 8 individuals being exactly evaluated per generation. An attempt to evaluate 12 individuals per generation gave worst results, due to the smaller number of generations required to reach 2000 exact evaluations.

DGAand *D*(GA-IPE-IF)**:** 3 demes of 40 individuals each have been used. Only 4 individuals were evaluated exactly per deme and generation.

In this test-case only, the cost function is plotted against elapsed time in sec and the convergence history is given in Figure 13. It is obvious that it takes much more elapsed time for GA-IPE-IF to complete 2000 exact evaluations, despite its exploitation potential. *D*(GA-IPE-IF), however, achieves the same solution quality as GA-IPE-IF more than 4 times earlier and finally locates the best solution, which is plotted in Figure 14(a). The distribution of isentropic Mach number for this design is given in Figure 14(b).

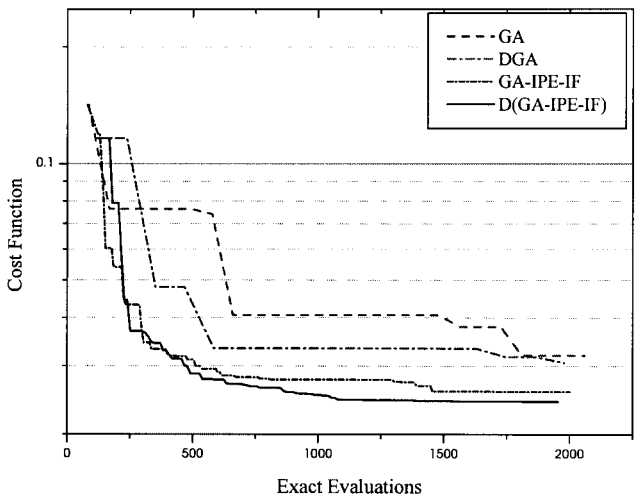

Figure 12. Design of a compressor cascade airfoil: convergence history in terms of exact evaluations.

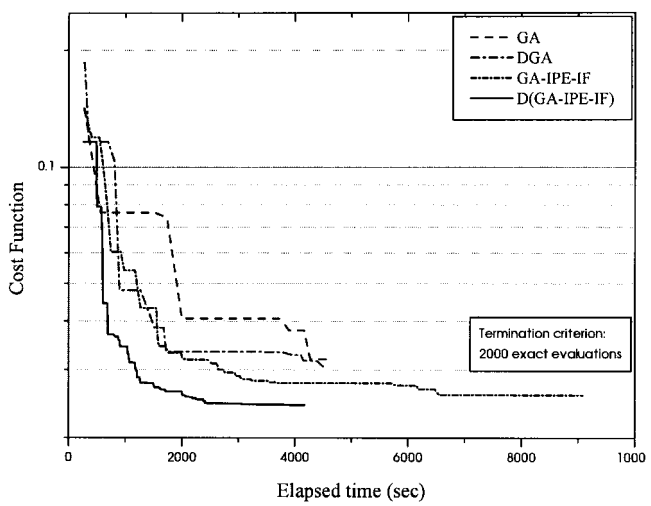

Figure 13. Design of a compressor cascade airfoil: convergence history in terms of elapsed time.

5. CONCLUSION—COMMENTS

This paper extended the idea of using the so-called inexact pre-evaluation phase in distributed genetic algorithms, giving rise to the scheme denoted as *D*(GA-IPE-IF). The goal of the study was to investigate whether the distributed variant of GA-IPE-IF is well performing and

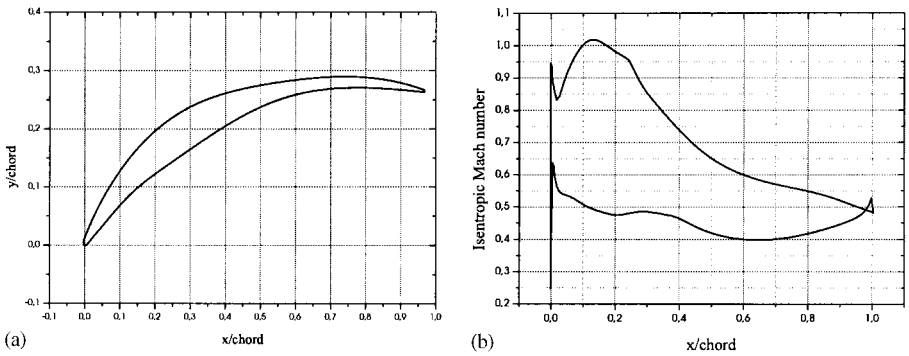

Figure 14. Design of a compressor cascade airfoil: optimal airfoil shape and the corresponding isentropic Mach number distribution around it.

to quantify the gain in CPU cost or, equivalently, the reduction in the required number of evaluations for the same quality of final solution. In the sake of completeness and fairness, D(GA-IPE-IF) was constantly compared to DGA and GA-IPE-IF as well as to traditional GA.

