An Asynchronous Metamodel-Assisted Memetic Algorithm for CFD-based shape optimization

Evgenia A. Kontoleontos, Varvara G. Asouti, Kyriakos C. Giannakoglou

An Asynchronous Metamodel-Assisted Memetic Algorithm for CFD-based shape optimization

Journal:	*Engineering Optimization*
Manuscript ID:	GENO-2010-0217.R2
Manuscript Type:	Original Article
Date Submitted by the Author:	11-Feb-2011
Complete List of Authors:	Kontoleontos, Evgenia; National Technical University of Athens Asouti, Varvara; National Technical University of Athens Giannakoglou, Kyriakos; National technical University of Athens
Keywords:	Shape Optimization, Computational Fluid Dynamics, Memetic Algorithm, Adjoint Method, Asynchronous Metamodel-Assisted Evolutionary Algorithm

Note: The following files were submitted by the author for peer review, but cannot be converted to PDF. You must view these files (e.g. movies) online.

amama.tex
cfd.bib
eas.bib
gENO2e.cls
gENO.bst
ltt.bib

Engineering Optimization
Vol. 00, No. 00, September 2010, 1–21

An Asynchronous Metamodel–Assisted Memetic Algorithm for CFD–based shape optimization

Evgenia A. Kontoleontos, Varvara G. Asouti and Kyriakos C. Giannakoglou*

National Technical University of Athens, School of Mechanical Engineering, Parallel CFD & Optimization Unit, P.O. Box 64069, Athens 157 10, Greece

(*September 2010*)

This paper presents an asynchronous metamodel–assisted memetic algorithm for the solution of CFD–based optimization problems. This algorithm is appropriate for use on multiprocessor platforms and may solve computationally expensive optimization problems in reduced wall–clock time, compared to conventional evolutionary or memetic algorithms. It is, in fact, a hybridization of non–generation–based (asynchronous) evolutionary algorithms, assisted by surrogate evaluation models, a local search method and the Lamarckian learning process. For the objective functions gradient computation, in CFD applications, the adjoint method is used. Issues concerning the "smart" implementation of local search in multi–objective problems are discussed. In this respect, an algorithmic scheme for reducing the number of calls to the adjoint equations to just one, irrespective of the number of objectives, is proposed. The algorithm is applied to CFD–based shape optimization of the tubes of a heat exchanger and of a turbomachinery cascade.

Keywords: Shape Optimization; Computational Fluid Dynamics; Memetic Algorithm; Adjoint Method; Asynchronous Metamodel–Assisted Evolutionary Algorithm.

Nomenclature

AEA	Asynchronous EA
AMA	Asynchronous MA
$AMAEA$	Asynchronous MAEA
$AMAMA$	Asynchronous MAMA
DB	Database (archive) of previously evaluated individuals

*Corresponding author. E-mail: kgianna@central.ntua.gr

ISSN: 0305-215X print/ISSN 1029-0273 online

EA	Evolutionary Algorithm
IPE	Inexact Pre–Evaluation
LS	Local Search
$MAEA$	Metamodel–Assisted EA
$MAMA$	Metamodel–Assisted MA
MOO	Multi–Objective Optimization
RBF	Radial Basis Function (network)
SOO	Single–Objective Optimization

1. Introduction

Engineering optimization problems can be solved using either stochastic or gradient–based optimization methods. Evolutionary algorithms (EAs) are, by far, the most frequently used global search methods. They may accommodate any analysis software as a black–box tool and reach the optimal solution without being trapped into local optima. Unfortunately, solving optimization problems associated with a computationally demanding evaluation software, such as CFD codes, becomes expensive. To reach the optimal solution(s), EAs may require a great number of objective function evaluations, increasing thus the CPU cost. On the other hand, gradient–based methods are appropriate for local search but can easily be trapped to local optima. They should be supported by tools computing or approximating the gradient of the objective function.

One may overcome the weaknesses of stochastic and gradient–based methods through their hybridization. In hybrid schemes, EAs are frequently used to explore the design space whereas gradient–based methods undertake the refinement of promising individuals. Hybrid methods can be devised in several ways, Poloni *et al.* (2000), Sefrioui and Périaux (2000), Désidéri and Janka (2003), Duvigneau *et al.* (2006), Karakasis *et al.* (2007), Kampolis and Giannakoglou (2009, 2011).

By definition, memetic algorithms (MAs), are hybrid optimization methods since they combine global and local search (LS), Dawkin (1976), Hart (1994), Knowles and Corne (2000), Ong and Keane (2004), Krasnogor and Smith (2005), Ong *et al.* (2006). In MAs, two basic learning mechanisms, namely the Lamarckian and the Baldwinian ones, are employed. In the former, any individual refined during the LS replaces both the genotype and phenotype of the starting one in the population whereas, in the latter, only the objective vector is allowed to be updated.

Unfortunately, even with hybrid methods, a computationally demanding evaluation software makes the optimization task very expensive. In order to reduce its wall–clock time, surrogate evaluation models (also known as metamodels) can be used. Metamodel–Assisted EAs (MAEAs), in which the metamodels are trained separately from the evolution which is exclusively based on them, can be found in Bull (1999), Pierret and Van den Braembussche (1999), but are beyond the scope of this paper. This paper is concerned with EAs (MAs, in fact) assisted by on–line trained metamodels, in conformity with the method presented in Karakasis and Giannakoglou (2006), Giannakoglou *et al.* (2001). In each generation, the metamodels undertake the so–called inexact pre–evaluation (IPE) of candidate solutions and pinpoint the most promising among them to undergo CFD–based evaluation. MAs supported by the IPE technique, i.e. the metamodel–assisted MAs (MAMAs), have been presented in the past by the same group, Georgopoulou and Giannakoglou (2009); there, the metamodels were also differentiated to approximate the gradient. Relevant works on MAMAs can be found in Zhou *et al.* (2007a,b).

To reduce the ellapsed time of an optimization problem, population members within each generation can be concurrently evaluated on different CPUs. This is the simplest way to exploit parallelization in EAs. In the literature, the term "parallel EAs" (PEAs) denotes much more than this, see Cantú-Paz (1998), Nowostawski and Poli (1999), Alba and Tomassini (2002). PEAs are suited for either cluster or grid computing, Lim *et al.* (2007), Melab *et al.* (2006), Liakopoulos *et al.* (2008), Luna *et al.* (2006). Note that the evolution on a generation–by–generation basis limits the parallel efficiency of a PEA due to the synchronization barrier at the end of each gereration. Thus, asynchronous EAs (AEAs), Alba and Troya (2001), Asouti and Giannakoglou (2009), which maximize the exploitation of the available computational resources, have been proposed instead.

In this paper, the combined use of the AEA introduced in Asouti and Giannakoglou (2009) and, then, enhanced also by metamodels (AMAEA), Asouti *et al.* (2009), along with a gradient–based method gives rise to a new asynchronous metamodel–assisted memetic algorithm (AMAMA). In the proposed AMAMA (as in the existing AEA), the population is arranged on a 2D structured mesh and divided into overlapping demes. The selection of the new individual to undertake evaluation on an instantaneously idle processor results from inter– and intra–deme processes. The use of metamodels is based on the IPE technique, revisited to efficiently cooperate with the AEA. Over and above, the AMAMA regularly performs LS, using gradient–based methods. LS includes the computation of the gradient of the objective function with respect to the design variables, the refinement of the individual using the steepest descent method and the re–evaluation of the refined individual. To the authors knowledge, an asynchronous memetic algorithm assisted by metamodels is presented for the first time in the relevant literature.

In CFD–based optimization problems, the gradient of the objective function can be computed using the adjoint method. In general, the adjoint equations in discrete form can be derived through either the continuous, Pironneau (1974), Jameson (1988), Anderson and Venkatakrishnan (1997), Papadimitriou and Giannakoglou (2007, 2008, 2009), or the discrete adjoint approach, Elliot and Peraire (1996), Giles and Pierce (1997), Duta *et al.* (2002). In the former, the adjoint equations are derived as p.d.e.'s (similar to the state equations governing the flow problem) and, then, discretized. In the latter, the discrete adjoint equations result directly from the discretized state equations. In this paper, the continuous adjoint formulation for incompressible flows with heat transfer is employed. In order to further reduce the CPU cost in multi–objective optimization (MOO) problems, a scheme according to which the adjoint equations are solved only once, instead of as many times as the objectives, is proposed. Though, in the present paper, the derivatives of the approximated SPEA2 utility function are computed as in Kampolis and Giannakoglou (2008), an important novelty is that these are used as "frozen" weighting factors in the aggregated objective function handled by the adjoint method and, as a consequence, a single run of the adjoint method is required. Regarding CPU cost, this is an important advantage.

The proposed method is demonstrated on single– and multi–objective CFD–based, engineering problems. These include the two–objective shape optimization of the tubes of a tube bank heat exchanger and the single–objective optimization (SOO) problem of a turbomachinery cascade. For the turbulence flow case, closure is effected by the Spalart–Allmaras turbulence model and the adjoint to both the mean–flow and turbulence equations is computed, as in Zymaris *et al.* (2009). So, there is no need to make the assumption that the variation in turbulence viscosity is neglected, which is a source of inaccuracies. This is presented for the first time for incompressible flows with heat transfer and constitutes the third originality of this paper. Statistics on the solution to

function minimization are shown in the appendix for the sake of completness.

