Aerodynamic Design of Compressor Airfoils using Hierarchical, Distributed, Metamodel-Assisted Evolutionary Algorithms

M. K. Karakasis, K. C. Giannakoglou, and D. G. Koubogiannis[*]

Laboratory of Thermal Turbomachines, National Technical University of Athens,
P.O. Box: 64069, Athens 157 10, Greece, e-mail: kgianna@central.ntua.gr

ABSTRACT

This paper presents state-of-the-art techniques which enhance noticeably the efficiency of evolutionary algorithms in aerodynamic shape optimization and, in particular, in turbomachinery blade airfoil design problems. These techniques rely upon the combined use of *hierarchical* (more than one levels of search, using evaluation tools of different modeling accuracy and CPU cost), *distributed* (simultaneously evolving population subsets, allowing the regular exchange of promising solutions between them), *metamodel-assisted* (many non-promising individuals, generated during the evolution, are filtered out during the inexact pre-evaluation phase, using on-line trained artificial neural networks) *evolutionary algorithms*. Their combination in a single search method, abbreviated to HDMAEA or HD(EA-IPE) and ported on a multiprocessor computing platform, is presented. This method is applied to the design of the stator airfoil of a highly-loaded compressor cascade. A flow turning of 45° at transonic flow conditions is achieved with markedly low total pressure losses.

INTRODUCTION

In industrial design-optimization problems, particularly those requiring computationally demanding evaluation software, such as Navier-Stokes equations solvers, the advantage of evolutionary algorithms (EAs) in escaping local optima often becomes overshadowed by the excessive evaluations required to reach the optimal solution(s). To alleviate this problem, several remedies have been proposed. These can be used separately from each other or, as this paper presents, in combination so as to maximize the reduction in CPU cost and/or wall clock time of the overall optimization task. Four of these techniques, all of them being used in this paper, are summarized below:

Metamodel-Assisted Evolutionary Algorithms (MAEAs): Metamodels are surrogate evaluation models which are trained on previously evaluated candidate solutions to the problem. They are used within the EA to get rid of as many as possible calls to the exact and costly evaluation tool. Polynomial-based response surfaces (Engelund et al., 1993), statistical methods (Ratle, 1999), artificial neural networks (Pierret and Van den Braembussche, 1999; Giotis et al., 2000) are among the most frequently used metamodels. In the literature, MAEAs in which the metamodel is set up prior to launching the EA (Papila et al., 1999) can be found. Closer interaction between the EA and the metamodel is achieved by using on-line trained metamodels (Giannakoglou, 1999, 2002; Emmerich et al., 2006). Since the latter are used for the Inexact Pre-Evaluation of each generation members, this approach will be referred to as EA-IPE; in this paper, terms MAEA and EA-IPE are used interchangeably. Note that particular attention should be paid to the use of MAEAs in multi-objective optimization problems, as discussed in Karakasis and Giannakoglou (2006).

Distributed EAs (DEAs): DEAs rely upon the evolution of a number of small sized sub-populations or demes, rather than a single population. Optimal performance can be achieved by associating differently tuned EAs, i.e. EAs with different exploitation and exploration mechanisms, with

[*]D. G. Koubogiannis is actually with the Technological Educational Institute of Athens.

each deme. Demes evolve concurrently and communicate regularly through synchronous or asynchronous migrations of promising individuals (Tanese, 1989), according to various communication schemes.

Hierarchical EAs (HEAs): An hierarchical EA establishes a multilevel search mechanism with different evaluation tools at each level. By convention, the highest accuracy and CPU cost evaluation software is associated with the highest level which is responsible for delivering the final optimal solution. At the lower levels, cheaper evaluation tools based on simplified models are used instead; the lower levels undertake the exploration of the search space and forward promising individuals to the highest level. Although MAEAs behave as a two-level HEAs, we refrain from classifying them as such, since the low-level evaluation tool (metamodel) is not a flow solver, to speak in terms of aerodynamics.

Parallelized EAs: The inherent ability of EAs to allow the concurrent evaluation of their population members on multiprocessor platforms reduces the wall clock time of the overall optimization.

The combination of some or all of these enhancement techniques within a single EA was the subject of previous works by the same group. In Karakasis et al. (2003), the distributed variant of EA-IPE was proposed; multiple EAs, each employing IPE to filter out badly performing individuals, were combined to form the demes of a distributed EA. In Karakasis et al. (2007), the hierarchical distributed EA-IPE algorithm was first proposed and tested in internal and external aerodynamics.

