Multilevel Optimization Strategies Based on Metamodel–Assisted Evolutionary Algorithms, for Computationally Expensive Problems

I.C. Kampolis, A.S. Zymaris, V.G. Asouti, K.C. Giannakoglou

Abstract— **In this paper, three multilevel optimization strategies are presented and applied to the design of isolated and cascade airfoils. They are all based on the same general–purpose search platform, which employs *Hierarchical, Distributed Metamodel–Assisted Evolutionary Algorithms* (HDMAEAs). The core search engine is an Evolutionary Algorithm (EA) assisted by local metamodels (radial basis function networks) which, for each population member, are trained anew on a "suitable" subset of the already evaluated solutions. The hierarchical scheme has a two–level structure, although it may accommodate any number of levels. At each level, the user may link (a) a different evaluation tool, such as low or high fidelity discipline–specific software, (b) a different optimization method, selected amongst stochastic and deterministic algorithms and/or (c) a different set of design variables, according to coarse and fine problem parameterizations. In the aerodynamic shape optimization problems presented in this paper, the three aforementioned techniques resort on (a) Navier–Stokes and integral boundary layer solvers, (b) evolutionary and gradient–descent algorithms where the adjoint method computes the objective function gradient and (c) airfoil parameterizations with different numbers of Bézier control points. The EAs used at any level are coarse–grained distributed EAs with a different MAEA at each deme. The three variants of the HDMAEA can be used either separately or in combination, in order to reduce the CPU cost. The optimization software runs in parallel, on multiprocessor systems.**

NOMENCLATURE

DB	Database (of already evaluated solution)
DEA	Distributed EA
EA	Evolutionary Algorithm
$HDMAEA$	Hierarchical Distributed MAEA
HEA	Hierachical EA
IPE	Inexact Pre–Evaluation
$MAEA$	Metamodel–Assisted EA
PEA	Parallel EA
RBF	Radial Basis Funstion (network)
SQP	Sequential Quadratic Programming

I. INTRODUCTION

During the last two decades, the development of computational methods for solving complex industrial design–optimization problems has been boosted. Among the existing methods, Evolutionary Algorithms (EAs) have gained particular attention since they are friendly and robust optimization tools, suitable for discontinuous and multimodal objective functions. None the less, it is hard to say that conventional EAs are routinely being used to solve industrial problems, due to the large number of calls to the evaluation software. In aerodynamic design, where the evaluation relies upon computationally expensive CFD tools, the development of design methods that minimize the use of the CFD software is an area of active research.

According to the literature, possible remedies to this problem are: (a) the use of Metamodel–Assisted Evolutionary Algorithms (MAEAs), (b) the use of Parallel EAs (PEAs), (c) the use of hierarchical evolutionary search and (d) the hybridization of EAs with deterministic optimization methods. All of them are ingredients of the proposed method and will be discussed below. To avoid ambiguities, in what follows, *high fidelity model* stands for the more accurate and costly evaluation tool the designer selects for analyzing candidate solutions and computing objective function values f, the *low fidelity model* provides cheaper approximation of f based on assumptions and simplifications (both might be referred to as *exact models* since they are problem specific tools) and a *surrogate* or *metamodel* stands for a mathematical approximation to the analysis model. Surrogates of the high or low fidelity models and the constraint functions can be devised.

MAEAs rely on the smart management of calls to the exact evaluation tool and its metamodel during the evolution, leading to a considerable economy in CPU cost, [1]. EAs may use metamodels trained on samples selected separately from the evolution, [2], [3]; in this case, the metamodel should be updated regularly depending on the deviation between the f values computed on the metamodel and the exact tool. On the other hand, EAs assisted by *on-line* trained metamodels can be used. A locally valid metamodel is constructed on the fly for each new individual, by training it on previously evaluated neighboring individuals. Through the metamodel–based evaluation, a few promising members in each population are identified and only these are to be re–evaluated on the exact tool. This will be referred to as the *Inexact Pre–Evaluation (IPE) technique*, [4]. Among the most frequently used metamodels, response surface methods, polynomial interpolation and various types of artificial neural networks, Gaussian processes, etc can be found. There is a large literature on MAEAs, [1], [4], [5], [6], [7], [8], [9], [10], [11] to mention only a few of the relevant papers.

PEAs are efficient variants of EAs which may directly map onto the topology of a parallel computing platform, [12], [13], [14]. PEAs are used as fine–grained (cellular) and coarse–grained (distributed) EAs. In Distributed EAs (DEAs), which will be used in this paper, a few population subsets (demes) evolve in semi–isolation and regularly exchange promising individuals, [15], [16]. By associating

National Technical University of Athens, Greece, (email: kgianna@central.ntua.gr).

different evolution parameters with each deme, diversity and new species formation increase.

Another technique for increasing the efficiency of EAs, without resorting to optimization methods other than EAs, is to perform a hierarchical search (Hierachical EAs, HEAS, [17], [18]). In HEAs, low and high fidelity models are used with a certain hierarchy according to a multilevel (usually, two–level) search structure: at the low level, the low fidelity tool undertakes the exploration of the design space whereas, at the high level, a restricted search based on the high fidelity model performs practically on migrated promising solutions. Alternatively, a two–level search can use different sets of design variables at each level (some design variables at the low level and full parameterization at the upper level), with the same evaluation tool. In aerodynamic shape optimization problems where shapes can be parameterized, for instance, using Bézier curves or surfaces, we propose to maintain a small number of design variables (Bézier control points) at the low level and enrich them, using knot insertion theory, when moving to the upper level. At each level of the hierarchical or multilevel scheme, a distributed MAEA can be used, giving rise to the so–called *Hierarchical Distributed Metamodel–Assisted Evolutionary Algorithm* (HDMAEA).