2. Flow Model–Objective Function

The CFD model used for the aero–thermodynamic evaluation of candidate solutions is an in–house Navier–Stokes flow solver for incompressible flows based on the artificial compressibility technique, Anderson *et al.* (1995), and a vertex–centered–finite volume scheme. The Navier–Stokes equations for the 2D steady flow of an incompressible fluid are symbolically written as

$$R_{\vec{U}} = 0 \tag{1}$$

where $\vec{U}=[p, v_i]^T$ is the vector of the mean flow state variables, with p the static pressure and $v_i, i=1,2$ the velocity components. During numerical solution, a pseudo–time derivative of $\vec{U}$ is added to the steady state residuals, given by

$$\begin{aligned} R_p &= \beta^2 \frac{\partial v_j}{\partial x_j} \\ R_{v_i} &= v_j \frac{\partial v_i}{\partial x_j} + \frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[(\nu+\nu_t)\left(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}\right)\right] \end{aligned} \tag{2}$$

where β is the artificial compressibility coefficient, ρ is the constant density and $x_i, i=1,2$ the Cartesian coordinates. ν and ν_t are the bulk and turbulent viscosity, respectively. Based on the Spalart and Allmaras (1994) turbulence model, the viscosity coefficient is given by $\nu_t = \tilde{\nu} f_{v_1}$, where $\tilde{\nu}$ is the solution variable in the corresponding state equation, $R_{\tilde{\nu}}=0$, where

$$R_{\tilde{\nu}} = \frac{\partial(v_i\tilde{\nu})}{\partial x_i} - \frac{\partial}{\partial x_i}\left[\left(\nu+\frac{\tilde{\nu}}{\sigma}\right)\frac{\partial\tilde{\nu}}{\partial x_i}\right] - \frac{c_{b_2}}{\sigma}\left(\frac{\partial\tilde{\nu}}{\partial x_i}\right)^2 - \tilde{\nu}P(\tilde{\nu}) + \tilde{\nu}D(\tilde{\nu}) \tag{3}$$

The production $P(\tilde{\nu})$ and destruction $D(\tilde{\nu})$ terms are given by

$$P(\tilde{\nu}) = c_{b1}\tilde{S}, \quad D(\tilde{\nu}) = c_{w1} f_w(\tilde{S})\frac{\tilde{\nu}}{d^2} \tag{4}$$

Terms f_{v_1}, f_w, $\tilde{S}$, and constants c_{b_1}, c_{b_2}, c_{w_1} and σ are all defined in Spalart and Allmaras (1994). d is the distance of each grid node from the wall. Depending on the application, the energy equation,

$$R_T = \frac{\partial(v_i T)}{\partial x_i} - \alpha\frac{\partial^2 T}{\partial x_i^2} = 0 \tag{5}$$

where T is the temperature and $\alpha = \frac{k}{\rho c_p}$ the thermal diffusivity, must also be satisfied. This is solved in a segregated manner after iteratively solving the other state equations. c_p and k stand for the specific heat capacity and thermal conductivity, respectively.

Depending on the application, the objective functions f_i to be minimized are

$$f_1 = -\int_{S_{I,O}} \frac{1}{\rho}\left(p + \frac{1}{2}\rho v^2\right) v_i n_i dS$$
$$f_2 = -\int_{S_{I,O}} T dS \tag{6}$$

and correspond to the volume–averaged total pressure losses and the temperature difference between the inlet (S_I) to and the outlet (S_O) from the flow domain, respectively. n_i correspond to the components of the normal to the boundary vector and v is the norm of the velocity vector.

In the M objective problem, during the LS action, the objective function F is defined by concatenating the M objectives f_i into a single scalar function

$$F = \sum_{i=1}^{M} \omega_i f_i \tag{7}$$

where ω_1, ω_2 are weighting factors as it will become clear in section 5. Otherwise, F may stand either for f_1 or f_2.

3. Gradient Computation–The Continuous Adjoint Method

In the continuous adjoint method, the augmented objective function F_{aug} is defined as the sum of the objective function F and the field (Ω) integral of the residual of the state equations ($R_{\vec{U},\tilde{\nu},T}=0$) multiplied by the adjoint variables ($\vec{V}=(q,\, u_i,\, \tilde{\nu}_a,\, T_a)$), $F_{aug} = F + \int_\Omega \vec{V} R_{\vec{U},\tilde{\nu},T} d\Omega$. Its variation with respect to the design variable array, $\vec{b} \in R^N$, is expressed as follows, as in Papadimitriou and Giannakoglou (2007),

$$\frac{\delta F_{aug}}{\delta \vec{b}} = \frac{\delta F}{\delta \vec{b}} + \int_\Omega q \frac{\delta R_p}{\delta \vec{b}} d\Omega + \int_\Omega u_i \frac{\delta R_{v_i}}{\delta \vec{b}} d\Omega + \int_\Omega \tilde{\nu}_a \frac{\delta R_{\tilde{\nu}}}{\delta \vec{b}} d\Omega + \int_\Omega T_a \frac{\delta R_T}{\delta \vec{b}} d\Omega$$
$$+ \int_S (q R_p + u_i R_{v_i} + \tilde{\nu}_a R_{\tilde{\nu}} + T_a R_T) \frac{\delta x_k}{\delta \vec{b}} n_k dS \tag{8}$$

The development of the field integrals of Equation (8), based on the Gauss divergence theorem, gives rise to

$$\begin{aligned}\frac{\delta F_{aug}}{\delta \vec{b}} = &\frac{\delta F}{\delta \vec{b}} + \int_\Omega \frac{\delta p}{\delta \vec{b}} R_q d\Omega + \int_\Omega \frac{\delta v_i}{\delta \vec{b}} R_{u_i} d\Omega + \int_\Omega \frac{\delta \tilde{\nu}}{\delta \vec{b}} R_{\tilde{\nu}_a} d\Omega + \int_\Omega \frac{\delta T}{\delta \vec{b}} R_{T_a} d\Omega \\ &+ \int_S (u_j n_j + \frac{\partial F}{\partial p}) \frac{\partial p}{\partial \vec{b}} dS - \int_S \nu \frac{\partial u_i}{\partial x_j} n_j \frac{\partial v_i}{\partial x_k} \frac{\delta x_k}{\delta \vec{b}} dS + \int_S \mathcal{BC}_{1,i} \frac{\delta v_i}{\delta \vec{b}} dS \\ &+ \int_\Omega \tilde{\nu}_a \tilde{\nu} \mathcal{C}_d(\tilde{\nu}, \vec{v}) \frac{\partial d}{\partial \vec{b}} d\Omega - \int_S \tilde{\nu}_a \frac{1}{\sigma} (\nu + \tilde{\nu}) \frac{\partial}{\partial x_j} \left(\frac{\partial \tilde{\nu}}{\partial \vec{b}} \right) n_j dS \\ &- \int_S \tilde{\nu}_a \frac{1}{\sigma} \frac{\partial \tilde{\nu}}{\partial x_j} \frac{\partial \tilde{\nu}}{\partial \vec{b}} n_j dS - \int_S 2 \frac{c_{b2}}{\sigma} \tilde{\nu}_a \frac{\partial \tilde{\nu}}{\partial x_j} \frac{\partial \tilde{\nu}}{\partial \vec{b}} n_j dS + \int_S \mathcal{BC}_2 \frac{\delta \tilde{\nu}}{\delta \vec{b}} dS \\ &+ \int_S \tilde{\nu}_a \tilde{\nu} e_{jli} e_{jmq} \frac{\mathcal{C}_S}{S} \frac{\partial v_q}{\partial x_j} n_l \frac{\partial v_k}{\delta \vec{b}} dS + \int_S (\tilde{\nu}_a R_{\tilde{\nu}} + T_a R_T) \frac{\delta x_k}{\delta \vec{b}} n_k dS \\ &- \int_S \alpha T_a \frac{\delta}{\partial \vec{b}} \left(\frac{\partial T}{\partial x_i} n_i \right) dS + \int_S \alpha T_a \frac{\partial T}{\partial x_i} \frac{\delta n_i}{\partial \vec{b}} dS - \int_S \alpha \frac{\partial T_a}{\partial x_j} n_j \frac{\partial T}{\partial x_k} \frac{\delta x_k}{\delta \vec{b}} dS \\ &+ \int_S \mathcal{BC}_3 \frac{\delta T}{\delta \vec{b}} dS - \int_S \nu \frac{\partial \tilde{\nu}_a}{\partial x_j} n_j \frac{\partial \tilde{\nu}}{\partial x_k} \frac{\delta x_k}{\delta \vec{b}} dS \end{aligned} \tag{9}$$

where

$$R_q = \frac{\partial u_j}{\partial x_j} \tag{10a}$$

$$\begin{aligned} R_{u_i} = &v_j \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + \frac{\partial}{\partial x_j} \left[(\nu + \nu_t) \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right] + \beta^2 \frac{\partial q}{\partial x_i} + \tilde{\nu} \frac{\partial \tilde{\nu}_a}{\partial x_i} \\ &+ \frac{\partial}{\partial x_l} \left(e_{jli} e_{jmq} \frac{\mathcal{C}_S}{S} \frac{\partial v_q}{\partial x_m} \tilde{\nu} \tilde{\nu}_a \right) + T \frac{\partial T_a}{\partial x_i} \end{aligned} \tag{10b}$$

$$\begin{aligned} R_{\tilde{\nu}_a} = &v_j \frac{\partial \tilde{\nu}_a}{\partial x_j} + \frac{\partial}{\partial x_j} \left[\left(\nu + \frac{\tilde{\nu}}{\sigma} \right) \frac{\partial \tilde{\nu}_a}{\partial x_j} \right] - \frac{1}{\sigma} \frac{\partial \tilde{\nu}_a}{\partial x_j} \frac{\partial \tilde{\nu}}{\partial x_j} - 2 \frac{c_{b2}}{\sigma} \frac{\partial}{\partial x_j} \left(\tilde{\nu}_a \frac{\partial \tilde{\nu}}{\partial x_j} \right) \\ &- \tilde{\nu}_a \tilde{\nu} \, \mathcal{C}_{\tilde{\nu}}(\tilde{\nu}, \vec{v}) - \frac{\delta \nu_t}{\delta \tilde{\nu}} \frac{\partial u_i}{\partial x_j} \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) - (-P + D) \, \tilde{\nu}_a \end{aligned} \tag{10c}$$

