ERCOFTAC 2006 Design Optimization International Conference,
April 5-7 2006, Gran Canaria, Spain

EVOLUTIONARY ALGORITHMS WITH SURROGATE MODELING FOR COMPUTATIONALLY EXPENSIVE OPTIMIZATION PROBLEMS

Kyriakos C. Giannakoglou*, Marios K. Karakasis*, Ioannis C. Kampolis*

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

Key words: Design, Optimization, Evolutionary Algorithms, Metamodels, Distributed and Hierarchical Search

Abstract. *During the last decade, the development of efficient optimization tools that utilize Evolutionary Algorithms (EAs) as the core search tool gained particular attention and reached a certain level of maturity. These tools enabled the extensive use of EAs in large-scale industrial applications, in which the analysis (evaluation) tool is computationally expensive. A literature survey reveals that the majority of new, promising variants of EAs are conceptually based on the reduction of the otherwise excessive number of calls to the evaluation software. This reduction is possible through the use of various techniques such as: (a) the use of computationally cheap surrogate evaluation models or metamodels trained on samples collected suring or separately from the evolution (Metamodel-Assisted EAs, MAEAs), (b) the use of more than one evaluation tools, with different approximation errors and computing cost, according to a hierachical structure (Hierarchical EAs, HEAs) and (c) the use of Distributed EAs (DEAs), which subdivide the entire population into concurrently evolving, semi-isolated subsets, which regularly exchange promising individuals. These techniques and the most efficient combination of all of them in a single search method (Hierachical, Distributed Metamodel-Assisted EAs, HDMAEAs), are discussed in this paper. Due to space limitations, only three applications are presented; however, more applications as well as technical details on the presented methods can be found in the cited papers by the authors.*

1 MAEAs and the Inexact Pre-Evaluation Technique

There is large literature on the development of metamodels for data interpolation or approximation and the use of metamodels in MAEAs. The popularity of EAs in large-scale engineering applications increased because of their capability to accommodate metamodels, which may considerably reduce their cost. The way metamodels can assist EAs in the search of optimal solutions is through acting as low CPU cost, surrogate evaluation models. The higher the CPU cost of a single evaluation, the higher the economy in CPU cost achieved by MAEAs, compared to conventional EAs.

A possible classification of MAEAs takes into account whether the metamodels used are trained separately, i.e. *off-line*, or during, i.e. *on-line*, the evolution. Note that this classification does not take into account the metamodel type; so, polynomial-based response surfaces, Gaussian random field models (such as kriging), artificial neural networks (such as multi-layer perceptrons or radial-basis function networks), etc. can be used.

EAs assisted by *off-line* trained metamodels rely on surrogates, built before launching the EA. They approximate the response of the exact evaluation software over the entire search space, thus they will be referred to as *global* metamodels. Once such a metamodel has been trained, it is used to evaluate candidate solutions generated by the EA [29, 17] and locate "optimal" solutions. These need to be evaluated again with the exact evaluation tool; should discrepancy exist between the fitness according to the model and the metamodel, the latter must be retrained by considering also the outcome of the additional exact evaluations [6, 30, 27, 5].

In contrast, EAs may be supported by *on-line* metamodel(s), trained or updated during the evolution. For their training, individuals already evaluated are utilized. Many algorithmic variants are possible. In [31, 20], after a user-defined number of generations, all population members are evaluated with the exact model, producing new data to update the metamodel used thus far. In [13, 28, 36], each population member is first evaluated using the metamodel and only the most promising among them are re-evaluated by the exact model. The last variant was first proposed by our research group, some years ago, and will be referred to as EAs incorporating the *Inexact Pre-Evaluation* (IPE) phase (EA-IPE).

EA-IPE uses the metamodels to screen the population members and restricts the number of exact evaluations per generation to potentially good members. An outline of the IPE algorithm, for the general multi-objective case, follows [15, 22, 23]. The reader should keep in mind that, in each generation g, the population $\mathcal{P}$ consists of the set of λ offspring $\mathcal{P}_\lambda$, that of μ parents $\mathcal{P}_\mu$ and the archive of elite individuals $\mathcal{P}_\alpha$. Thus, $\mathcal{P} = \mathcal{P}_\lambda \cup \mathcal{P}_\mu \cup \mathcal{P}_\alpha$.