Last but not least is the possibility to use different search methods at the different levels. The idea to hybridize, for instance, EAs and gradient–based search methods, is not a new one. A series of papers appeared in the literature addressing possible hybridization schemes, [19], [20], [21], [22], [23]. In this paper, the hybridization of the distributed MAEAs with gradient–based search methods (steepest–descent, conjugate gradients or Newton–like methods supported by the adjoint method to compute the gradient of the objective function, [24]) is carried out within the multilevel scheme; the two search tools are associated with different levels of the multilevel scheme. Since EAs still undertake the key search role, the hybrid method will be still referred to as HDMAEA.

II. The Proposed Multilevel Optimization

As mentioned before, each level of the proposed multilevel optimization algorithm can be associated with a different evaluation tool (*Multilevel Evaluation*), different search algorithm (*Multilevel Search*) and/or a different set of design variables (*Multilevel Parameterization*). The practical implementation of the three proposed schemes are discussed in this section. Emphasis is laid to the *Inter–level Migration* policy, which is discussed in a separate subsection.

A. Multilevel Evaluation

We have already shed some light into the principles of an HDMAEA employing the low fidelity model at the low level and the high fidelity one at the high level. Such an algorithm is schematically shown in fig.1, with L=2 levels. We assume that, at each level, the search is undertaken by a DMAEA. At the two levels, the number of demes and/or their populations are not necessarily the same. It is usually recommended that the low level uses a higher number of demes than the high level, possibly with a higher population

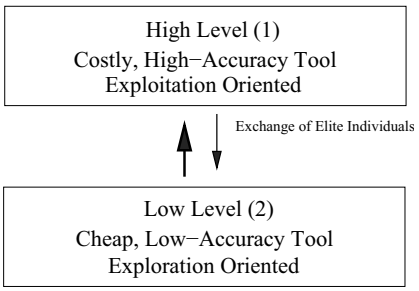

Fig. 1. Multilevel structure: The low level utilizes a low–CPU cost and low–accuracy tool to explore the design space with a minimum impact to the wall clock time. The high level, using the high fidelity, high–CPU cost tool is restricted to exploiting the information from the low level.

per deme. This can be easily understood since the low level is responsible mainly for searching in unexplored areas of the search space with low cost. For this reason, higher mutation probabilities (usually a $3-4\%$) and lower selection pressure should be used at the low level. A two–way communication between the two levels is used. Individuals that migrate from one level to another are automatically (re)evaluated using the destination level model. At the low level evolution terminates, if it consistently fails to provide the upper level with well performing solutions. In a well–tuned multilevel evaluation HDMAEA, the gain in CPU cost is expected to increase as the execution time ratio of the high and low fidelity models increases.

A multilevel evaluation HDMAEA maintains separate "databases"(DB) at each level. A DB gathers all the previously evaluated solutions (paired sets of design variables and performances; the values of the constraint functions should also be stored in the DB if there are not lumped into the objective function using suitable weights) and is used to train local metamodels for the IPE task (see below). At each level, the metamodels are trained on datasets evaluated with the level's model. Parameters defining the use of metamodels or, even, the metamodel type may differ from level to level.

B. Multilevel Search

It aims at increasing the overall efficiency of the design platform by employing different search algorithms at each level. At the low level, a MAEA is usually chosen so as to quickly detect near–optimum regions. Elite individuals of the MAEA are transferred to the upper level, which employs a local search technique (often, but not necessarily, a gradient–based search algorithm; a stochastic local search method could be used instead) so as to "refine" them at low cost. The regular backward migration (from the high to the low, MAEA–based level) is also allowed to increase the selective pressure at near–optimum regions of the design space. In backward migration, some of the high level elite members migrate to the low level. A multilevel search mode that uses gradient–based optimization requires the availability of a low–cost software to compute the gradient of f. Multi–objective problems can also be handled by concatenating the objectives in a single utility function. In this case, however,

the cost of computing the gradient of f at many non–dominated solutions is not negligible at all.

The local search methods supported by our platform are: (a) steepest descent, (b) conjugate gradient and (c) trust region SQP, where the Hessian matrix is approximated using the BFGS method. The adjoint method is used to compute the gradient of f (this is an in–house tool, [24], associated with the in–house Navier–Stokes solver).

C. Multilevel Parameterization

This mode allows the association of each level with a different set of design variables. The low level solves a problem in a reduced dimension search space and locates a near–optimum solution, at low cost. Through migration followed by a suitable transformation, elite individuals stemming from the low level enter the high level population. The inverse transformation is used for individuals migrating from the high level. As it will be shown in the results section, the same evaluation model is employed at both levels; due to the small number of design variables it handles, the low level is much more efficient.