$$R_{T_a} = v_i \frac{\partial T_a}{\partial x_i} + \alpha \frac{\partial^2 T_a}{\partial x_i^2} \tag{10d}$$

$$\begin{aligned} \mathcal{BC}_{1,i} = &u_i v_j n_j + (\nu + \nu_t) \frac{\partial u_i}{\partial x_j} n_j + (u_j v_j + \beta^2 q + \tilde{\nu}_a \tilde{\nu} + T T_a) n_i + \\ &\tilde{\nu}_a \tilde{\nu} \mathcal{C}_S(\tilde{\nu}) \frac{1}{S} e_{jli} e_{jmq} \frac{\partial v_q}{\partial \vec{b}} n_l + \frac{\partial F}{\partial v_i} \end{aligned} \tag{11a}$$

$$\mathcal{BC}_2 = \frac{\delta \nu_t}{\delta \tilde{\nu}} u_i \left(\frac{\partial v_i}{\partial x_j} - \frac{\partial v_j}{\partial x_i} \right) n_j - \tilde{\nu}_a v_j n_j + \left(\nu + \frac{\tilde{\nu}}{\sigma} \right) \frac{\partial \tilde{\nu}_a}{\partial x_j} n_j + \frac{\partial F}{\partial \tilde{\nu}} \tag{11b}$$

$$\mathcal{BC}_3 = T_a v_i n_i + \alpha \frac{\partial T_a}{\partial x_i} n_i + \frac{\partial F}{\partial T} \tag{11c}$$

and, based on Equation (7), for constant ω_1 and ω_2,

$$\frac{\partial F}{\partial p} = -\omega_1 v_i n_i, \quad \frac{\partial F}{\partial v_i} = -\omega_1 (\frac{1}{2} v^2 n_i + v_i v_\lambda n_\lambda + p n_i) \tag{12}$$

$$\frac{\partial F}{\partial \tilde{\nu}} = 0, \quad \frac{\partial F}{\partial T} = -\omega_2 \tag{13}$$

Terms $\mathcal{C}_S$, $\mathcal{C}_{\tilde{\nu}}$, $\mathcal{C}_d$, S are all defined in Zymaris *et al.* (2009) and e_{jli} stands for the permutation symbol.

The adjoint field equations and their boundary conditions are derived by eliminating field integrals depending on $\frac{\delta p}{\delta \vec{b}}$, $\frac{\delta v_i}{\delta \vec{b}}$, $\frac{\delta \tilde{\nu}}{\delta \vec{b}}$, $\frac{\delta T}{\delta \vec{b}}$ from Equation (9). The field adjoint to the mean-flow, turbulence and energy equations are given by

$$R_q = 0, \quad R_{u_i} = 0, \quad R_{\tilde{\nu}_a} = 0, \quad R_{T_a} = 0 \tag{14}$$

The adjoint boundary conditions are defined in a similar way. For instance, at the inlet,

$$u_i n_i = \omega_1 v_i n_i, \quad u_i t_i = 0 \tag{15}$$

(where t_i are components of the unit, tangent to the boundary vector) for the normal and tangential velocities and zero Dirichlet conditions for $\tilde{\nu}_a$ and T_a. Along the solid walls, zero Dirichlet conditions are imposed to u_i, $\tilde{\nu}_a$ and T_a and zero Neumann to q. The outlet conditions for q and u_i are coupled based on the system of two equations $\mathcal{BC}_{1,i} = 0$ (for $i = 1$ and 2), after arbitrarily zeroing one of these variables, Zymaris *et al.* (2009). The $\tilde{\nu}_a$ and T_a outlet conditions result from $\mathcal{BC}_2 = 0$ and $\mathcal{BC}_3 = 0$, respectively.

After having computed the adjoint fields, by numerically satisfying the adjoint equations and their boundary conditions, the variation of the F_{aug} becomes independent of variations in the state variables, leading to the expressions of the sensitivity derivatives in terms of $\vec{V}$. Based on Equation (8) to (15), the sensitivity derivatives of F_{aug}, are given by

$$\begin{aligned}\frac{\delta F_{aug}}{\delta \vec{b}} = \frac{\delta F}{\delta \vec{b}} &- \int_{S_W} \nu \frac{\partial u_i}{\partial x_j} n_j \frac{\partial v_i}{\partial x_k} \frac{\delta x_k}{\delta \vec{b}} dS - \int_{S_W} \alpha \frac{\partial T_a}{\partial x_j} n_j \frac{\partial T}{\partial x_k} \frac{\delta x_k}{\delta \vec{b}} dS \\ &- \int_{S_W} \nu \frac{\partial \tilde{\nu}_a}{\partial x_j} n_j \frac{\partial \tilde{\nu}}{\partial x_k} \frac{\delta x_k}{\delta \vec{b}} dS + \int_\Omega \tilde{\nu}_a \tilde{\nu} \, \mathcal{C}_d(\tilde{\nu}, \vec{v}) \frac{\partial d}{\partial \vec{b}} d\Omega \end{aligned} \tag{16}$$

The wall boundary (S_W) and field integrals in Equation (16) depend on the state and adjoint variables.

4. Global Search Method : The Asynchronous MAEA (AMAEA)

As global search method, the asynchronous EA presented in Asouti and Giannakoglou (2009) and, later, enhanced by metamodels (AMAEA, Asouti *et al.* (2009)) is used. In this section, its basic features for the solution of MOO problems, with M functions to be minimized, namely

$$\min \vec{f}(\vec{b}) = \min\{f_1(\vec{b}), ..., f_M(\vec{b})\} \tag{17}$$

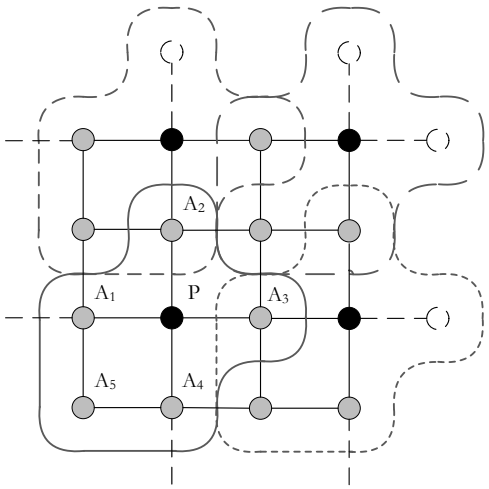


Figure 1.: Asynchronous EA: close–up view of part of the supporting mesh. The pole P, along with it's five evaluation agents A_1 to A_5 for the deme marked with the continuous line, are shown.

are presented. The comparison between two candidate solutions $\vec{b}_1$, $\vec{b}_2$ is based on dominance criteria and the approximation of the Pareto front (for ranking individuals and maintaining diversity) on the SPEA2 technique, Zitzler *et al.* (2001). Its SOO counterpart can be extracted in a straightforward manner.

The basic features of the AEA are the topological structure of the population, its division into demes and the specific way demes overlap and share individuals. Candidate solutions to the problem are associated with nodes of a $n_1 \times n_2$ structured supporting mesh which is periodic along its opposite sides. The mesh is subdivided into demes $\mathcal{D}_p$ of six nodes each: a pole P, which acts as the front–end of each deme where the best individual of the deme is stored, and five evaluation agents A_1 to A_5, Figure 1. Thus, on a $n_1 \times n_2$ mesh (both n_1 and n_2 must be even), with a total number of $N_{mesh} = n_1 n_2$ nodes, the number of poles equals to $N_{poles} = N_{mesh}/4$ and the number of evaluation agents equals to $N_{agents} = 3N_{mesh}/4$. The application of the evolution operators is restricted within each deme. The demes interact through shared nodes. According to Figure 1, each deme shares four of its five agents (all but A_5) with its four neighbouring demes.

The optimization starts by randomly generating N_{CPU} individuals at N_{CPU} randomly selected evaluation agents and assigning their evaluation to N_{CPU} available processors. Upon completion of the evaluation of any individual $\vec{b}_a$, the corresponding CPU ($CPU_{\vec{b}_a}$) becomes idle. Instantaneously, a new individual (new $\vec{b}_a$) to undergo evaluation is generated, through intra– and inter–deme operations. An intra–deme operation, based on dominance criteria, decides whether the just evaluated individual must displace or not the corresponding pole(s) ($\vec{b}_p$).

Then, the next agent to undergo evaluation is selected from the deme with the maximum priority through an inter–deme operation. The priority metric is defined as the product of age– and cost–based priorities, i.e. $Pr_p = Pr_p^{age} Pr_p^{cost}$, see Asouti and Giannakoglou (2009) for more details. The age A_k of an agent is the difference between the serial number of the last evaluation carried out for this agent and the serial number of the current evaluation. The age of any pole is the average age of its agents. The age–based priority is set equal to the pole's age divided by the maximum age of all poles. In MOO problems, the cost–based priority Pr_p^{cost} is defined using strength– and density–based criteria (SPEA2) and is non–dimensionalized by the difference between its maximum

and minimum values. The agent with the maximum age A_k, within the deme ($\mathcal{D}_p^n$) with the maximum priority, is the one selected to generate the new $\vec{b}_a$. All design variables are real coded and the new $\vec{b}_a$ is formed by superimposing the weighted difference between two agents of $\mathcal{D}_p^n$ to the individual currently associated with the pole ($\vec{b}_p$), as

$$\vec{b}_a = \vec{b}_p + \omega_r(\vec{b}_{k_1} - \vec{b}_{k_2}), \text{ with } k_1, k_2 \in \mathcal{D}_p^n \ \& \ k_1 \neq k_2 \tag{18}$$

where $\omega_r \in [0, 1]$. A non–uniform mutation scheme with a small user–defined probability is, finally, applied to $\vec{b}_a$. The mutated individual is, then, sent for evaluation to $CPU_{\vec{b}_a}$. Due to its asynchronous operation, this algorithm is suitable for multiprocessor systems with $N_{CPU} \leq N_{agents}$ (even heterogeneous) processors.