The main conclusions are outlined below:

- Given the superiority of GA-IPE-IF or DGA with respect to conventional GAs, D(GA-IPE-IF) offers a new optimization algorithm, which is both faster and with better exploration abilities than GA-IPE-IF and DGA. A practical outcome of the research exposed in this paper could be a hierarchy of genetic optimization methods, where the four variants used are listed in CPU-cost descending order:

  - GA
  - DGA
  - GA-IPE-IF
  - D(GA-IPE-IF)

- Apart from reduction in the number of required exact evaluations, D(GA-IPE-IF) offers the additional advantage of optimally exploiting a greater number of interconnected processors, since the simultaneously evolving demes are associated with more than one distinct GA-IPE-IF algorithms. Consequently, the upper limit of non-idle processor can be much higher than σN_{pop}, which is the upper limit in a single-populated GA-IPE-IF (assuming that a sequential evaluation software is used). Sharing, applied to demes that tend to become too uniform, is necessary to avoid premature convergence.

ACKNOWLEDGEMENTS

The first author was supported by a A.S. Onassis Public Benefit Foundation Scholarship.

REFERENCES

1. Goldberg DE. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley: Reading, MA, 1989.
2. Michalewicz Z. *Genetic Algorithm + Data Structure = Evolution Programs* (2nd edn). Springer-Verlag: Berlin, 1994.
3. Giotis AP, Giannakoglou KC. Single- and multi-objective airfoil design using genetic algorithms and artificial intelligence. In *EUROGEN 99, Evolutionary Algorithms in Engineering and Computer Science*, Jyvaskyla MK *et al.* (eds). Wiley: Finland, 1999.
4. Giotis AP, Giannakoglou KC, Periaux J. A reduced cost multiobjective optimization method based on the pareto front technique, neural networks and PVM. In *ECCOMAS 2000*, Barcelona, September 2000.
5. Giannakoglou KC. *Acceleration of Genetic Algorithms using Artificial Neural Networks—Theoretical Background.* Von-Karman Institute LS 2000-07, May 2000.
6. Giannakoglou KC, Giotis AP. *Acceleration of Genetic Algorithms using Artificial Neural Networks—Application of the Method.* Von-Karman Institute LS 2000-07, May 2000.
7. Giannakoglou KC, Giotis AP, Karakasis MK. Low-cost genetic optimization based on inexact pre-evaluations and the sensitivity analysis of design parameters. *Journal of Inverse Problems in Engineering* 2001; **9**:389–412.
8. Giannakoglou KC. Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. *Progress in Aerospace Sciences* 2002; **38**:43–76.
9. Doorly DJ, Peiro J. Supervised parallel genetic algorithms in aerodynamic optimization. In *13th AIAA CFD Conference*, Snowmass, Colorado, USA, June 30–July 2, 1997.
10. Poggio T, Girosi F. Networks for approximation and learning. *Proceedings of the IEEE* 1990; **78**(9):1481–1497.
11. Haykin S. *Neural Networks* (2nd edn). Prentice-Hall International, Inc: Englewood Cliffs, NJ, 1998.
12. Herrera F, Lozano M, Moraga C. Hybrid distributed real-coded genetic algorithms. *PPSN V, Lecture Notes in Computer Science*, vol. 1498. Springer: Berlin, 1998; 603–612.
13. Voigt HM, Born J. A structured distributed genetic algorithm for function optimization. *PPSN II* 1992; 199–208.
14. Koubogiannis DG, Poussoulidis LC, Rovas DV, Giannakoglou KC. Solution of flow problems using unstructured grids on distributed memory platforms. *Computer Methods in Applied Mechanics and Engineering* 1998; **160**: 89–100.
15. Drela M, Giles MB. Viscous-inviscid analysis of transonic and low reynolds number airfoils. *AIAA Journal* 1987; **25**(10):1347–1355.