The herein described HDMAEA scheme includes two levels of search. A Navier-Stokes time-marching solver for unstructured grids (*high level*) and an integral boundary layer method coupled with an external flow solver (*low level*) are used. A radial basis function (RBF) network acts as metamodel. It is used in the local sense, i.e. it is trained on paired input-outputs collected during the evolution. The training and use of metamodels at each level is carried out separately from the other level. This means that previous evaluation results are kept in separate databases (DBs) at each level. The two-level search tool was used for the design of the stator blade airfoil of a highly-loaded compressor so as to give minimum total pressure losses.

Just a few previous works with similarities to the paper in hand can be reported: the simpler multilevel approach proposed by Sefrioui and Périaux (2000) and the finite-element method based flywheel design optimization by Eby et al. (1998), which was based on an island genetic algorithm with different levels of resolution per island. Note, however, that the present method additionally employs metamodel-assisted search (the IPE screening) at each level.

THE HD(EA-IPE) SCHEME

The hierarchical optimization algorithm, figure 1, consists of two levels of distributed MAEAs. At each level, the flow is numerically solved using a different tool, with different accuracy and computational cost. In particular:

- At the high level, a Navier-Stokes (N-S) equation solver based on a time-marching, vertex-centered finite volume method on unstructured grids with triangular elements is used (Koubogiannis et al., 2003). The quasi-3D equations, with variable streamtube thickness in the streamwise direction, are solved. The convection terms are discretized by means of Roe's flux difference splitting, while for the diffusion terms the assumption of linearly distributed primitive variables within each triangular element is made. Second order accuracy in space is obtained through MUSCL extrapolation, while monotonicity is guaranteed by means of appropriate limiters. The Shear Stress Transport (SST) variant of the blended $k-\omega$ turbulence model (Menter, 1993) is used. The role of the high level is to scrutinize design subspaces pinpointed as promising by the low level search and capture flow features (such as flow separation) which cannot be

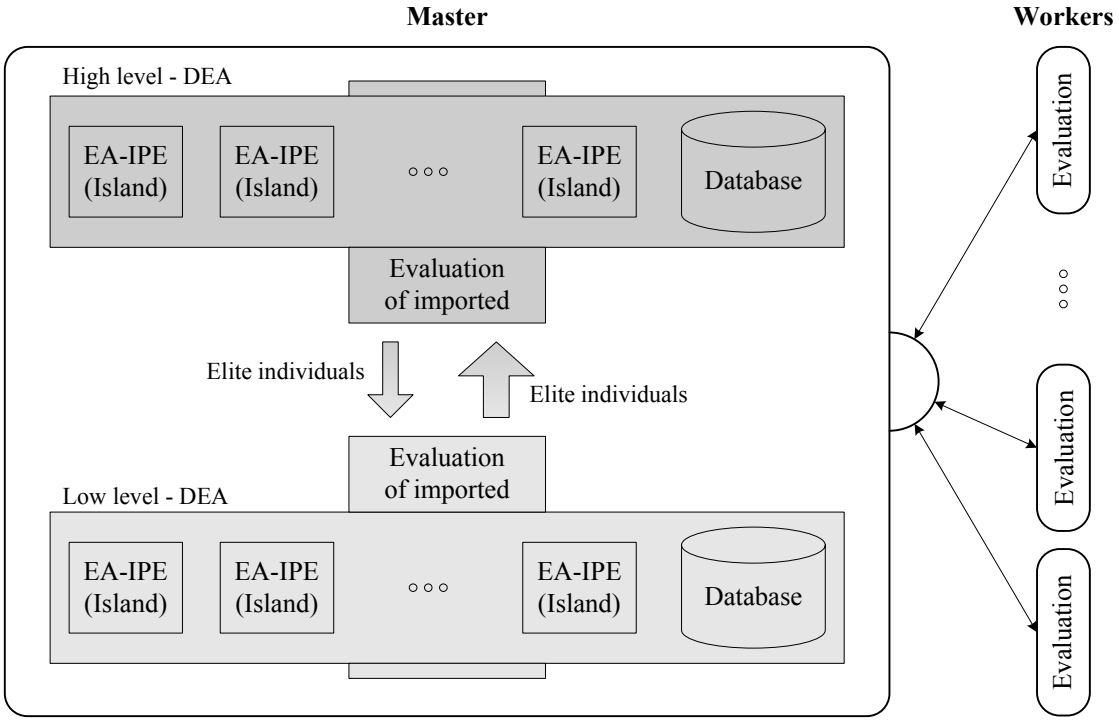


Figure 1: A two-level Hierarchical Distributed Evolutionary Algorithm implemented on parallel computing systems. Each level is associated with its own evaluation model, database and metamodel. The metamodels are used according to the IPE concept, as described in the text.