The use of a multilevel parameterization MAEA requires a couple of practical problems to be solved. By way of example, assume that the aerodynamic shapes are parameterized using Bézier (or NURBS) curves, [25]. Then, the corresponding knot insertion (during upward migrations) and knot removal theory (during downward migrations) can be employed. When an elite airfoil from the low level (described by a few control points) migrates to the upper level, a new control point can be added (the $N+1$ points become $N+2$), as follows:

$$\vec{R}_0 = \vec{r}_0$$
$$\vec{R}_i = \frac{i}{N+1}\vec{r}_{i-1} + \left(1 - \frac{i}{N+1}\right)\vec{r}_i,\ 0 < i < N+1$$
$$\vec{R}_{N+1} = \vec{r}_N$$

where $\vec{r}$ and $\vec{R}$ denote the position vectors of the initial and resulting curves. By successively applying eq.1 (N_1–N_2 consecutive times, where N_1 and N_2 are the numbers of control points at the high and low level, respectively) the transformation of parameterization is accomplished. However, what is not obvious is the definition of the search space at the high level, given the lower and upper bounds of the small number of control point coordinates used at the low level (and vice–versa). Empirical algorithms have been devised to solve this problem; we believe it is beyond the scope of this paper to analyze them in detail. Note that, when an immigrant joints the high level, the coordinates of its design variables may fall outside the bounds currently used. In such a case, the bounds are readapted and all population members are transformed, accordingly.

D. Inter–level migration

The inter–level agent, which is responsible for all migrations between two adjacent levels, runs in its own thread waiting for the synchronization barrier for both levels to notify that a certain generation has been reached. Once this signal is received, the synchronization starts gathering elite individuals from both levels, transforming individuals according to the parameterization at the destination level (if needed; this is the case of multilevel parameterization) and re–evaluating (if a different evaluation software is associated with each level). This procedure is described in pseudocode, listing 1. During migration, evolution at all levels is suspended. This is accomplished by the implementation of a generic agent–client system inside each level, as in listing 2. Regarding the inter–level migration, two markers are associated with each level: (a) the generation after which the first migration occurs and (b) the migration frequency, measured in terms of number of generations. The level that first reaches the synchronization barrier suspends its evolution until the same happens for the other. Then, the thread of the migration agent resumes execution in order to carry out the inter–level migration, as described before. Throughout the entire optimization process, the migration between two successive levels behaves differently. In the beginning, there is a large deviation between the fitness values of the elites of the two levels. The low level provides better performing solutions, due to the higher number of generations that have evolved. The information is, however, propagated to the high level, which eventually catches up, evolving for a significantly lower number of generations that correspond to a reduced CPU cost. Towards the end of the process, the elites of both levels represent individuals associated with almost equal fitness values; in case of large deviations between high and low level, due to the simplifications of the low level tool, the evolution of the low level is terminated.

```
while (Running)
{
  WaitUntilGenerationReached();
  GatherElitesFromLevels();
  if (DifferentParameterizations)
    TransfromParameterizations();
  ReEvaluateElites();
  SendElitesToLevels();
  SignalLevels();
}
```

Listing 1. The inter–level migration agent in pseudocode.

```
InitializeLevel();
generation=0;
while (NotConverged)
{
  EvaluateCandidateSolutions();
  CalculateStatistics();
  NotifyAgents(generation);
  WaitForAgents();
  CreateNewIndividuals();
  generation++;
}
```

Listing 2. Actions performed by every level.

III. MAEAs; The IPE Technique

EA is the key search engine used in this work. This is, herein, used in the form of MAEA which exhibits the same effectiveness without requiring a large number of calls to the exact model. In this paper, radial basis function (RBF) networks are used [26].

A MAEA implementing the Inexact Pre–Evaluation (IPE) techinque starts as a conventional EA. In each generation, all the population members are evaluated on the exact model and recorded in the DB. The continuously augmenting DB is the pool from which datasets for the training of local metamodels are selected. Once the DB has been filled with a sufficient number of entries (this is a user–defined parameter), the population members are first evaluated on metamodels. For each member, its closer entries are selected from the DB. The user defines the minimum and maximum number of training patterns; this allows each metamodel to be trained on a slightly different number of patterns, the criterion being the "structure" of the available information in its neighborhood. The reader should refer to [16] for a detailed description. A much more sophisticated algorithm should be used in multi–objective optimization, as described in [27].

At the end of the IPE phase, all offsprings are sorted with respect to their approximate fitness value. Only the most promising among them (usually $5-10\%$ of their population size) are re–evaluated using the exact model and added to the DB. Since the evaluation using the exact model is the only computationally intensive task, the CPU cost per generation is proportional to the number of individuals evaluated on the exact model.

IV. Applications: Results & Discussion

Design problems in external aerodynamics and turbomachinery have been worked out using the proposed multilevel optimization platform. The demonstration covers all three multilevel techniques, used separately on different test problems. The exact evaluation model (the high fidelity one in multilevel evaluation) is a CFD code. This is an in–house Navier–Stokes (N/S, [28]) flow solver based on a time–marching, vertex–centered, finite volume formulation for adapted unstructured meshes, employing a second–order accurate Riemann solver and the Spalart–Allmaras turbulence model, [29]. As low fidelity model (or the only exact tool, in some cases), a viscous–inviscid flow interaction method coupled with an integral boundary layer solver (V-II, [30]) is used.