The efficiency of an AEA can substantially be improved if metamodels (trained on an appropriate subset of the previously evaluated individuals) are employed. This gives rise to the so–called AMAEA. Metamodels, i.e. on–line trained radial basis function (RBF, Haykin (1999)) networks are used for the IPE of candidate solutions, as mentioned in the intoduction. Inspired by MAEAs, in Asouti *et al.* (2009), the use of metamodels was embedded in the asynchronous EA. Metamodels are activated only after completing and archiving a user–defined minimum number of exact evaluations. From this point on, for each vacant CPU, instead of generating a single individual, N_{IPE} trial ones are generated by the evolution operators applied within $\mathcal{D}_p^n$. For each one of them, a local metamodel is trained on a small number of data selected from the archive of previously evaluated individuals (DB). The training patterns are selected from the DB based on the minimum distance (in the design space) from the trial individual and approximate ("inexact", IPE) fitness values are computed ($\vec{\hat{f}}$) for all of them. The "best" among the N_{IPE} individuals, according to the metamodel, is the one to be re–evaluated by the problem–specific (CFD) tool.

5. The Proposed Asynchronous MAMA (AMAMA)

As already explained, the proposed AMAMA is based on the AMAEA described above, with the additional implementation of LS. Individuals to undergo LS are selected based on dominance criteria applied to a set formed by the just evaluated individual ($\vec{b}_a, \vec{f}(\vec{b}_a)$) and the current front of non–dominated individuals ($\mathcal{P}_a$). Whenever a new individual enters $\mathcal{P}_a$ (i.e. becomes non–dominated), this is automatically selected to undergo LS. Returning from LS, this may displace the current individual according to the Lamarckian learning rules. LS requires the computation of the gradient of the objective function with respect to the design variables ($d\vec{f}/d\vec{b}$), the refinement–update of $\vec{b}_a$ by means of the steepest descent method and, finally, the re–evaluation of the updated individual with the problem–specific (CFD) evaluation tool providing $\vec{f}(\vec{b}_a)$. The refinement is constrained by the user–defined upper and lower bounds of all design variables. Apart from the randomly generated individuals during the starting phase of the method, any other individual may be selected to undergo LS. So, practically, any number of processors may simultaneously undergo LS. LS does not affect the implementation of metamodels.

In SOO problems, the gradient of the objective function, defines the direction of the refinement of $\vec{b}_a$. In MOO problems, the SPEA2 utility function $\phi = \phi(f_1, f_2, ..., f_M)$ defines the descent direction in the objective space. A ϕ value is assigned to each individual by taking a subset of the currently available individuals along with the individual under consideration. In the proposed method, LS aims at improving the current front of

non–dominated solutions with respect to all objectives or, differently stated, that the direction of LS is "perpendicular" to this front. To determine the direction of improvement, $\nabla\phi = \frac{\partial\phi}{\partial\vec{b}_a}$ must be computed and used.

To this end, two basic issues should be addressed. The first one is related to the computation of $\nabla\phi$, i.e. how to overcome the difficulty in computating $\frac{\partial\phi}{\partial\vec{b}_a}$, given that ϕ is a non–differentiable function of $f_i, i=1,M$. The second one deals with the reduction of the gradient computation cost.

A remedy to the fist problem has already been presented in Kampolis and Giannakoglou (2008), where an exact differentiation of an approximation of the non–differentiable Heaviside function is employed. This approximation is used in this paper as well. Based on this, an approximate $\frac{\partial\phi}{\partial f_i}$ value (to be precise, this is the exact derivative of a function approximation) is assigned to each individual undergoing LS. By assuming that $\omega_i = \frac{\partial\phi}{\partial f_i}$ and by the chain rule

$$\nabla\phi = \sum_{i=1}^{M} \frac{\partial\phi}{\partial f_i}\frac{\partial f_i}{\partial\vec{b}_a} = \sum_{i=1}^{M} \omega_i \frac{\partial f_i}{\partial\vec{b}_a} \tag{19}$$

It is obvious that if the adjoint method was utilized to compute the gradient $\frac{\partial f_i}{\partial\vec{b}_a}$ for M objectives, M calls to the adjoint solver would be necessary. Irrespective of the value of N, a single gradient computation with the adjoint method costs approximately as much as the solution of the flow equations. Thus, computing M gradients, updating the current individual using steepest descent and, finally, re–evaluating the updated individual altogether cost as if the flow equations were solved $M+1$ times.

In order to reduce this cost, by taking into consideration Equations (19) and (7), it is proposed to concatenate the M objectives into a scalar function F where the weighting factors are the gradients of the SPEA2 utility function ϕ. By doing so, a single solution of the adjoint equations, with "frozen" ω_i values (equations as in section 3) and ϕ instead of F is sufficient. This leads to reduced computational cost, equal only to two "equivalent" flow solutions (1+1) regardless of the values of M, since the adjoint equations are solved only once.

6. Applications

The proposed AMAMA was applied to two design–optimization problems, namely: (a) the two–objective tube shape optimization of a tube bank heat exchanger and (b) the SOO of a turbomachinery cascade. The first engineering case was studied using all variants of the asynchronous algorithm, namely AEA, AMAEA, AMA and AMAMA and each run was repeated 5 times, with different random number generator (RNG) seeds. The second case (SOO) is used to compare the AMAEA and AMAMA. An additional comparison of AMAEA and AMAMA, on mathematical functions (non–expensive runs which were repeated several times to get an average performance), is shown in Appendix A.

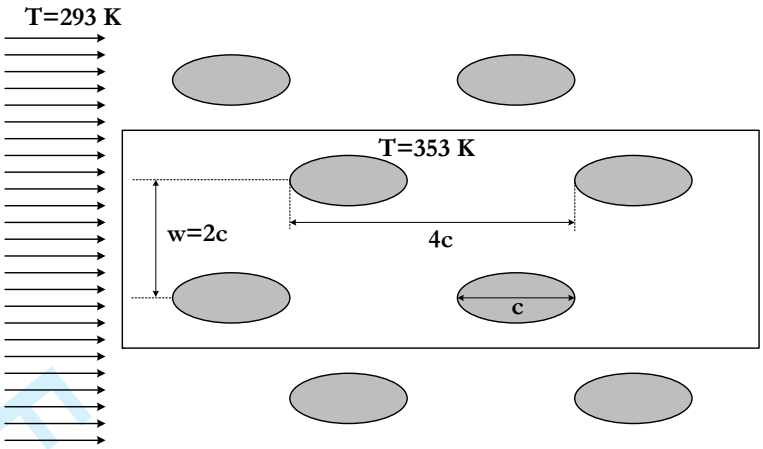


Figure 2.: Schematic representation of a tube bank heat exchanger. The black line confines the computational domain. Upstream and downstream extensions are not in scale.

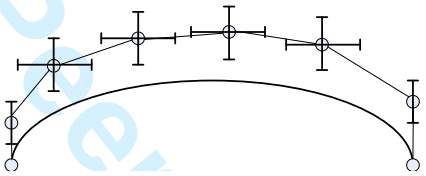


Figure 3.: Design of a tube heat exchanger. Parameterization of the upper side of the tube shape (symmetry). The design variables correspond to the coordinates of the Bézier control points which are marked with a vertical and/or horizontal straight line segment.

6.1. *Design of a tube heat exchanger*

This case is concerned with the two–objective shape optimization of the tubes of a staggered tube bank heat exchanger, for minimum volume–averaged total pressure losses, f_1, and maximum heat exchange, f_2, as in Equation 6, Hilbert *et al.* (2006). Heat exchangers containing banks of tubes in crossflow are widely used in industrial and power engineering applications. This 2D study is physically consistent with the flow over the mid–span plane of heat exchangers with the tube length in the longitudinal direction being much larger than its width, Zdravistch *et al.* (1995). The heat exchanger and the boundaries of the 2D computational domain are shown in Figure 2. Due to the periodicity, the computational domain contains only four tubes. The outlet boundary is extended several chord–lengths downstream the last tubes, not shown in Figure 2. The fluid enters the domain with $T_{inlet} = 293K$ and the flow Reynolds number based on the distance w is equal to $Re = 140$. High temperature fluid flows inside the tubes, ensuring constant wall temperature $T_{wall} = 353K$.