modeled by the low level tool. It is obvious that the optimal solution can be delivered exclusively by the high level.

- At the low level, a viscous-inviscid flow interaction tool is used. In particular, the MSES/MISES code for external/internal aerodynamics (Giles and Drela, 1987; Drela and Giles, 1987) is employed. The Euler equations are discretized on a conservative streamline grid and coupled to a two-equation integral boundary-layer method. Transition prediction is incorporated into the viscous flow model. The entire discrete equation set is solved as a fully coupled nonlinear system of equations, resulting to a fast code, unless strong flow separation exists. Strong viscous-inviscid interactions and limited flow separation are successfully dealt with. The role of the low level is to explore the design space at low CPU cost and, hence, assist the high level EA by providing promising solutions to it.

The distributed search within each level and the intra-level exchange of information, figure 1, looks like a normal D(EA-IPE) algorithm (Karakasis et al., 2003). The new feature in HD(EA-IPE) is the inter-level communication. A level agent coordinates the communication between demes (*intra-level communication*), gathers the elite individuals from all of them and, also, distributes the individuals imported from the adjacent level (*inter-level communication*) to the demes. In particular, after a predefined number of migrations between the level's demes, the best individuals from all of them are gathered and made available to the adjacent level. In return, the latter is requested to provide its own elite members. Before any level of the HD(EA-IPE) incorporates immigrants originating from an adjacent level, these individuals must be evaluated with the level's flow analysis tool. It is important not to merge objective function values computed using different evaluation tools, otherwise the selection process might be misled. At the high level, an incoming individual from the low level replaces an existing individual in a deme, only if it performs better than that. If the low level consecutively fails

to provide the high one with useful individuals, its evolution is terminated. As soon as the inter-level communication has been accomplished, the demes' evolution resumes.

In figure 1, each box marked with *EA-IPE (Island)* represents the standard EA-IPE algorithm, with μ parents and λ offspring. Its basic steps are described below:

Algorithm IPE (Inexact Pre-Evaluation). λ individuals are generated at random. A conventional EA (exact evaluations, no metamodels) runs for a few generations (*CPU demanding*). By doing so, the DB is populated by the minimum number of entries needed for the metamodel training.

IPE1. [Offspring evaluation] Once the DB contains a sufficient number of entries, IPE starts according to the following steps:

IPE1a. [Inexact Pre-Evaluation] For each new population member, a local metamodel (RBFN) is trained using the closest DB entries. The number of training patterns is not constant but depends on their distribution in the design space (Giannakoglou et al., 2001; Karakasis and Giannakoglou, 2004). The trained metamodel approximates the objective function value for each individual.

IPE1b. [Selection for re-evaluation] Based on the objective function values provided by the metamodel, the λ_e (usually $\lambda_e \simeq 0.1\lambda$) most promising individuals are singled out.

IPE1c. [Exact evaluation] The λ_e selected individuals are exactly re-evaluated (*CPU demanding*) and stored in the DB.

IPE2. [Fitness assignment] A fitness or cost value is assigned to each and every population member. As such, the outcome of the approximate metamodel-based evaluation or, of course, that of the exact evaluation is used.

IPE3. [Evolution operators] Parent selection, recombination and mutation of resulting offspring are performed.

IPE4. [Termination] The algorithm terminates, if a criterion based on either the maximum number of exact evaluations or that of idle generations is satisfied; otherwise, the evolution continues from step IPE1a. ∎

In the previous algorithm, the two tasks that are *CPU demanding* are clearly marked. Within each level, the local metamodels are trained on the outcomes of evaluations carried out using the level's flow analysis tool. Working with DEAs, the IPE algorithm is applied to each deme, in the same sense as previously described for the case of a single population (Karakasis et al., 2003).