A. Design of a Transonic Compressor Cascade using Multilevel Evaluation

This two–objective constrained optimization problem is concerned with the design of a $2D$ transonic compressor cascade for min. total pressure losses ($\omega{=}(p_{t1}{-}p_{t2})/.5\rho_1{V_1}^2$), objective 1) and max. static pressure rise (p_2/p_1, objective 2). Indices 1 and 2 denote that this quantity measured at the inlet or outlet, respectively. The flow conditions were: isentropic exit Mach number $M_{2,is}=0.6$, inlet angle $a_1=55.4^\circ$ and Reynolds number based on the chord length $Re{=}170000$.

The cascade airfoil was parameterized using two Bézier curves, separately for the pressure and suction sides; the control polygon of each side was formed by 4 control points. All but the leading and trailing edge control points were allowed to vary in both chordwise and normal–to–the–chord direction, summing up to 8 design variables. Though it seems that this parameterization is "poor", it proved to be quite sufficient for such a transonic airfoil.

The imposed constraints restrict the airfoil thickness at various chordwise positions using an external penalty function method. The three constraints were:

$$T(10\%){\geq}3.0\%,\ T(50\%){\geq}2.0\%,\ T(90\%){\geq}0.4\%$$

where $T(x)$ stands for the ratio of the airfoil thickness over the chord length at $x\%$ of the chord. The exit flow angle was not constrained, since this was "controled" by the second objective.

The multilevel evaluation method (configured as in table I) was used, based on the N/S (high level model) and V-II (low level model) codes. At both levels, search was based on a MAEA implementing the SPEA2 [31] technique. The use of RBF networks as metamodels in the multi–objective problem was in accordance to the techniques presented in [27]. The two flow solvers have an average CPU cost ratio of approximately $30:1$. So, at any synchronization of the multilevel algorithm (such as a migration phase), after k_1 evaluations based on N/S and k_2 on V-II, the average CPU cost was approximately equal to $k_1{+}k_2/30$ calls to the N/S software (the cost of a N/S run will be referred to as one "evaluation cost unit"). According to table I, the maximum number of entries to the Pareto fronts of the two levels was set to 35 (elite population size). If, at any generation or level, more than 35 non–dominated solutions were found, thinning was employed. In table I, all migration frequencies are measured in generations. So, for instance, the first migration took place at the end of the $20th$ low level generation and that of the $2nd$ high level generation (synchronization barrier). 15 elite individuals at most migrated to the high level. These are selected from the current front of non–dominated solutions, provided that this contains at least 15 solutions; otherwise, the entire current front (with less than 15 members) migrates to the high level. At the same time, the low level receives 5 high level elites at most (same procedure). The three demes at the low level were arranged on a ring, with clockwise (intra–level) exchange of elite individuals. With the exception of the first two generations, at each level, the offspring population size determines the number of local metamodels to be trained and only a few (re)evaluations (the user defined lower and upper bounds) were carried out.

This run was made three times using different seeds in the random number generator. Below we describe one of these runs which seems to have an "average" performance. The optimization code was allowed to evolve for a total of 311 total evaluation cost units. This number reflects an evolution of 209 and 51 generations for the low and high level, respectively or, equivalently, 1778 calls to the V–II code and

TABLE I

MULTILEVEL EVALUATION ALGORITHM SETTINGS

	High Level	Low Level
Number of Demes	1	3
Coding	Binary Gray	Binary Gray
Offspring population size	20	60
Parent population size	5	15
Elite population size	35	35
Intra–level migration frequency	-	4
Gen. of first inter–level migration	2	20
Elites imported on 1st migration	15	5
Inter–level migration frequency	5	20
Elites imported otherwise.	6	5
Minimum DB size for IPE	130	150
Exact evaluations per gen. (IPE)	1-3	3-9

249 calls to the N/S code. Upon convergence, the high level Pareto front consists of 13 individuals and is illustrated in fig.2. In fig.3 the contour alongside with the Mach number fields over three of the Pareto front cascades are presented. A total of 10 inter–level migrations were carried out. During all of them, the low level passed on useful information to the single high level deme. In the overall cost given above, the cost for re–evaluating all migrated individuals with the evaluation software of the destination level, is included.

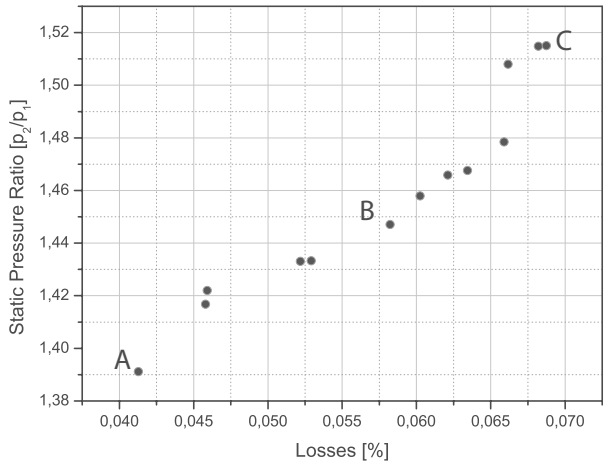

Fig. 2. Design of a transonic compressor cascade using Multilevel Evaluation: The Pareto front obtained after 311 evaluation cost units.