IPE starts after running a conventional EA for a couple of generation (g_{start}), based on the exact evaluation model, so as to produce a number of entries (pairs $(\mathbf{x}, \mathbf{f}(\mathbf{x}))$) for the database (DB), to be subsequently used for the metamodel training. Here, $\mathbf{x}$ denotes the values of the design variables associated with a candidate solution and $\mathbf{f}(\,\cdot\,)$ is the cost function; symbols in bold denote multivariate quantities or functions. Upon completion of the starting phase ($g > g_{start}$), any offspring $\mathbf{x} \in \mathcal{P}_{\lambda,g}$ is first evaluated using the metamodel(s) and $\tilde{\mathbf{f}}(\mathbf{x})$ becomes available. Based on $\tilde{\mathbf{f}}(\mathbf{x})$, a provisional (scalar) cost value is assigned to each offspring, namely $\tilde{\phi}(\mathbf{x}) = \tilde{\phi}(\tilde{\mathbf{f}}(\mathbf{x}), \{\tilde{\mathbf{f}}(\mathbf{z}) \mid \mathbf{z} \in \mathcal{P}_{\lambda,g} \setminus \{\mathbf{x}\}\})$.

In single-objective optimization, this cost assignment is trivial: $\tilde{\phi}(\mathbf{x}) = \tilde{f}(\mathbf{x})$. On the other hand, it is known that, in multi-objective optimization, the scalar cost assignment to a multivariate objective function is possible based on many different techniques. However, since $\tilde{\mathbf{f}}$ is just an approximation, it is desirable to avoid the inevitably large number of comparisons required by the commonly used cost assignment techniques, so a simple ranking in nondominated fronts is sufficient to compute $\tilde{\phi}$.

Based on the so-computed $\tilde{\phi}$ values, $\lambda_e < \lambda$ offspring are singled out in $\mathcal{P}_e$, as follows: $\mathcal{P}_e = \{\mathbf{x}_i, i = 1, 2, \ldots, \lambda_e : \tilde{\phi}(\mathbf{x}_i) < \tilde{\phi}(\mathbf{z}),\ \mathbf{z} \in \mathcal{P}_{\lambda,g} \setminus \mathcal{P}_e\}$. Then, the task of exact evaluations for the current generation starts for all $\mathbf{x} \in \mathcal{P}_e$; their exact objective function values $\mathbf{f}(\mathbf{x})$ are computed and stored in the DB.

Then, final cost values $\phi(\mathbf{x}) = \phi(\hat{\mathbf{f}}(\mathbf{x}), \{\hat{\mathbf{f}}(\mathbf{z}) \mid \mathbf{z} \in \mathcal{P}_g \setminus \{\mathbf{x}\}\})$ are assigned to any $\mathbf{x} \in \mathcal{P}_g$. Note

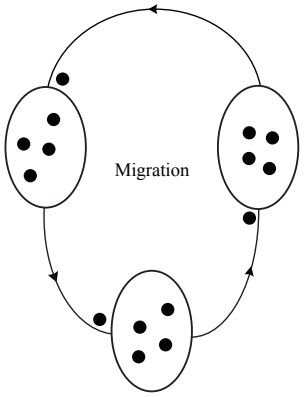

Figure 1: In the so-called *island model* of DEAs, the evolution of subpopulations (islands) is carried out simultaneously; they regularly exchange best individuals, according to various interconnection schemes (a simple variant is shown here).

that $\hat{\mathbf{f}}(\mathbf{x})$ stands for $\mathbf{f}(\mathbf{x})$, if $\mathbf{x}$ has been exactly evaluated, or $\tilde{\mathbf{f}}(\mathbf{x})$, otherwise. In multi-objecive problems, techniques such as NSGA [32], NSGA-II [9], SPEA [38], SPEA2 [37] undertake this task.