In this work, RBFNs are used as metamodels. They consist of a single hidden-neuron layer to perform a two-stage mapping: a nonlinear one from the design to the hidden layer's space followed by a linear mapping to the objective space. With appropriate selection of the RBF centers, the simplicity of their architecture reduces their training to the solution of a linear algebraic system of equations or a linear least-squares problem (Poggio and Girosi, 1990). RBFNs possess valuable properties for function approximation. Since these are trained on a small subset of the DB entries (local metamodels), their training cost is negligible compared even to the cost of an integral boundary layer method. The training patterns can either be interpolated or approximated, depending on the required network's generalization capability (Karakasis and Giannakoglou, 2004).

The HD(EA-IPE) algorithm is implemented so as to take advantage of a multiprocessor system. The requests for evaluation with a specific flow analysis tool are addressed from all levels to a single evaluation server, which assigns accordingly the available computing resources. The evaluation results are stored in appropriate DBs, one for each level. The use of a multiprocessor system by the two-level HD(EA-IPE) is also schematically shown in figure 1.

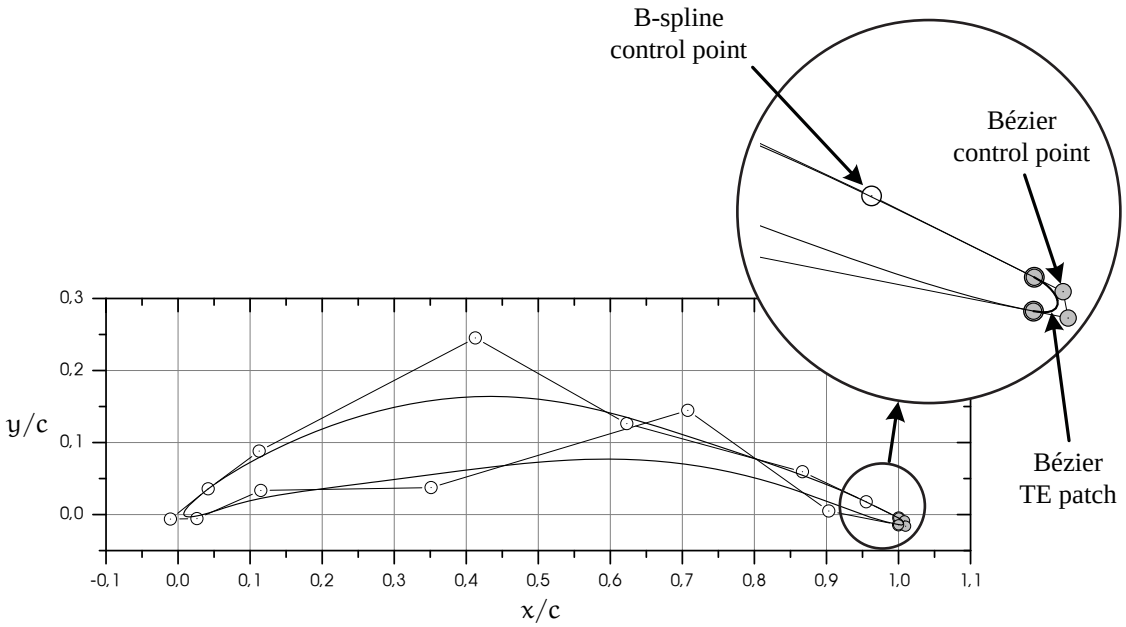


Figure 2: Parameterization of the optimal profile with a B-spline curve and a 4-control point Bézier patch curve at the trailing edge (TE) region.

DESIGN OF A HIGHLY-LOADED COMPRESSOR CASCADE

The HD(EA-IPE) method is used for the design of a highly-loaded compressor airfoil, with minimum total pressure losses at given flow conditions. The flow conditions are: isentropic exit Mach number $M_{2,is} = 0.496$, inlet flow angle $\alpha_1 = 45.0^\circ$, Reynolds number based on airfoil chord $Re = 8.41 \cdot 10^5$ and inlet turbulence intensity $\tau_u = 1.3\%$. The cascade has fixed axial velocity density ratio at the value $AVDR = 0.907$; this corresponds to constant streamtube thickness before the leading or after the trailing edge and a linear variation between them. It is desirable to have an inlet Mach number M_1 in the range between 0.70 and 0.75 and flow turning $\alpha_1 - \alpha_2$ of about 45° with fully attached flow. The last two requirements are imposed as constraints, as explained below.