B. Inverse Design of an Isolated Airfoil using Multilevel Search

This case is concerned with the inverse design of an isolated airfoil with free–stream Mach number equal to M_∞=0.5 and angle of attack equal to a_∞=3.0°. The target was to reproduce a given pressure distribution along the airfoil contour which was parameterized using Bézier curves with 7 control points on each side. Three of them (per side) were allowed to vary, summing up to 12 design variables. No constraints were imposed. The multilevel optimization was set up using two levels. At the low level a MAEA was used in order to pinpoint the promising areas on the design space

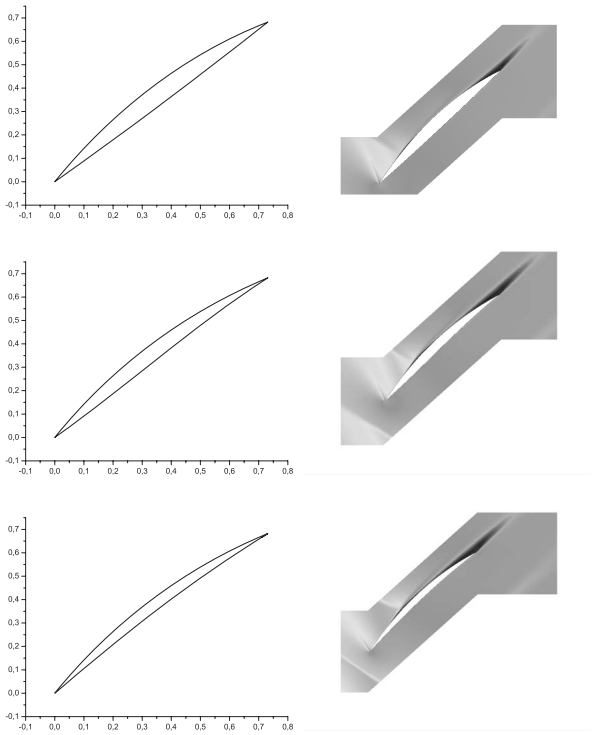

Fig. 3. Design of a transonic compressor cascade using Multilevel Evaluation: Indicative Pareto front members from fig.2, corresponding to members A, B and C (top to bottom). For each member the corresponding airfoil contour and Mach number field is illustrated.

and guide the high level search which employed the trust region SQP technique. The configuration is summarized in table II. The offspring population entry in the aforementioned table for level 1 (SQP level) denotes the number of design vectors updated using the SQP method. At both levels, the N/S solver was used for inviscid flows (Euler equations). The derivatives required for the SQP method were computed using the adjoint method, [24]. The cost of solving the adjoint equations was approximately equal to the cost of solving the flow equations. So, we assume that each high level evaluation (one call to the flow solver and the solution of the adjoint equations) costs one evaluation cost unit whereas each evaluation at the low level costs .5 units. The overall cost of the optimization was 650 evaluation cost units, or 300 high level evaluations and 700 low level ones. The two levels exchanged their elite individuals four times using the inter–level migration agent. It was, however, only during the first and third inter–level migrations that the best individual from the low level outperformed the current best individual of the high level (the stopping criterion for the low level optimization was set to two ineffective migrations). Fig.4 illustrates the SQP convergence of two immigrants, one of the first and the other of the third inter–level migration. The first one achieves a very good cost reduction (by almost two orders of magnitude, $f_{initial} = 5.44 \cdot 10^{-4}, f_{final} = 7.41 \cdot 10^{-6}$) while the second results to a reduction of about half an order of magnitude ($f_{initial} = 4.86 \cdot 10^{-6}, f_{final} = 1.30 \cdot 10^{-6}$), for it already lies very close to the optimal

solution; as in the previous case the run was performed using three different initial seedings for the random number generator and the "average performance" run corresponds to the presented results. The convergence of the multilevel algorithm is shown in fig.5 along with the convergence of a MAEA using the low–level configuration. The pressure coefficient of the final and reference (the one used to compute the target pressure distribution) airfoil are illustrated in fig. 6.

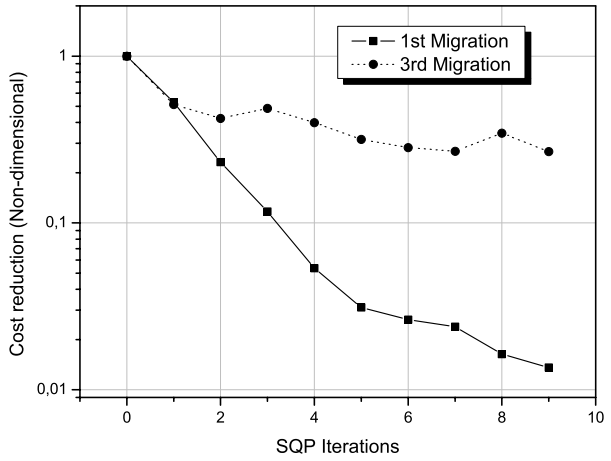

Fig. 4. Inverse design of an isolated airfoil using Multilevel Search: SQP convergence of two selected individuals during the first and third inter–level migrations.