The tube shape is symmetric along the horizontal axis and is parameterized using Bézier–Bernstein polynomials, with 8 control points on each side. 4 of them are allowed to vary in both directions, while the second and the seventh vary only in the normal to chord direction, resulting to 10 design variables in total, as presented in Figure 3. All tube cross sections are identical and located in pre–defined positions. The computational grids are formed by generating structured–like layers of triangles (i.e. quadrilaterals split into triangles) around each tube and, then, by filling in the remaining domain with triangular

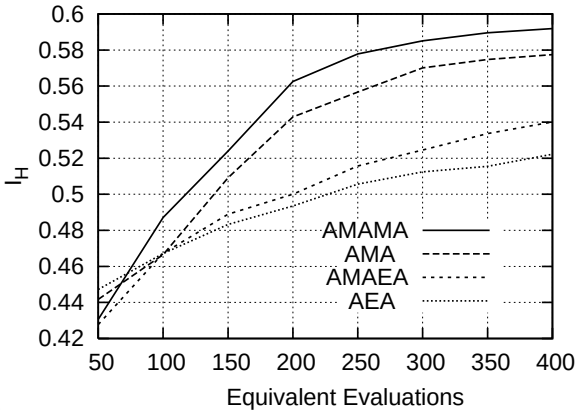


Figure 4.: Design of a tube heat exchanger. Evolution of the mean hypervolume indicator (I_H) of AEA, AMAEA, AMA and AMAMA. The hypervolume indicator I_H indicates the area dominated by the front of the non–dominated individuals; higher I_H values correspond to fronts closer to the Pareto front.

elements using the advancing front technique. This results to grids of ~ 85000 nodes and ~ 170000 triangular elements on the average.

This case was studied using all variants of the asynchronous algorithm, namely AEA, AMAEA, AMA and AMAMA with a 10×10 supporting mesh on 40 CPUs. Regarding the metamodel–based variants, $N_{IPE} = 7$ trial individuals were pre–evaluated before proceeding to the CFD–based evaluation and this occured only after having the first 50 entries recorded in the DB. With all variants, 5 runs with different RNG seeds were carried out. The same 5 RNG seed values were used but, as explained elsewhere in the paper, this guarantees nothing more that the starting populations were all the same. The evolution of the average (over the 5 runs) mean hypervolume indicator (I_H) is shown in Figure 4. It is obvious that AMAMA outperforms all other variants. From the same figure, the gain from using memetic algorithm, with or even without the extra assistance by the metamodels, can be seen.

For a selected AMAMA run, Figure 5 presents the computed front of non–dominated solutions at the cost of 400 CFD evaluations; in Figure 6 three designs (tube shapes), selected from this front, are shown. Out of the 400 evaluations of the AMAMA, 180 correspond to evaluations of the objective functions and 110 to LS actions, i.e. 110 gradient computations (solutions of adjoint equations) and 110 re–evaluations of the objective functions (solutions of state equations). 73 out of the 110 LS attempts led to an improved solution that entered the front of non–dominated solutions by the time they returned. 71.6% of the non–dominated individuals of the final front (i.e. 43 out of 60) resulted directly from LS, which confirms the important contribution of the LS in the algorithm. A close–up view of the computational grid around one of the four tubes, that corresponds to one among the computed optimal solutions of Figure 6, is shown in Figure 7.

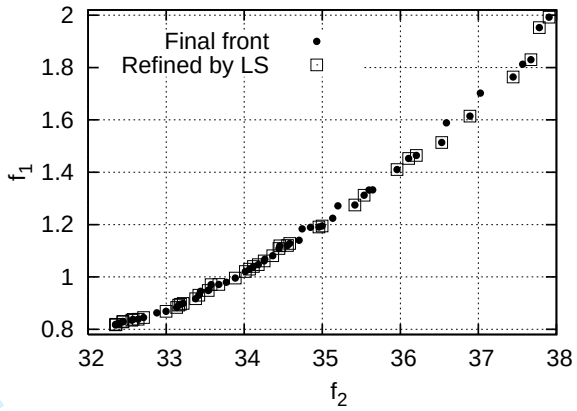
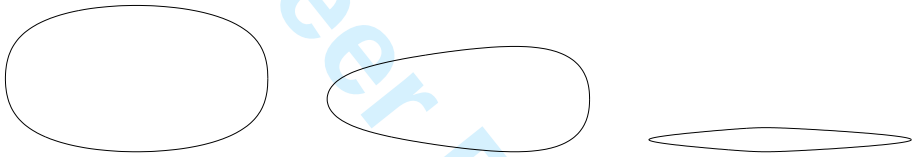


Figure 5.: Design of a tube heat exchanger. Front of non–dominated individuals at the cost of 400 evaluations. Front members resulted from a LS action are marked with an empty square. f_1 and f_2 stand for the volume–averaged total pressure losses and the exchanged heat, respectively, as in Equation 6.



(a) f_1=1.99, f_2=37.9 (b) f_1=1.46, f_2=36.2 (c) f_1=0.82, f_2=32.4

Figure 6.: Design of a tube heat exchanger. Three tube shapes corresponding to three non-dominated solutions, selected from Figure 5: (a) is a tube shape yielding maximum heat exchange with high total pressure losses, (c) is the other way round. (a) and (c) practically correspond to the edges of the Pareto front in Figure 5. Finally, (b) is a solution in the middle of the front.

6.2. *Design of a turbomachinery cascade airfoil*

The second engineering case is concerned with the design of a turbomachinery cascade airfoil for minimum volume–averaged total pressure losses, f_1, as in Equation 6. The cascade has fixed stagger angle equal to 35^o and fixed pitch-to-chord ratio equal to 0.6. It was designed for inlet flow angle $a_1 = 52^o$ and Reynolds number based on chord $Re_c = 9 \times 10^5$. The airfoil shape was parameterized using the Bézier–Bernstein polynomials with 8 control points on each side. 6 of them were allowed to vary, summing up to $12 + 12 = 24$ design variables. Figure 3, associated with the previous case, can also be used to describe the parameterization of the cascade airfoil. Geometrical constraints on the airfoil thickness t were imposed as follows

$$t(0.25c) \geq 0.05c,\ t(0.50c) \geq 0.045c,\ t(0.85c) \geq 0.017c$$

where c is the chord length. In addition, the minimum flow turning angle was constrained to $a_1 - a_2 \geq 22^o$.

This case was studied using AMAEA and AMAMA on 20 and 40 CPUs, both with a

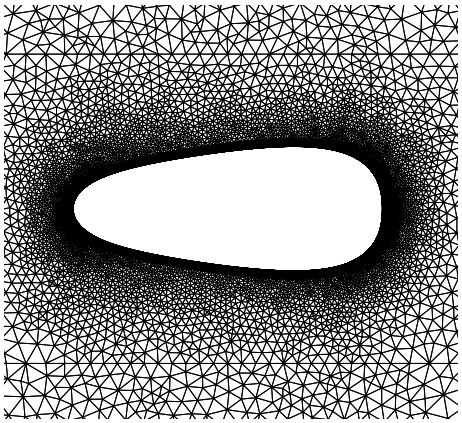


Figure 7.: A close–up view of the computational grid around one of the four tubes that corresponds to solution (b) of Figure 6.

10×10 supporting mesh. The metamodel is activated after 50 evaluations with $N_{IPE}=8$ trial individuals. Figure 8 compares the convergence histories of AMAEA and AMAMA and marks all current best solutions resulted from LS, on 20 CPUs and 40 CPUs. It is obvious that AMAMA performs constantly better than AMAEA during the evolution irrespective of the number of CPUs.

Statistics about the performance of LS on 20 and 40 CPUs can be found in Table 1. On 20 CPUs, the 300 equivalent evaluations of the AMAMA comprise 266 evaluations of the objective function and 17 LS actions i.e. 17 gradient computations and 17 re–evaluations. All of the LS actions undertaken were successful, in the sense that the outcome of each LS was better than the individual undergoing LS. However, it is more important to investigate whether the outcomes of LS actions become the best–so–far individual by the time they return or other processors have likely returned better individuals in the meantime. Based on the statistics of Table 1, 42% of the individuals undergoing LS on 20 CPUs became the best–so–far individual by the time they returned. The corresponding percentage on 40 CPUs was 55%.

According to Table 1 and Figure 8, as the number of CPUs increases, less LS actions are performed and this affects the best objective function value achieved within the affordable CPU cost. This confirms similar findings in the SOO mathematical case (Ackley function; see Appendix A), where (due to its low CPU cost) each run was repeated 30 times with different RNG seeds on various multiprocessor systems. According to these studies, there is a maximum number of processors above which the LS actions cease to be effective.

The optimum design obtained by AMAMA on 20 CPUs and a close–up view of the computational grid in the vicinity of the leading edge area of the optimum design are shown in Figure 9.

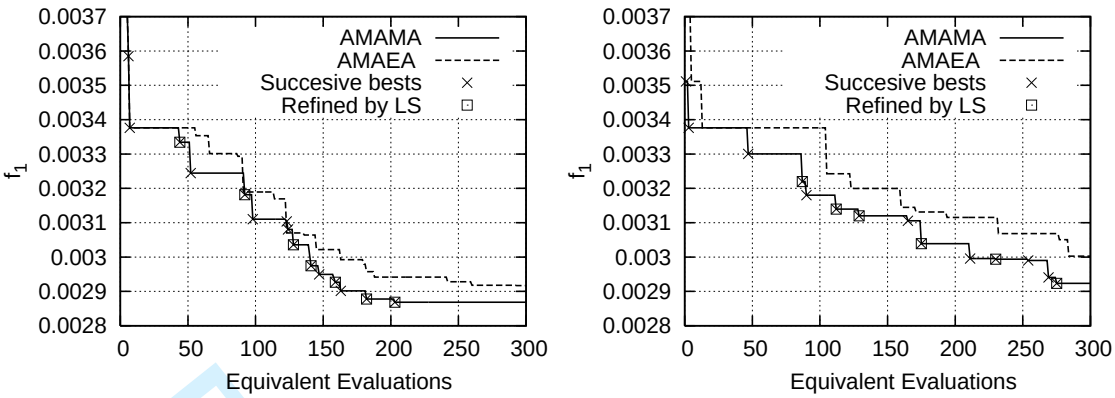


Figure 8.: Design of a turbomachinery cascade. Convergence history of AMAEA and AMAMA on 20 CPUs (left) and 40 CPUs (right) and LS refinements (square symbols) compared all successive bests (x symbols) during the evolution on 20 CPUs (left) and 40 CPUs (right). f_1 represents the volume–averaged total pressure losses as in Equation 6.