The objective is to minimize the total pressure (p_t) loss coefficient

$$\omega = \frac{p_{t1} - p_{t2}}{p_{t1} - p_1},$$

subject to the following geometrical or flow-related constraints:

1. Concerning the airfoil thickness:
   - maximum thickness $t_{\max}$ should exceed $0.10c$ (c stands for the chord length),
   - minimum thickness t_{75} in the range $0.65 \leq x/c \leq 0.85$ should exceed $0.022c$ and
   - minimum thickness t_{90} in the range $0.85 \leq x/c \leq 0.95$ should exceed $0.017c$.
2. The maximum curvature $\varkappa_{\mathrm{LE}}$ at the airfoil leading edge should not exceed $3.5 \cdot 10^2 c$.
3. The exit flow angle must be $-0.5^\circ \leq \alpha_2 \leq 0.5^\circ$.
4. The flow must remain attached, thus $c_f \geq 0.0$ over the entire airfoil contour.

Note that the fourth constraint intends to strengthen the requirement for attached flow, although flow separation, if this occurs, affects the objective function, too. All the above constraints were imposed via exponential penalty function multipliers, increasing the value of the objective function. The meta-models approximate the penalized objective function.

The airfoil is parameterized using a single 4th-order B-spline curve based on 14 control points. The trailing edge region is parameterized by a Bézier curve patch with 4 control points (viz. a cubic polynomial, figure 2). The latter was automatically attached to the B-spline airfoil profile so as not

to introduce excessive design variables to model a "conventionally shaped" training edge. Design variables are the B-spline control point coordinates, the stagger angle γ and solidity s/c (i.e. $14 \cdot 2 + 2 = 30$ design variables). The latter are allowed to take on values in the range $\gamma \in [15^\circ, 35^\circ]$ and $s/c \in [0.60, 0.70]$, respectively.

Basic concern in such an optimization problem is to prevent the algorithm from being trapped into local minima, due to a premature homogenization of the EA population. To avoid this, a high number of demes is used at both levels of the HD(EA-IPE), although this might cause convergence deceleration. In particular, 5 demes with $\mu = \lambda = 40$ are used at the low and 3 demes $\mu = \lambda = 30$ at the high level. Within each deme, the $\lambda_e = 0.1\lambda$ individuals chosen during the IPE phase are exactly evaluated. The IPE filter initiates automatically once 100 exact evaluations have been performed; by doing so, adequate DB entries for the metamodel training are available. The main configuration of the inter-level communication is outlined below:

	High level	**Low level**
Migration frequency (total generations of all demes)	8	120
Elite individuals imported for the first time	30	6
Elite individuals imported otherwise	10	6
Exactly evaluated immigrants	50%	100%

The imported individuals are randomly distributed to the level's demes, where they replace their worst members. Especially, at the high level, the replacement occurs only if the imported individual outperforms the one to be replaced. To save evaluation time during inter-level communication, the imported individuals at the high level are also pre-evaluated by the metamodel and only the best half of them are exactly re-evaluated through the N-S solver.

In figure 3 the convergence of HD(EA-IPE) is compared to that of a conventional EA with a single population of $\mu = \lambda = 90$ individuals, using the N-S solver as evaluation tool. Even the first migration of individuals upwards (from the low to the high level) provides a solution which is by $\sim 25\%$ better than that of a conventional EA at just $1/5$ of the computational cost. Allowing further evolution of the HD(EA-IPE) scheme leads to optimal solution improved by $\sim 50\%$ compared to that of the conventional EA. The same figure shows that the additional use of IPE within a HDEA is advantageous; the corresponding convergence outperforms that of a HDEA with the same population size but without assistance by the metamodels.

In all previously compared algorithms, the parameterized shape of an industrial cascade airfoil, designed to operate at different conditions without respecting all of our constraints, has been injected into the starting population, on purpose. By doing so, one may follow the adaptation of an individual to its new environment. This initial cascade (a) and indicative intermediate designs during the evolution are presented in figure 4. It should be noted that airfoils (b) and (c) have practically been located by the low level tool. The final one, profile (d), was the outcome of search-evolution at the high level.