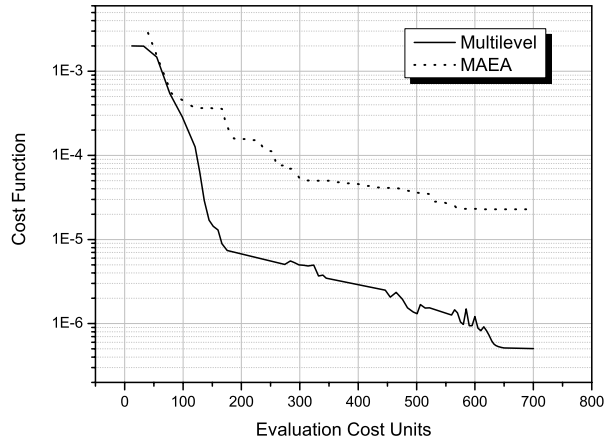

Fig. 5. Inverse design of an isolated airfoil using Multilevel Search: Convergence of the high level algorithm (trust region SQP) and comparisson with an "averaged" conventional MAEA.

C. Design of a Compressor Cascade using Multilevel Parameterization

This case is concerned with the design of the blade airfoil of a $2D$ axial compressor stator cascade, for minimum total pressure losses (minimum ω, at the following flow conditions: $\mathrm{Re}=4.0\cdot10^5$, $\mathrm{M}_1=0.6$ and $\alpha_1=53.6°$. The stagger angle was kept constant and equal to 30°. The CFD analysis software was the V-II code and the search tool was a MAEA,

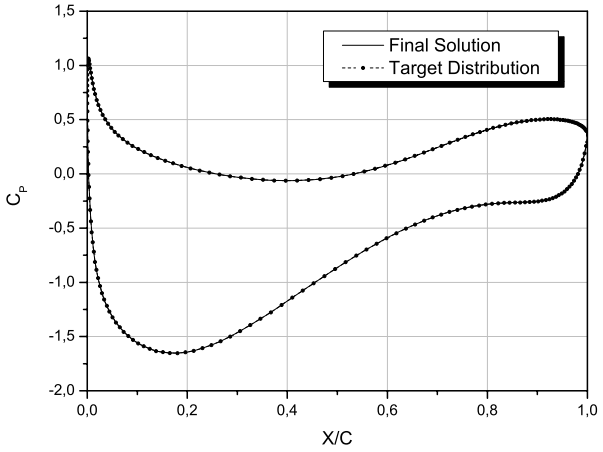

Fig. 6. Inverse design of an isolated airfoil using Multilevel Search: Pressure coefficient (C_P) of the target airfoil and final solution obtained from the Multilevel algorithm.

TABLE II
MULTILEVEL SEARCH ALGORITHM SETTINGS

	High Level	Low Level
Number of Demes	-	2
Coding	-	Binary Gray
Offspring population size	5	40
Parent population size	-	10
Elite population size	2	10
Intra–level migration frequency	-	4
Gen. of first inter–level migration	2	1
Elites imported on 1st migration	5	10
Inter–level migration frequency	10	30
Elites imported otherwise	5	4
Steepest descent step	0.001	-
SQP Trust Region(min)	3%	-
SQP Trust Region(max)	15%	-
SQP Trust Region(initial)	5%	-
Minimum DB size for IPE	-	150
Exact evaluations per gen. (IPE)	-	2-10

at both optimization levels. The multilevel configuration is given in table III. A number of constraints were imposed regarding the blade thickness at three chordwise locations and the flow exit angle, using an external penalty function method. The constraints are listed below:

$$T(10\%)\geq5.5\%,\ T(50\%)\geq7.5\%,\ T(90\%)\geq2.0\%,\ \alpha_2\leq30°$$

Two Bézier curves were used to parameterize the airfoil sides, each one with 7 (low level) or 17 (high level) control points. Transformations between the two parameterizations were made according to equation 1.

Two inter–level migrations proved to be sufficient. The low level elites transferred useful information to the high level only during the first migration cycle. At the end of the first migration, the lower and upper bounds of the design variables, at the high level, had to be adapted to the immigrants. With the new individuals, the high level MAEA rapidly approached the area around the global optimum. As a consequence, during the second migration, the best solution

TABLE III

MULTILEVEL PARAMETERIZATION ALGORITHM SETTINGS

	High Level	Low Level
Number of Demes	1	2
Coding	Binary	Binary
Offspring population size	30	40
Parent population size	6	10
Elite population size	5	10
Intra–level migration frequency	-	4
Gen. of first inter–level migration	1	12
Elites imported on 1st migration	10	5
Inter–level migration frequency	6	15
Elites imported otherwise.	4	5
Minimum DB size for IPE	150	150
Exact evaluations per gen. (IPE)	1-2	2-4

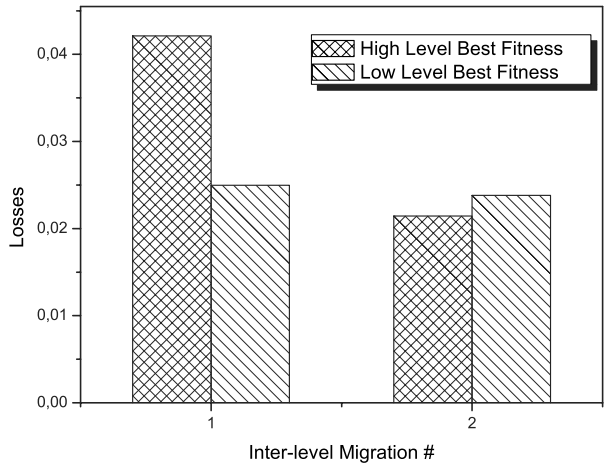

Fig. 7. Design of a compressor cascade using Multilevel Parameterization: Statistics on the quality of immigrants, during the inter–level migrations.