The elite selection operator E is then applied to $\mathcal{P}_e$ and updates the previous elite archive, $\mathcal{P}_{a,g+1} = \mathrm{E}(\mathcal{P}_e \cup \mathcal{P}_{\alpha,g})$. New parents are selected, $\mathcal{P}_{\mu,g+1} = \mathrm{S}(\mathcal{P}_{\mu,g} \cup \mathcal{P}_{\lambda,g})$, using the selection operator S and new offspring are created by means of the crossover C and mutation M operators, as follows: $\mathcal{P}_{\lambda,g+1} = \mathrm{M}(\mathrm{C}(\mathcal{P}_{\mu,g+1}, \mathcal{P}_{\alpha,g+1}))$. If the maximum number of exact evaluation has been reached, the algorithm terminates. Otherwise, $g \leftarrow g + 1$ and evolution goes on.

So far, no assumption is made about the incorporated metamodels. They can be of any kind and either *global* (periodically retrained) or *local* (built separately for each and every new individual, using local data) ones can be used. To our experience [15, 14] *local* metamodels provide better predictions and maximize the economy in CPU cost of EA-IPE.

2 Distributed Evolutionary Algorithms

A DEA is based on semi-isolated populations, evolving autonomously for a couple of generations and allows periodic migrations of individuals among them; this scheme is known as the *island model* and is illustrated in fig. 1. In the literature, it has been demonstrated that DEAs outperform single-population EAs [7, 35, 3, 1].

To optimize the DEA performance, a couple of parameters must be tuned. Among them are the number of individuals allowed to migrate (*migration rate*), the number of generations between two successive migrations (*migration frequency*), the selection scheme for the emigrants and for those to be replaced in the host island (*emigration and replacement policies*), the graph connecting islands, etc. [4, 8]. Usually, either the best individual(s) in a island and/or some randomly selected ones emigrate to another, where they replace either the worst and/or some randomly selected individuals [34, 2]. Different evolution policies (crossover type, mutation probability, elitism) can be employed in different islands to increase exploration and exploitation [33, 18]. These policies may dynamically vary by monitoring the performance of the islands [10]; additional measures can be taken to preserve diversity, like the spreading of infections to islands which tend to become too uniform [11, 21].

The island model is not the only possible one. An alternative basis for creating DEAs (the *cellular model*) is by restricting the selection of mates of each individual only amidst its neighbouring ones, according to a specified topology [16, 26, 12].

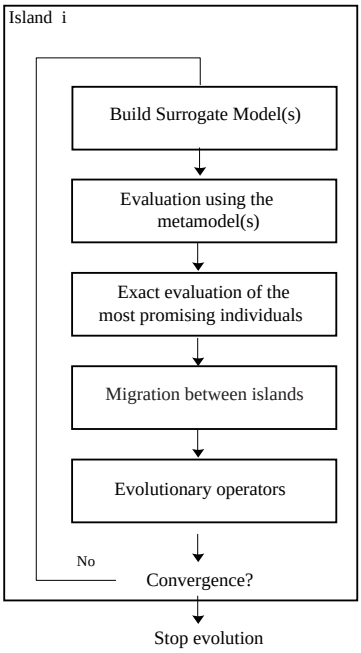

Figure 2: The inexact pre-evaluation (IPE) employed at the i-th island of a DEA. The evolution is identical to that of a single-population MAEA except for the extra migration step.

Incorporating metamodels for the IPE of offspring in a DEA may further increase its performance. IPE is more suitable for the island model of DEAs, since in the cellular model generations are loosely defined, because mating occurs locally and so it does not require the synchronisation of the population members. Henceforth, the term DEA will refer to the island model of fig. 1, where each island evolves autonomously and, so, IPE can readily be introduced. This is illustrated in fig. 2.

3 Hierarchical Evolutionary Algorithms

As described in a previous section, MAEAs are based on the utilization of low-cost evaluation tools (metamodels), which interpolate or approximate existing fitness or cost values. The term *metamodel* was used to denote generic prediction tools, which do not consider the "physics" of the problem or the accuracy of the evaluation tool used to collect the training data. Or, in other words, the same metamodel used in MAEAs can also be used for completely different applications.

The same concept can be generalized by establishing and using a *hierarchy* of evaluation tools, of different CPU cost and accuracy each. It is not necessary that there are only two evaluation levels as in the aforementioned MAEAs. For instance, one may replace the generic metamodel by a low-cost numerical solver to the physical problem under consideration. This gives rise to the so-called Hierarchical EAs (HEAs).