The optimal airfoil, along with the corresponding control points, has already been shown in figure 2. For this airfoil, the stagger angle is $\gamma = 15.4^\circ$ and the solidity $s/c = 0.699$. Figure 5 shows the computational grid (consisting of 13743 nodes and 26910 triangles) around the optimal cascade and the computed iso-Mach number contours. The isentropic Mach number distribution in the optimal cascade is shown in figure 6(a), indicating a shockless flow. In figure 6(b), the friction coefficient is shown, indicating that no separation occurs (at least according to the turbulence model used). The optimal cascade turns the flow to $\alpha_2 = 0.02^\circ$ with total pressure losses $\omega = 1.75\%$. Finally, all the geometrical constraints are respected and in particular $t_{\max} = 0.10c$, $t_{75} = 0.023c$, $t_{90} = 0.017c$ and $\varkappa_{\mathrm{LE}} = 3.3 \cdot 10^2 c$.

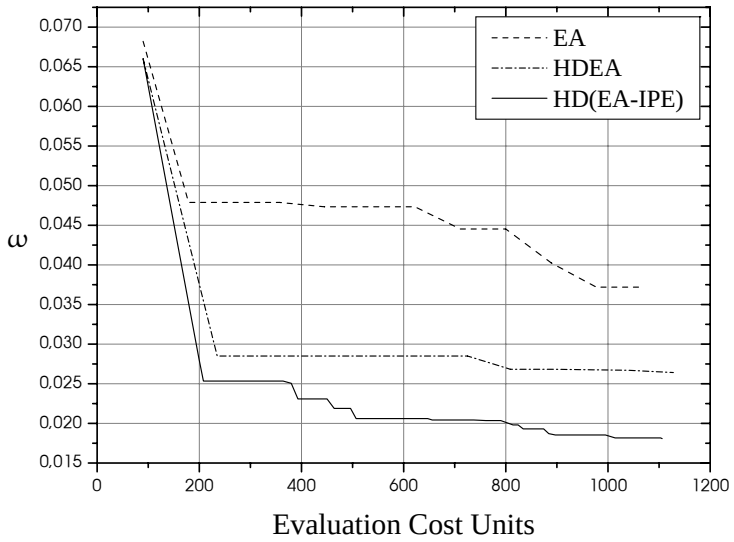
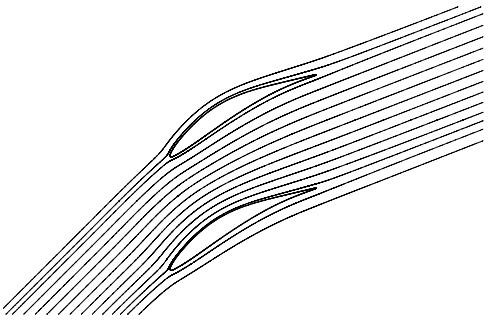
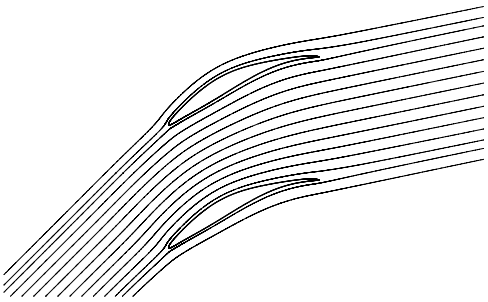


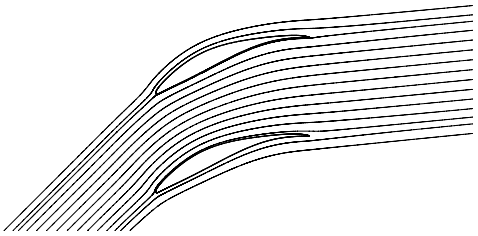
Figure 3: Convergence history. The total pressure loss coefficient, penalized to account for the violation of constraints (if any), of the best solution provided by the higher level is monitored in terms of CPU cost. Each cost unit is equivalent to the computational cost of solving the Navier-Stokes equations in a typical cascade geometry. The cost of the evolution of the lower level is also taken into account.