Table 1.: Design of a turbomachinery cascade. Statistics regarding the performance of LS on 20 CPUs and 40 CPUs.

	20 CPUs	40 CPUs
Equivalent Evaluations	300	300
Number of LS actions	17	11
Improved LS actions	17	11
Bests refined by LS	7/17	6/11

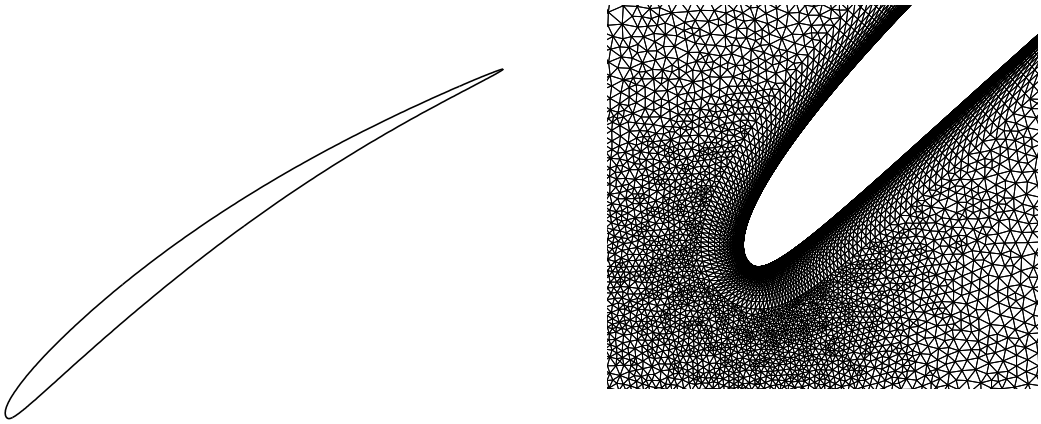


Figure 9.: Design of a turbomachinery cascade. The optimum design obtained by AMAMA on 20 CPUs (left) and a close–up view of the computational grid close to the leading edge area of the optimum design (right).

7. Conclusions

This paper extended a well performing asynchronous evolutionary algorithm (AEA), devised in the past for use without (Asouti and Giannakoglou (2009)) or with (Asouti

et al. (2009)) assistance by surrogate evaluation models (metamodels). The new extension includes:

- the additional implementation of local search (LS) for the most promising individuals, which transformed the AEA to an asynchronous memetic algorithm (AMA),
- the use of metamodels during the global search task, which upgraded the AMA to an asynchronous metamodels-assisted MA (AMAMA) and
- a "smart" formulation of the LS process based on a single call to the adjoint method computing the gradient of the objective function, which additionally reduces the wall clock time of the optimization in problems with more than one objectives.

Regarding the latter, this paper proposed a way to handle a scalar (rather than a vector) objective function during LS. The scalar function to be minimized is the synthesis of the MOO objective functions, multiplied by appropriate coefficients which ensure that steepest descent will provide non-dominated solutions by moving "normal" to the front of current non-dominated solutions. The gradient of the non-differentiable SPEA2 function used to quantify the quality of current solutions based on dominance criteria was computed by approximating this function with a continuous one and, then, differentiating the latter, as originally proposed by in Kampolis and Giannakoglou (2008). In contrast to this previous work, instead of solving as many adjoint equations as the objectives, computing their gradients and combining them to find the gradient of the LS scalar function, a low-cost scheme was proposed, which sets up this scalar function (with frozen coefficients) and then solves a single adjoint equation to compute its gradient.

A by-product of the proposed method was the formulation and programming of the continuous adjoint method for incompressible fluid flows with heat transfer. Though the energy equation is fully decoupled from the momentum and mass conservation equations, their adjoint equations are fully coupled.

The solution of the presented case studies was carried out on multiprocessor platforms, where the asynchronous search method (EA or MA) presents the advantage of reducing (almost eliminating) the idle times of CPUs. This was achieved by "immediately" assigning a new evaluation on any CPU that becomes idle after completing a previous evaluation and by eliminating the synchronization barrier (end-of-generation) of synchronous EAs or MAs. Based on the studied cases, the memetic algorithm, with or without the extra use of metamodels, performs better than both AEA and AMAEA. Based on the findings of (Asouti and Giannakoglou (2009)) and (Asouti *et al.* (2009)), we may conclude that the later outperforms (synchronous) EAs and MAEAs.

Appendix A. Mathematical Benchmarks

Two mathematical benchmarks, the Ackley and ZDT3 functions, were also used to compare the AMAEA and AMAMA. Due to their low CPU cost, each run was repeated 30 times, with different RNG seeds and the average performance of these runs is plotted. For the needs of the AMAMA, the derivatives of the objective function(s) are computed analytically but this computation is assumed to cost as much as the computation of the objective function(s), i.e. as if the adjoint method was used.

The minimization of the Ackley function with $N = 20$ optimization variables is presented first. The average behaviour of 30 runs with stopping criterion of 3000 evaluations each is of concern. 5, 20 and 40 CPUs were used, by performing 90 runs in total. In all cases, a 8×8 supporting mesh was used and the metamodels, in both AMAEA

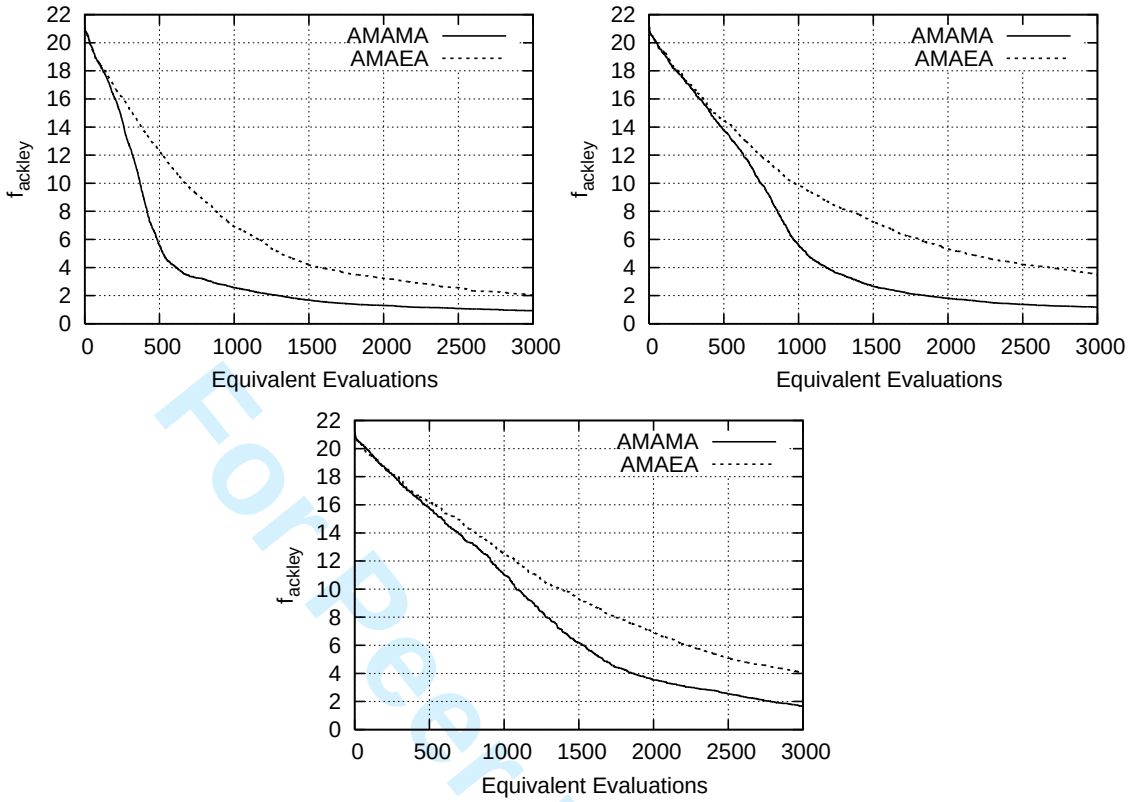


Figure A1.: Minimization of the Ackley function. The mean convergence histories of AMAEA and AMAMA on 5, 20 and 40 CPUs.

and AMAMA, employed after collecting the first 100 entries in the DB, by generating $N_{IPE} = 3$ trial individuals for each new individual. The mean convergence histories of the 30 runs of AMAEA and AMAMA on 5, 20 and 40 CPUs are presented in Figure A1.

Table A1 summarizes the mean, minimum and maximum values as well as the standard deviation of the objective function of all the aforementioned computations; the t_0 values from the t–tests between AMAMA–AMAEA are also shown. Irrespective of the number of CPUs, from the t–tests performed between AMAMA–AMAEA, it is clear that AMAMA performs better than AMAEA. In particular, the t_0 values from the comparison between AMAMA and AMAEA for 5 ($t_0 = 6.367$), 20 ($t_0 = 9.828$) and 40 CPUs ($t_0 = 15.465$) ensure that AMAMA is significantly better than AMAEA.

Table A1 shows also the mean number of LS actions performed by the AMAMA. As the number of CPUs increases, less LS actions are performed and this affects the mean objective function value achieved within the affordable CPU cost (Table A1). One may comment on the outcome of runs on various multiprocessor systems, according to which the AMAMA performance worsens above a certain number of CPUs. Since the refinement of any individual has a CPU cost of approximately two evaluations, $2(N_{CPU} - 1)$ evaluations on the average are expected to end in the meantime, assuming that no other CPU simultaneously undergoes LS. By increasing N_{CPU}, it is more likely that among the $\sim 2(N_{CPU} - 1)$ individuals evolved and evaluated in the meantime, at least one dominates the outcome of LS. Should this be the case, this LS action, unfortunately, becomes ineffective and this reflects on the results illustrated in Figure A1.