HEAs for aerodynamic shape optimization problems can be devised by considering the available CFD software for the numerical prediction of flowfields. It is known that the same flow problem can be analyzed through Navier-Stokes (N-S) solvers with different turbulence models (algebraic, one- or two-equation differential models, large-eddy simulations, etc.), different treatments of the turbulence equations at the wall (integration of the flow equations to the wall or through the wall function technique) and, of course, with different grids. Integral boundary layer

methods, coupled with an external flow solver and a viscous-inviscid flow interaction scheme (V-II), whenever applicable, offer an alternative fast way to model the flow problem. So, a typical HEA for aerodynamic shape optimization may consist of two levels, the lower one utilizing the fast V-II tool, and the upper one based on the N-S solver. The low-level EA runs concurrently with the high-level EA. It is not necessary, neither recommended, that both EAs use the same evolution parameters (parent and offspring population sizes, crossover and mutation parameters, parent selection scheme, etc.). The two levels communicate regularly by exchanging their elite individuals. Immigrants at any level should be evaluated with the level's exact evaluation tool, before being incorporated into the level's population. Numerical experiments have shown that a well tuned HEA with balanced exploration —at the lower level— and exploitation —at the higher one— clearly outperforms a conventional EA.

Though metamodels are considered to be "non-physical", low-level evaluation tools, additional gain in the optimization CPU cost can be achieved by letting metamodels screen the population members at each level, in accordance to the IPE technique. Note that it is not wise to mix cost values obtained from different evaluation models in a single DB, from which training data for the metamodel are drawn. Thus, a hierarchical MAEA should use as many DBs as the number of levels or the number of exact evaluation tools.

4 Hierarchical Distributed MAEAs

In the last section on the theory of low-cost EAs, recent progress made by hybridizing all the aforementioned CPU-saving techniques is presented. In fact, this section deals with *Hierarchical, Distributed* MAEAs (HDMAEAs), based on the inexact pre-evaluation of candidate solutions (hence the term HD(EA-IPE)).

First, let us make clear that the notion of *hierarchy* in an EA can be used in some other ways as well. For instance, a simple kind of hierarchy, based on different evolution policies, has been presented in [19], where some of the DEA islands (*low-level islands*) explore the design space, whereas other (*high-level islands*) exploit the promising solutions located by the former. The use of different models in different islands, with a communication scheme ensuring the propagation of promising solutions from islands using the low-accuracy models to the one employing the high-accuracy model, was first proposed in [25]. We further extended these ideas and proposed the HD(EA-IPE) algorithm, which makes use of different models of the underlying physical process wihout discarding the indisputable benefits from IPE [24].

Practically, a HEA can be readily extended to accommodate the *island model* described in a previous section. It suffices to assign multiple DEAs to levels with evaluation models of different precision and CPU cost. These are the *exact models* of each level, to distinguish them from the metamodels trained on their results, which are also used. These metamodels can be used to provide approximations to the level's exact objective function, according to what was previously referred to as the IPE phase, which is now separately applied over each island, as in fig. 2. The HD(EA-IPE) scheme is schematically presented in fig. 3. The IPE phase is more useful at the higher levels, where the cost of an exact evaluation is high. The use of metamodels asks for DBs storing the outcomes of previous evaluations. Thus each level has its own DB, to which the results of the evaluations with the level's exact model are stored. This is shared by all level's islands but remains inaccessible from the other levels (fig. 3).