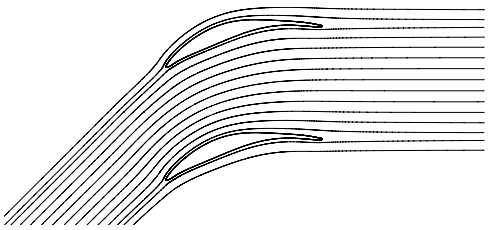
(a) Geometry: $s/c = 0.680$, $\gamma = 30.0^\circ$. Flow: $M_1 = 0.640$ and $M_2 = 0.487$, $\alpha_2 = 21.2^\circ$, $\omega = 2.45\%$.



(b) Geometry: $s/c = 0.748$, $\gamma = 25.0^\circ$. Flow: $M_1 = 0.708$ and $M_2 = 0.489$, $\alpha_2 = 11.6^\circ$, $\omega = 1.42\%$.



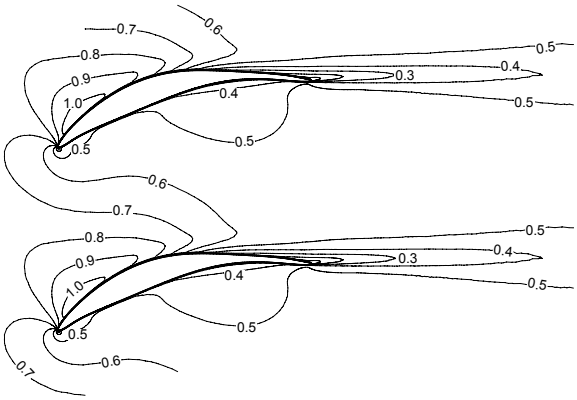
(c) Geometry: $s/c = 0.605$, $\gamma = 20.8^\circ$. Flow: $M_1 = 0.722$ and $M_2 = 0.487$, $\alpha_2 = 5.51^\circ$, $\omega = 2.15\%$.



(d) Geometry: $s/c = 0.699$, $\gamma = 15.4^\circ$. Flow: $M_1 = 0.732$ and $M_2 = 0.487$, $\alpha_2 = 0.02^\circ$, $\omega = 1.75\%$ (final).

Figure 4: Evolution of the optimal solution.

(a) Unstructured grid or low-Reynolds number turbulence model computations.

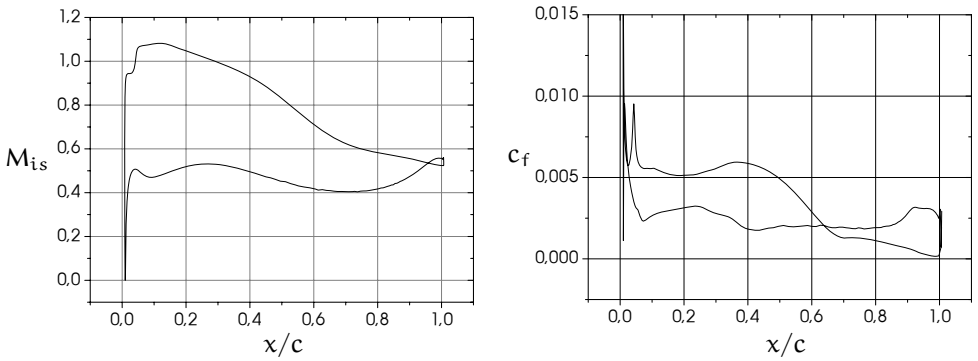


(b) Constant Mach number lines

Figure 5: Grid and iso-Mach lines in the optimal cascade.

CONCLUSIONS

The design of a highly-loaded optimal airfoil, with 45° turning at transonic flow conditions was made possible using a hierarchical distributed metamodel-assisted evolutionary algorithm. Several flow related or geometrical constraints have been imposed using penalty multipliers. With a two-level hierarchical search, using a viscous-inviscid flow interaction method for the exploration of the search space and a Navier-Stokes equation solver at the high level, the optimal solution can be found with a considerable economy in CPU cost, compared to EAs without or with just some of the aforementioned "enhancements". The use of metamodels within each level and their on-line training on the results of



(a) M_{is} distribution

(b) c_f distribution.

Figure 6: Isentropic Mach number and friction coefficient distributions of the optimal cascade flow, giving $M_2 = 0.487$, $\alpha_2 = 0.02^\circ$ and $\omega = 1.75\%$.

previous evaluations at the same level proved to be one of the keys for reducing the CPU cost.

The studies presented in this paper have been carried out using the optimization software **EASY v1.5** (**E**volutionary **A**lgorithms **SY**stem)) developed and brought to market by NTUA (EASY, 2006).