The second mathematical benchmark considered is the two-objective minimization of

Table A1.: Minimization of the Ackley function. Comparison of AMAEA and AMAMA on 5, 20 and 40 CPUs.

	5 CPUs		20 CPUs		40 CPUs	
	AMAEA	AMAMA	AMAEA	AMAMA	AMAEA	AMAMA
f_{mean}	2.037	0.933	3.481	1.175	4.067	1.662
f_{min}	0.167	0.019	1.992	0.055	2.567	0.686
f_{max}	4.204	2.013	6.679	2.498	5.309	2.510
s	0.663	0.680	1.032	0.766	0.683	0.509
t_0		6.367		9.828		15.465
LS		107.4		97.6		61.2

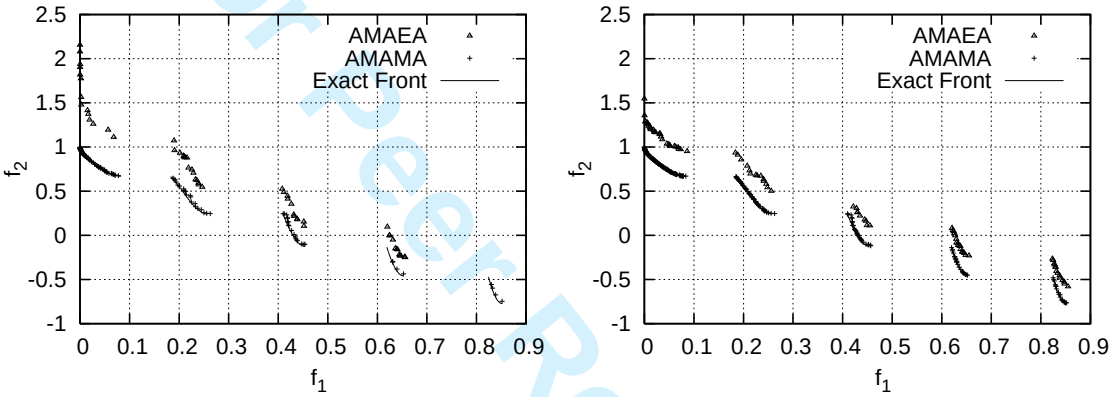


Figure A2.: Minimization of the ZDT3 function. Comparison of the (exact) Pareto front with the fronts of non–dominated individuals that correspond to the run with highest hypervolume indicator (I_H) values, among the 30 runs computed using AMAEA and AMAMA at the cost of 2000 (left) and 3000 (right) evaluations. After 3000 evaluations, the AMAMA front can hardly be distinguished from the exact front.

the ZDT3 function, Zitzler *et al.* (2002), with $N = 30$ optimization variables. For this case, a 6×6 supporting mesh, 20 CPUs and a stopping criterion of 3000 evaluations were used. As before, 30 AMAEA and AMAMA runs with different RNG seeds were performed. For both, the use of metamodel was initiated after collecting 500 entries in the DB, with $N_{IPE} = 4$ trial individuals. Figure A2 presents the fronts of non–dominated individuals for AMAEA and AMAMA after 2000 and 3000 evaluations. These correspond to the best run, according to the hypervolume indicator (I_H), among the 30 runs with different RNG seeds. After 2000 evaluations, the front computed by the AMAMA is significantly better than that of AMAEA and lies, practically, on the exact Pareto front. After 3000 evaluations, the AMAMA front is enriched with more members and can hardly be distinguished from the (exact) Pareto front whereas the AMAEA has not yet reached the exact Pareto front.

These findings are also justified by the hypervolume indicator, which was computed with an arbitrary reference point, $(f_1, f_2) = (1, 5)$. The mean value and standard deviation of I_H of the 30 runs are shown in Table A2, along with the t_0 value from the t–test between AMAMA–AMAEA. Once more, it is absolutely clear that the AMAMA performs better than the AMAEA.

Table A2.: Minimization of the ZDT3 function. Comparison of the mean value and standard deviation of the hypervolume indicator I_H of the 30 runs along with the t_0 value from the t–test between AMAMA vs. AMAEA.

	AMAEA	AMAMA	Exact Front
I_H	4.760	4.790	4.803
s	0.023	0.074	-
t_0		2.210	

References

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

Alba, E. and Troya, J., 2001. Analyzing synchronous and asynchronous parallel distributed genetic algorithms. *Future Generation Computer Systems*, 4, 451–465.

Anderson, W., Rausch, R., and Bonhaus, D., 1995. Implicit/Multigrid Algorithms for Incompressible Turbulent Flows on Unstructured Grids. *In*: *12th AIAA Computational Fluid Dynamics Conference and Open Forum* AIAA-1995-1740, San Diego, CA, USA, 1067–1081.

Anderson, W. and Venkatakrishnan, V., 1997. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *In*: *35th Aerospace Sciences Meeting and Exhibit* AIAA-1997-0643, Reno, NV, USA.

Asouti, V. and Giannakoglou, K., 2009. Aerodynamic optimization using a parallel asynchronous evolutionary algorithm controlled by strongly interacting demes. *Engineering Optimization*, 41 (3), 241–257.

Asouti, V., Kampolis, I., and Giannakoglou, K., 2009. A Grid-Enabled Asynchronous Metamodel-Assisted Evolutionary Algorithm for Aerodynamic Optimization. *Genetic Programming and Evolvable Machines (SI:Parallel and Distributed Evolutionary Algorithms, Part One)*, 10 (3), 373–389.

Bull, L., 1999. On Model-Based Evolutionary Computation. *Soft Computing — A Fusion of Foundations, Methodologies and Applications*, 3 (2), 76–82.

Cantú-Paz, E., 1998. A survey of parallel genetic algorithms. *Calculateurs Paralleles, Reseaux et Systemes Repartis*, 10, 141–171.

Dawkin, R., 1976. *The Selfish Gene*. Oxford University Press.

Désidéri, J. and Janka, A., 2003. Hierarchical Parametrization for Multilevel Evolutionary Shape Optimization with Application to Aerodynamics. *In*: *EUROGEN 2003, Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems*, Barcelona (Spain).

Duta, M., Giles, M., and Campobasso, M., 2002. The harmonic adjoint approach to unsteady turbomachinery design. *International Journal for Numerical Methods in Fluids*, 40 (3-4), 323–332.

Duvigneau, R., Chaigne, B., and Désidéri, J., Multi-level parameterization for shape optimization in aerodynamics and electromagnetics using a particle swarm optimization algorithm. , 2006. , Research Report RR-6003, INRIA.

Elliot, J. and Peraire, J., 1996. Aerodynamic design using unstructured meshes. *In*: *27th Fluid Dynamics Conference* AIAA-1996-1941, New Orleans, LA, USA.

Georgopoulou, C. and Giannakoglou, K., 2009. A Multi-Objective Metamodel-Assisted

Memetic Algorithm with Strength-based Local Refinement. *Engineering Optimization*, 41 (10), 909–923.

Giannakoglou, K., Giotis, A., and Karakasis, M., 2001. Low-cost genetic optimization based on inexact pre-evaluations and the sensitivity analysis of design parameters. *Inverse Problems in Engineering*, 9, 389–412.

Giles, M. and Pierce, N., 1997. Adjoint equations in CFD: duality, boundary conditions and solution behaviour. *In*: *13th AIAA Computational Fluid Dynamics Conference* AIAA-1997-1850, nowmass Village, CO, USA.

Hart, W., 1994. *Adaptive Global Optimization with Local Search*. San Diego, USA: PhD thesis, University of California.

Haykin, S., 1999. *Neural networks: A comprehensive foundation*. New Jersey, USA: Prentice Hall.

Hilbert, R., *et al.*, 2006. A multi–objective shape optimization of a heat exchanger using parallel genetic algorithms. *Interational Journal of Heat and Mass Transfer*, 49, 2567–2577.

Jameson, A., 1988. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3, 233–260.

Kampolis, I. and Giannakoglou, K., 2008. A multilevel approach to single- and multiobjective aerodynamic optimization. *Computer Methods in Applied Mechanics and Engineering*, 197 (33-40), 2963–2975.

Kampolis, I. and Giannakoglou, K., 2009. Distributed Evolutionary Algorithms with Hierarchical Evaluation, Engineering Optimization. *Engineering Optimization*, 41 (11), 1037–1049.

Kampolis, I. and Giannakoglou, K., 2011. Synergetic use of different evaluation, parameterization and search tools within a multilevel optimization platform. *Applied Soft Computing*, 11, 645–651.

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

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

Knowles, J. and Corne, D., 2000. M-PAES: A memetic algorithm for multiobjective optimization. *In*: *2000 Congress on Evolutionary Computation – CEC '00*, September, San Diego, CA. NewYor, 325–332.

Krasnogor, N. and Smith, J., 2005. A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IEEE Transactions on Evolutionary Computation*, 9 (5), 474–488.

Liakopoulos, P., Kampolis, I., and Giannakoglou, K., 2008. Grid-enabled, hierarchical distributed metamodel-assisted evolutionary algorithms for aerodynamic shape optimization. *Future Generation Computer Systems*, 24 (7), 701–708.

Lim, D., *et al.*, 2007. Efficient Hierarchical Parallel Genetic Algorithms using Grid computing. *Future Generation Computer Systems*, 23 (4), 658–670.

Luna, F., Nebro, A., and Alba, E., 2006. Observations in using Grid-enabled technologies for solving multi-objective optimization problems. *Parallel Computing*, 32 (5–6), 377–393.