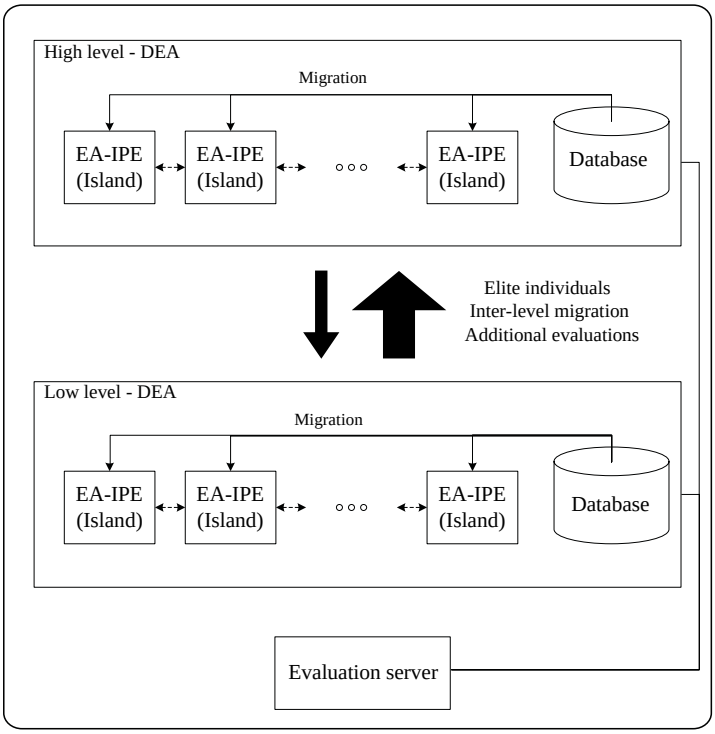

Figure 3: Implementation of the HD(EA-IPE). Each island evolves autonomously; synchronization occurs only prior to migrations. Requests for evaluations are posted to the evaluation server, who manages the computational resources, such as the available processing units. The results of evaluations are stored in the commonly accessible DB, from which training patterns are retrieved to construct or update the metamodel(s). Individuals exchanged between adjacent levels are evaluated with the importing level's exact evaluation model before incorporation to the level's islands.

An add-on feature of the HD(EA-IPE) algorithm compared to D(EA-IPE) is the inter-level communication. Imposing an hierarchy means that each level can communicate only with its adjacent ones, i.e. its lower and upper level. For each level to communicate, its islands should have accomplished a minimum number of generations. Once this criterion is satisfied for two adjacent levels, the exchange of elite individuals, gathered from all level's islands, takes place. When importing individuals from another level, these should first be evaluated with the level's exact model. In this way, the mixing of heterogeneous objective function values, which could mislead the selection process, is avoided. The level's immigrants, after having been re-evaluated, are distributed to the islands. Individuals coming from the lower level replace the worst members of the host islands, only if they outperform them. If they originate from the higher level, the replacement takes place without any further examination. Should a level consecutively fail to provide useful individuals to its higher one, its evolution as well as that of all its lower levels terminates, in order to save CPU cost. The inter-level communication, without technical details, is presented in fig. 4.

5 HDMAEA: Selected Examples

The gain in CPU time expected from a hierarchical EA can be demonstrated by examining the problem of minimizing an analytical function, viz. the Rastrigin function, defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 + 0.5\left(n - \sum_{i=1}^{n} \cos(2\pi x_i)\right), \qquad (1)$$
$$x_i \in [-5.12, 5.12],\ i = 1, 2, \ldots, n, \quad n = 30.$$

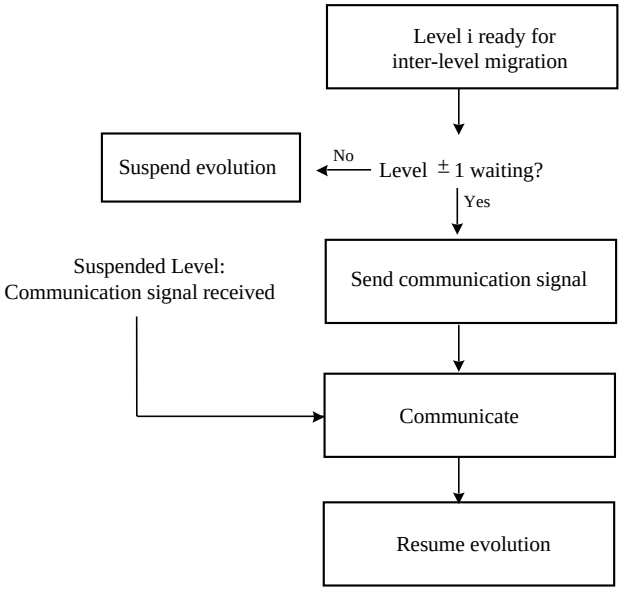

Figure 4: Communication between successive levels in a HDEA. Level i communicates only with its lower $i-1$ and upper $i+1$ one.