ACKNOWLEDGEMENTS

This work 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, the Program PYTHAGORAS II.

REFERENCES

M. Drela and M. B. Giles. Viscous-inviscid analysis of transonic and low Reynolds number airfoils. *AIAA Journal*, 25(10):1347–1355, October 1987.

EASY. Evolutionary Algorithms SYstem, 2006. URL http://velos0.ltt.mech.ntua.gr/EASY.

D. Eby, R. C. Averill, W. F. Punch III, and E. D. Goodman. Evaluation of injection island GA performance on flywheel design optimization. In *Proceedings of the 3rd Conference on Adaptive Computing in Design and Manufacturing*, pages 121–136, Plymouth, UK, 1998. Springer-Verlag.

M. Emmerich, K. Giannakoglou, and B. Naujoks. Single and multi-objective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, August 2006.

W. C. Engelund, D. O. Stanley, R. A. Lepsch, M. M. McMillian, and R. Unal. Aerodynamic configuration design using response surface methodology analysis. In *Aircraft Design, Systems and Operations Meeting*, pages 11–13, Monterey, CA, August 1993. AIAA-1993-3967.

K. C. Giannakoglou. Designing turbomachinery blades using evolutionary methods. In *ASME Turbo Expo '99*, Indianapolis, USA, June 1999. ASME Paper 99-GT-181.

K. C. Giannakoglou. Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. *Progress in Aerospace Sciences*, 38(1):43–76, 2002.

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*, 9(4):389–412, 2001.

M. B. Giles and M. Drela. A two-dimensional transonic aerodynamic design method. *AIAA Journal*, 25(9):1199–1206, September 1987.

A. P. Giotis, K. C. Giannakoglou, and J. Périaux. A reduced-cost multi-objective optimization method based on the Pareto front technique, neural networks and PVM. In E. Oñate, G. Bugeda, and B. Suárez, editors, *ECCOMAS 2000, Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering*, Barcelona, Spain, September 2000. CIMNE.

M. K. Karakasis and K. C. Giannakoglou. On the use of surrogate evaluation models in multi-objective evolutionary algorithms. In P. Neittaanmäki, T. Rossi, S. Korotov, E. Oñate, J. Périaux, and D. Knörzer, editors, *ECCOMAS 2004, Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering*, Jÿvaskÿla, Finland, July 2004.

M. K. Karakasis and K. C. Giannakoglou. On the use of metamodel-assisted, multi-objective evolutionary algorithms. *Engineering Optimization*, 38(8):941–957, December 2006.

M. K. Karakasis, A. P. Giotis, and K. C. Giannakoglou. Inexact information aided, low-cost, distributed genetic algorithms for aerodynamic shape optimization. *International Journal for Numerical Methods in Fluids*, 43(10–11):1149–1166, December 2003.

M. K. Karakasis, D. G. Koubogiannis, and K. C. Giannakoglou. Hierarchical distributed evolutionary algorithms in shape optimization. *International Journal for Numerical Methods in Fluids*, 53 (3):455–469, January 2007.

D. G. Koubogiannis, A. N. Athanassiadis, and K. C. Giannakoglou. One- and two-equation turbulence models for the prediction of complex cascade flows using unstructured grids. *Computers & Fluids*, 32(3):403–430, March 2003.

F. R. Menter. Zonal two equation $k-\omega$ turbulence models for aerodynamic flows. AIAA Paper 93-2906, 1993.

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, 1999. AIAA-1999-3167.

S. Pierret and R. A. Van den Braembussche. Turbomachinery blade design using a Navier-Stokes solver and artificial neural network. *Journal of Turbomachinery*, 121(2):326–332, April 1999. ASME Paper 99-GT-4.

T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9): 1481–1497, September 1990.

A. Ratle. Optimal sampling strategies for learning a fitness model. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 3, pages 2078–2085, Washington, DC, July 1999.

M. Sefrioui and J. Périaux. A hierarchical genetic algorithm using multiple models for optimization. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. M. Guervós, and H.-P. Schwefel, editors, *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature - PPSN VI*, volume 1917 of *Lecture Notes in Computer Science*, pages 879–888, Paris, France, September 2000. Springer-Verlag.

R. Tanese. Distributed genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 434–439, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.