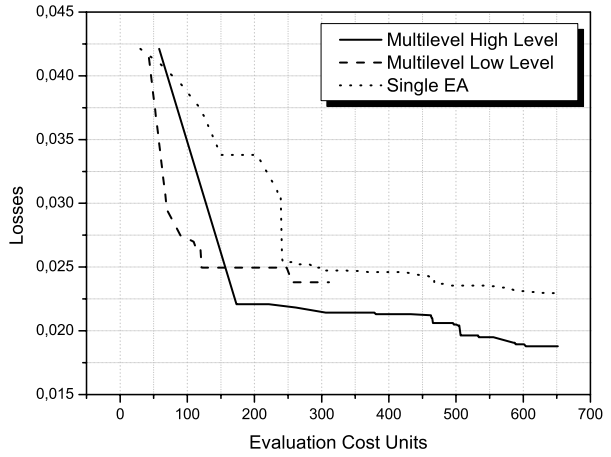

Fig. 8. Design of a compressor cascade using Multilevel Parameterization: Convergence of the multilevel algorithm.

at the high level was very good and the low level could not provide a better individual. Since the termination criterion for the inter–level migration was set to one unsuccessful migration, the second migration caused the lower level search to terminate (fig. 7). The aforementioned strict termination criterion was imposed on purpose since the evaluation cost at each level was the same (same code). We recall that the low level tool approaches faster the optimal solution not because of the lower cost of the corresponding evaluation model but due to the lower dimension of the search space; for the same reason, metamodels performed better, too. Fig.8 illustrates the convergence plot of the multilevel algorithm. The final solution satisfies all the imposed constraints and yields a cascade with $\omega = 1.88\%$.

Finally, the design process was repeated using a single–level MAEA (configured exactly as the high level MAEA). An average convergence of this algorithm (three runs) is presented in fig.8, too. The multilevel algorithm outperforms the single–level MAEA. It is important to stress that the coarse parameterization was unable to locate the final solution (the run with the small number of control points was repeated up to the end, i.e. without the termination criterion imposed by the inter–level migration algorithm, and could only slightly improve the best low level solution shown in fig.8; the new convergence is shown in the figure), due to the coarse airfoil parameterization. In contrast, the low level greatly assisted the search at the high level, even after a single migration.

V. CONCLUSIONS & ON–GOING RESEARCH

In this paper a general–purpose multilevel optimization platform was presented. At each level, the search for the optimal solution can be based on different evaluation tools, search algorithms or design variable sets. These three techniques have been used separately, although it is straightforward to use them in combination. The performance of their combined use is actually under investigation.

One– and two–objective problems have been examined. It has been shown that all three multilevel techniques yield optimal solutions on a limited computational budget. They use the low level to explore the search space at low computational cost and restrict the use of the computationally expensive high level only for the refinement of promising solutions.

Although parallelization issues are beyond the scope of this paper, all runs have been performed on a heterogeneous PC cluster. A central manager allows the optimal use of the cluster, by assigning each evaluation request to the appropriate computational resource through user–defined specifications (i.e. minimum required memory).

ACKNOWLEDGMENT

The authors would like to thank Dr. M.K. Karakasis for his contribution.

The project was co-funded by the European Social Fund (75%) and National Resources (25%) —Operational Program for Educational and Vocational Training II (EPEAEK II) and, particularly, Program PYTHAGORAS II.

REFERENCES

[1] K. C. Giannakoglou, "Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence," *Progress in Aerospace Sciences*, vol. 38, no. 1, pp. 43–76, 2002.

[2] S. Pierret and R. Van den Braembussche, "Three-dimensional turbine blade design using a Navier-Stokes solver and artificial neural network," *IMechE C557/154/99*, 1998.

[3] N. Papila, W. Shyy, N. Fitz-Coy, and R. T. Haftka, "Assessment of neural net and polynomial-based techniques for aerodynamic applications," in *17th AIAA Applied Aerodynamics Conference*. Norfolk, VA, USA: AIAA-1999-3167, 1999.

[4] K. C. Giannakoglou, A. P. Giotis, and M. K. Karakasis, "Low-cost genetic optimization based on inexact pre-evaluations and the sensitivity analysis of design parameters," *Journal of Inverse Problems in Engineering*, vol. 9, no. 4, pp. 389–412, 2001.

[5] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, and K. Giannakoglou, "Metamodel-assisted evolution strategies," in *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature — PPSN VII*, ser. Lecture Notes in Computer Science, vol. 2439. Springer-Verlag, 2002, pp. 361–370.

[6] D. Büche, N. Schraudolph, and P. Koumoutsakos, "Accelerating evolutionary algorithms with gaussian process fitness function models," *IEEE Transactions on Systems, Man, and Cybernetics — Part C: Applications and Reviews*, vol. 35, no. 2, pp. 183–194, May 2005.