In this case, for each candidate solution $\mathbf{x}$, the exact evaluation tool returns the value of the above function. Since for a two-level HEAan approximate evaluation tool is also needed, this was defined by

$$f(\mathbf{x}) = \delta f + (1+\delta A)\sum_{i=1}^{n}(x_i+\delta c)^2 + (0.5+\delta B)\left(n - \sum_{i=1}^{n}\cos(2\pi x_i + \delta\phi)\right) \tag{2}$$

where

$$\delta f = 1.5n, \quad \delta A = -0.5, \quad \delta c = -0.7, \quad \delta B = 0.1, \quad \delta\phi = 1.5.$$

Both the exact and approximate models with univariate inputs are illustrated in fig. 5. The convergence plots of each one of the two levels are shown in fig. 6. A hypothetical CPU cost ratio of 30 : 1 between the exact (eq. 1) and the approximate (eq. 2) model is assumed, so as to "mimic" the CPU cost ratio of N-S and V-II solvers to be used in the next two examples. Based on this assumption, the cost of the hierarchical optimization is about five times less than the cost of a MAEA based on the exact evaluation model. The reduction in the total CPU

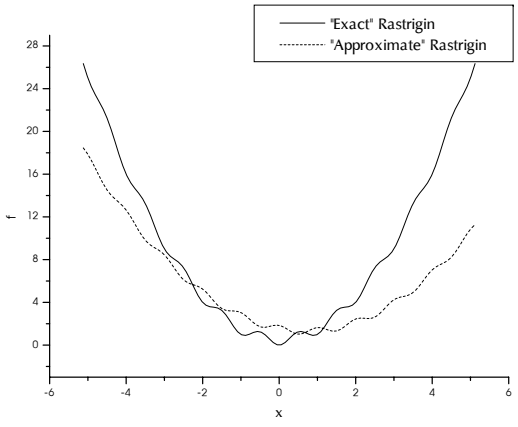

Figure 5: Comparison of the exact univariate Rastrigin function and its purposely built approximate model.

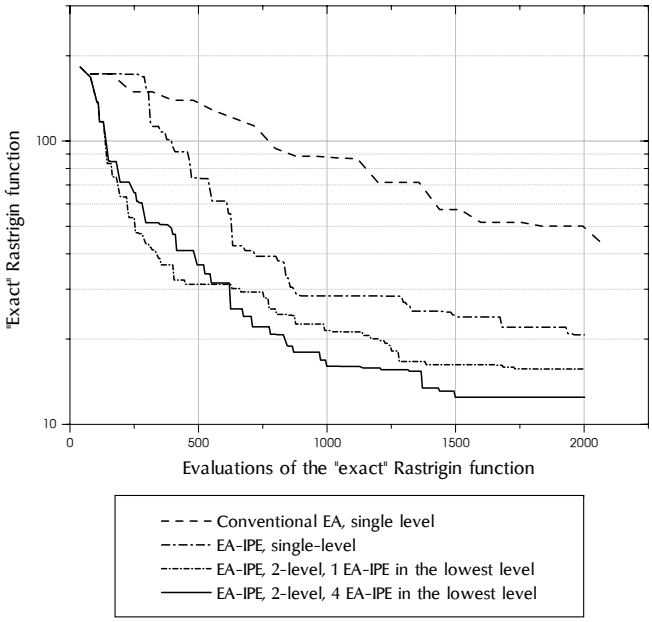

Figure 6: Convergence history for the Rastrigin problem. The third curve corresponds to HEA with IPE ate each level. The fourth curve corresponds to a HD(EA-IPE) where four islands were used at the lower level, only.

cost is even greater compared to the cost of a conventional EA. It should be stretched out that, the use of a distributed MAEA in the lower level accelerates the convergence of the hierarchical optimization algorithm further.

The second case is concerned with the design of an isolated airfoil with minimum drag, at $\mathrm{Re} = 6.5 \cdot 10^6$, $\mathrm{M}_\infty = 0.73$, $\alpha_\infty = 3.19°$. The airfoil drag should be lower than $c_d = 0.0128$, which is the drag of an existing airfoil (RAE 2822) at the same flow conditions; RAE 2822 is considered to be the reference shape and its known lift coefficient value $c_l = 0.764$ is enforced as equality constraint. The maximum airfoil thickness, which should exceed 11% of the chord length, is imposed as geometrical constraint.