[7] H. Ulmer, F. Streichert, and A. Zell, "Evolution strategies assisted by gaussian processes with improved pre-selection criterion," in *Proceedings of the 2003 Congress on Evolutionary Computation (CEC '03)*, vol. 1, Canberra, Australia, 2003, pp. 692–699.

[8] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing Journal – A Fusion of Foundations, Methodologies and Applications*, vol. 9, no. 1, pp. 3–12, 2005.

[9] Y. S. Ong, P. B. Nair, and A. J. Keane, "Evolutionary optimization of computationally expensive problems via surrogate modeling," *AIAA Journal*, vol. 41, no. 4, pp. 687–696, Apr. 2003.

[10] K. Rasheed, S. Vattam, and X. Ni, "Comparison of methods for developing dynamic reduced models for design optimization," In Proceedings of the Congress on Evolutionary Computation (CEC'2002), pp. 390395, 2002.

[11] M. Reyes-Sierra and C. Coello, "A study of fitness inheritance and approximation techniques for multi-objective particle swarm optimization," in *The 2005 IEEE Congress on Evolutionary Computation*, vol. 1, 2005, pp. 65–72.

[12] E. Cantu-Paz, "A survey of parallel genetic algorithms," *Calculateurs Paralleles, Reseaux et Systemes Repartis*, vol. 10, no. 2, pp. 141–171, 1998.

[13] M. Nowostawski and R. Poli, "Parallel genetic algorithm taxonomy," Proc. of the Third International Conference on Knowledge-based Intelligent Information Engineering Systems KES'99, pages 88-92. Aug. 1999., 1999.

[14] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, Oct. 2002.

[15] D. J. Doorly, J. Peiró, and S. Spooner, "Design optimisation using distributed evolutionary methods," in *37th Aerospace Sciences Meeting and Exhibit*. Reno, NV, USA: AIAA-1999-111, Jan. 1999.

[16] M. K. Karakasis and K. C. Giannakoglou, "Inexact information aided, low-cost, distributed genetic algorithms for aerodynamic shape optimization," *International Journal for Numerical Methods in Fluids*, vol. 43, no. 10–11, pp. 1149–1166, 2003.

[17] F. Herrera, M. Lozano, and C. Moraga, "Hierarchical distributed genetic algorithms," *International Journal of Intelligent Systems*, vol. 14, no. 9, pp. 1099–1121, 1999.

[18] M. K. Karakasis, D. G. Koubogiannis, and K. C. Giannakoglou, "Hierarchical distributed evolutionary algorithms in shape optimization," *International Journal for Numerical Methods in Fluids*, vol. 53, pp. 455–469, 2007.

[19] D. Quagliarella and A. Vicini, "Coupling genetic algorithms and gradient based optimization techniques," in *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science — Recent Advances and Industrial Applications*. John Wiley & Sons Ltd., Nov. 1997, pp. 289–309.

[20] C. Poloni, A. Giurgevich, L. Onesti, and V. Pediroda, "Hybridization of a multiobjective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2, pp. 403–420, 2000.

[21] Y. S. Ong, K. Y. Lum, P. B. Nair, D. M. Shi, and Z. K. Zhang, "Global convergence of unconstrained and bound constrained surrogate-assisted evolutionary search in aerodynamic shape design," in *Proceedings of the 2003 Congress on Evolutionary Computation (CEC '03)*, vol. 3, Canberra, Australia, Dec. 2003, pp. 1856–1863.

[22] F. Muyl, L. Dumas, and V. Herbert, "Hybrid method for aerodynamic shape optimization in automotive industry," *Journal of Computers and Fluids*, vol. 33, no. 5-6, pp. 849–858, 2004.

[23] J. Knowles, "Local-search and hybrid evolutionary algorithms for pareto optimization," PhD thesis, Department of Computer Science, University of Reading, UK, 2002.

[24] D. Papadimitriou and K. Giannakoglou, "A continuous adjoint method with objective function derivatives based on boundary integrals for inviscid and viscous flows," *Journal of Computers and Fluids*, vol. 36, no. 2, pp. 325–341, 2007.

[25] L. Piegl and W. Tiller, *The NURBS Book*, 2nd ed. Monographs in Visual Communication. Springer-Verlag, Germany, 1997.

[26] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. New Jersey, USA: Prentice Hall, 1999.

[27] M. K. Karakasis and K. C. Giannakoglou, "On the use of metamodel-assisted, multi-objective evolutionary algorithms," *Engineering Optimization*, vol. 38, no. 8, pp. 941–957, 2005.

[28] N. Lambropoulos, D. Koubogiannis, and K. Giannakoglou, "Acceleration of a Navier-Stokes equation solver for unstructured grids using agglomeration multigrid and parallel processing," *Computer Methods in Applied Mechanics and Engineering*, vol. 193, pp. 781–803, 2004.

[29] P. Spalart and S. Allmaras, "A one-equation turbulence model for aerodynamic flows," AIAA Paper 92-0439, 1992.

[30] M. Drela and M. B. Giles, "Viscous-inviscid analysis of transonic and low Reynolds number airfoils," *AIAA Journal*, vol. 25, no. 10, pp. 1347–1355, Oct. 1987.

[31] E. Zitzler, M. Laumans, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Eurogen 2001, Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems*. Barcelona: CIMNE, 2002, pp. 19–26.