The convergence history is plotted in fig. 7. A clear superiority of the HD(EA-IPE) scheme is apparent not only over the conventional EA but also over algorithms that use metamodels to reduce the calls to the N-S equation solver. Note that the HEA is based on a N-S (high level) and a V-II (low level) tool. The conventional EA required approximately 6162 min (CPU time on an Intel Pentium 4 processor) to reduce c_d by 22.5%. With HD(EA-IPE), the same reduction in c_d is achieved in 1542 min. From another point of view, drag is reduced by 34.2% compared to the reference shape, by allowing the HEA to run as long as the conventional EA (approximately 6162 min).

The last case aims at the design of a compressor cascade operating at $\mathrm{M}_1 = 0.618$, $\mathrm{Re} = 8.41 \cdot 10^5$, $\alpha_1 = 47.0°$. The objective is to minimize the mass-averaged total pressure losses at the exit plane, $\omega = (p_{01} - p_{02})/(p_{01} - p_1)$, while preserving the prescribed flow turning (27°). The losses of the reference cascade, computed by the N-S code, are $\omega = 0.0187$. The outlet flow angle is imposed as an aerodynamic constraint. The minimum maximum airfoil thickness is constrained to 10% of the chord.

The convergence history is plotted in fig. 8. In this case, the supremacy of the HD(EA-IPE) algorithm is even more pronounced, since there is a better agreement between the N-S solver and V-II method results. Here, the computational cost of evaluating a candidate solution with the V-II method is approximately 1/80 of the N-S equation solving cost.

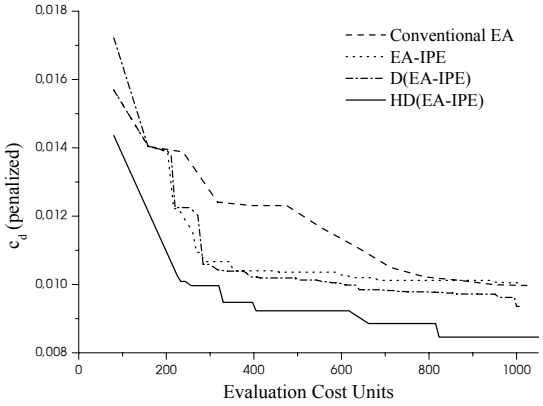

Figure 7: Drag minimization of an isolated airfoil: Comparison of convergence histories of EA, DEA and HDEA, all of them assisted by the Inexact Pre-Evaluation phase (IPE), The convergence of a conventional EA is also shown.

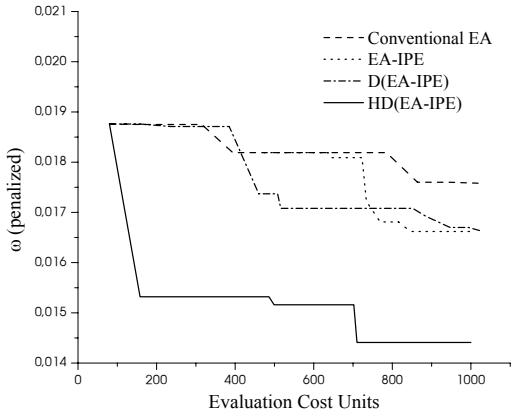

Figure 8: Losses minimization in a compressor cascade: Comparison of convergence histories of EA, DEA and HDEA, all of them assisted by the Inexact Pre-Evaluation phase (IPE), The convergence of a conventional EA is also shown.

Acknowledgement

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 the Program PYTHAGORAS II.

REFERENCES

[1] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462, October 2002.

[2] E. Alba and J. M. Troya. Influence of the migration policy in parallel distributed GAs with structured and panmictic populations. *Applied Intelligence*, 12(3):163–181, May 2000.

[3] E. Alba and J. M. Troya. Improving flexibility and efficiency by adding parallelism to genetic algorithms. *Statistics and Computing*, 12(2):91–114, April 2002.

[4] T. C. Belding. The distributed genetic algorithm revisited. In L. J. Eshelman, editor, *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 114–121, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers.

[5] 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*, 35(2):183–194, May 2005.

[6] L. Bull. On model-based evolutionary computation. *Soft Computing — A Fusion of Foundations, Methodologies and Applications*, 3(2):76–82, 1999.

[7] E. Cantú-Paz. A survey of parallel genetic algorithms. *Calculateurs Parallèlles, Résaux et Systèmes Répartis*, 10(2):141–171, 1998.

[8] E. Cantú-Paz and D. E. Goldberg. On the scalability of parallel genetic algorithms. *Evolutionary Computation*, 7(4):429–449, 1999.

[9] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo 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 849–858, Paris, France, September 2000. Springer-Verlag.

[10] D. J. Doorly and J. Peiró. Supervised parallel genetic algorithms in aerodynamic optimisation. In *13th AIAA Computational Fluid Dynamics Conference*, Snowmass Village, CO, USA, July 1997. AIAA-1997-1852.

[11] D. J. Doorly, J. Peiró, and S. Spooner. Design optimisation using distributed evolutionary methods. In *37th Aerospace Sciences Meeting and Exhibit*, Reno, NV, USA, January 1999. AIAA-1999-111.

[12] S. E. Eklund. A massively parallel architecture for distributed genetic algorithms. *Parallel Computing*, 30(5–6):647–676, May–June 2004.

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

[14] 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.

[15] 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.

[16] M. Gorges-Schleuter. ASPARAGOS an asynchronous parallel genetic optimization strategy. In J. D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 422–427, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers.

[17] R. M. Greenman and K. R. Roth. Minimizing computational data requirements for multi-element airfoils using neural networks. In *37th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, USA, January 1999. AIAA-1999-0258.

[18] F. Herrera and M. Lozano. Gradual distributed real-coded genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(1):43–63, April 2000.

[19] F. Herrera, M. Lozano, and C. Moraga. Hybrid distributed real-coded genetic algorithms. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature — PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 603–612, Amsterdam, The Netherlands, 1998. Springer-Verlag.

[20] Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5):481–494, October 2002.

[21] 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*, 43(10–11):1149–1166, 2003.

[22] 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.

[23] M. K. Karakasis and K. C. Giannakoglou. On the use of metamodel-assisted, multi-objective evolutionary algorithms. *Engineering Optimization*, 2005. To appear.

[24] M. K. Karakasis, D. G. Koubogiannis, and K. C. Giannakoglou. Hierarchical distributed evolutionary algorithms in shape optimization. *International Journal for Numerical Methods in Fluids*, 2005. Submitted for publication.

[25] S.-C. Lin, W. F. Punch, and E. D. Goodman. Coarse-grain parallel genetic algorithms: categorization and new approach. In *Proceedings of the 6th IEEE Symposium on Parallel and Distributed Processing*, pages 28–37, October 1994.

[26] B. Manderick and P. Spiessens. Fine-grained parallel genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 428–433, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers.

[27] H. Nakayama, K. Inoue, and Y. Yoshimori. Approximate optimization using computational intelligence and its application to reinforcement of cable-stayed bridges. 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.

[28] 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)*, volume 3, pages 1856–1863, Canberra, Australia, December 2003.

[29] M. Papadrakakis, N. D. Lagaros, and Y. Tsompanakis. Structural optimization using evolution strategies and neural networks. *Computer Methods in Applied Mechanics and Engineering*, 156(1–4):309–333, April 1998.

[30] 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.

[31] 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.

[32] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1995.

[33] R. Tanese. Parallel genetic algorithms for a hypercube. In J. J. Grefenstette, editor, *Proceedings of the 2nd International Conference on Genetic Algorithms*, pages 177–183, Cambridge, MA, USA, July 1987. Lawrence Erlbaum Associates.

[34] 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.

[35] M. Tomassini. Parallel and distributed evolutionary algorithms: A review. In K. Miettinen, M. M. Mäkelä, P. Neittaanmäki, and J. Périaux, editors, *Evolutionary Algorithms in Engineering and Computer Science — Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications*, pages 113–133. John Wiley & Sons, Chichester, UK, 1999.

[36] 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)*, volume 1, pages 692–699, Canberra, Australia, 2003.

[37] E. Zitzler, M. Laumans, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for multiobjective optimization. In K. Giannakoglou, D. Tsahalis, J. Périaux, K. Papailiou, and T. Fogarty, editors, *Evolutionary Methods for Design, Optimisation and Control*, pages 19–26, Barcelona, Spain, 2002. CIMNE.

[38] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, November 1999.