Melab, N., Cahon, S., and Talbi, E., 2006. Grid computing for parallel bioinspired algorithms. *Journal of Parallel and Distributed Computing*, 66 (8), 1052–1061.

Nowostawski, M. and Poli, R., 1999. Parallel genetic algorithm taxonomy. *In*: *Third International Conference on Knowledge-based Intelligent Information Engineering Systems*

KES'99.

Ong, Y. and Keane, A., 2004. Meta-Lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Computation*, 8 (2), 99–110.

Ong, Y., *et al.*, 2006. Classification of adaptive memetic algorithms: a comparative study. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 36 (1), 141–152.

Papadimitriou, D. and Giannakoglou, K., 2007. A continuous adjoint method with objective function derivatives based on boundary integrals for inviscid and viscous flows. *Computers & Fluids*, 36 (2), 325–341.

Papadimitriou, D. and Giannakoglou, K., 2008. Aerodynamic shape optimization using adjoint and direct approaches. *Archives of Computational Methods in Engineering*, 15 (4), 447–488.

Papadimitriou, D. and Giannakoglou, K., 2009. The continuous direct-adjoint approach for second order sensitivities in viscous aerodynamic inverse design problems. *Computers & Fluids*, 38 (8), 1539–1548.

Pierret, S. and Van den Braembussche, R., 1999. Turbomachinery Blade Design using a Navier-Stokes Solver and Artificial Neural Network. *Journal of Turbomachinery*, 121 (2), 326–332.

Pironneau, O., 1974. On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, 64, 97–110.

Poloni, C., *et al.*, 2000. Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 186 (2-4), 403–420.

Sefrioui, M. and Périaux, J., 2000. A hierarchical genetic algorithm using multiple models for optimization. *In*: M. Schoenauer, K. Deb and et al., eds. *Proceedings of the 6th international conference on parallel problem solving from nature (PPSN VI). Lecture Notes in Computer Science.*, Vol. 1917 Paris: Springer-Verlag, 879–888.

Spalart, P. and Allmaras, S., 1994. A one–equation turbulence model for aerodynamic flows. *La Recherche Aérospatiale*, 1, 5–21.

Zdravistch, F., Fletcher, A., and Behnia, M., 1995. Numerical laminar and turbulent fluid flow and heat transfer predictions in tube banks. *International Journal of Numumerical Methods for Heat & Fluid Flow*, 5 (8), 717–733.

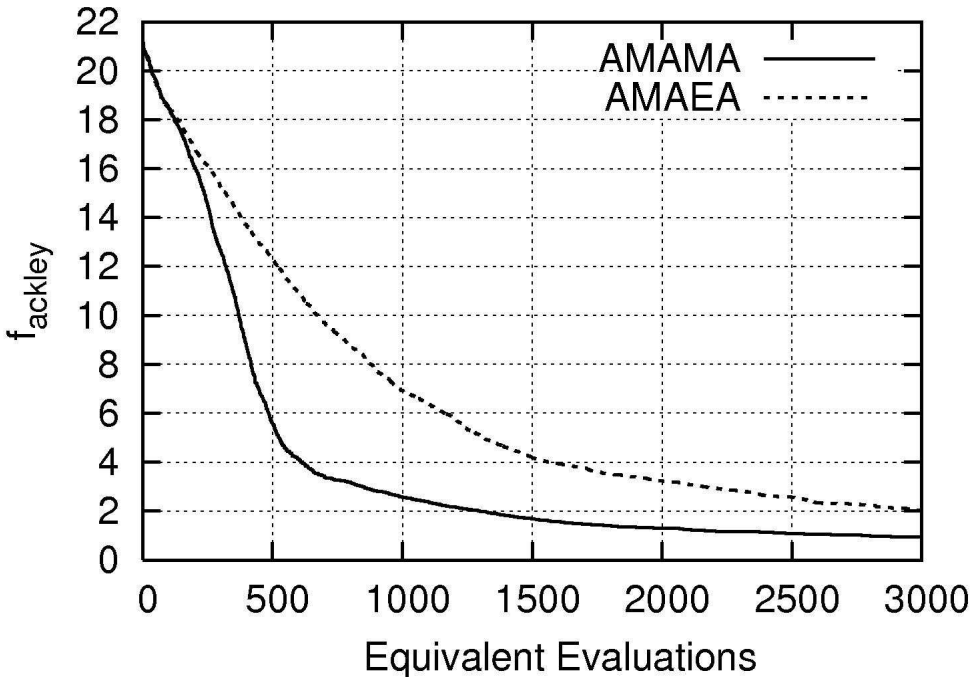
Zhou, Z., *et al.*, 2007a. Memetic algorithm using multi-surrogates for computationally expensive optimization problems. *Journal of Soft Computing*, 11 (10), 957–971.

Zhou, Z., *et al.*, 2007b. Combining global and local surrogate models to accelerate evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 37 (1), 66–76.

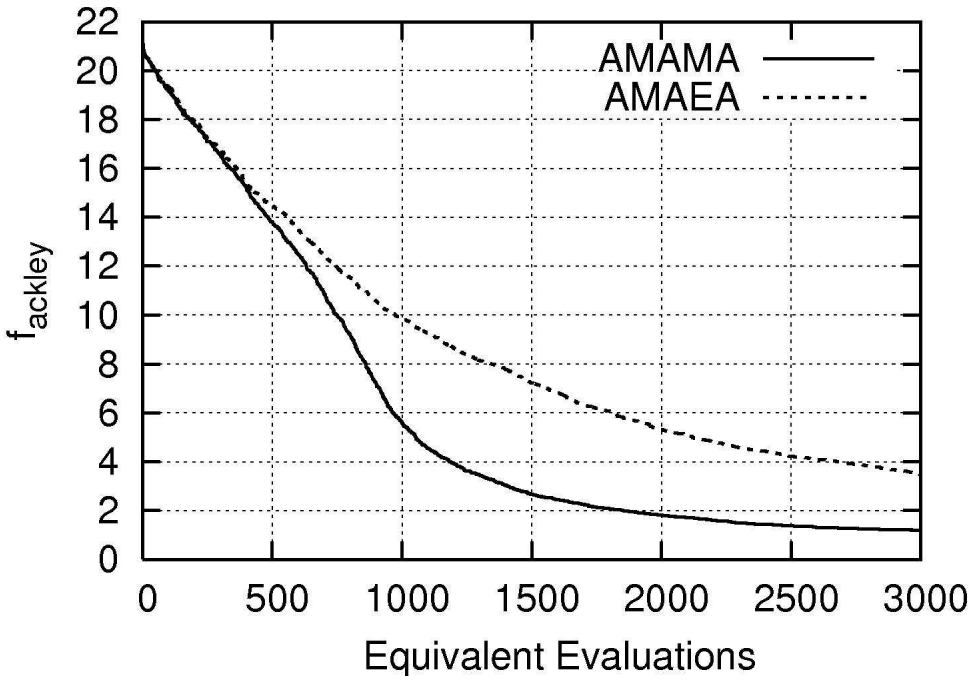
Zitzler, E., Deb, K., and Thiele, L., 2002. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8, 173–195.

Zitzler, E., Laumanns, M., and Thiele, L., 2001. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. *In*: *EUROGEN 2001, Evolutionary methods for design, optimisation and control with application to industrial problems*, Athens.

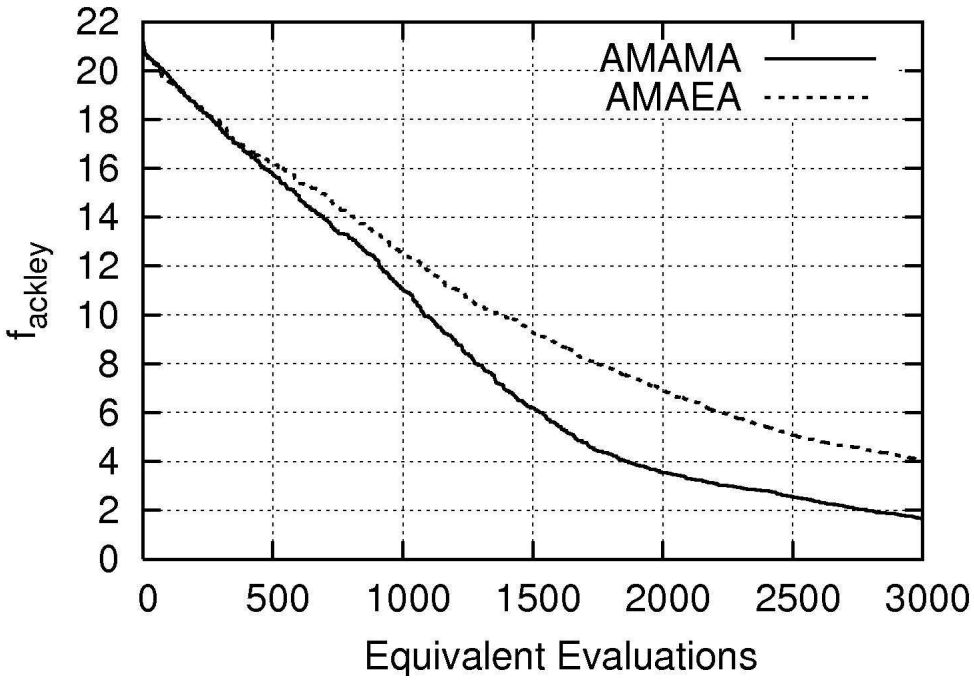
Zymaris, A., *et al.*, 2009. Continuous Adjoint Approach to the Spalart-Allmaras Turbulence Model for Incompressible Flows. *Computers & Fluids*, 38 (8), 1528–1538.



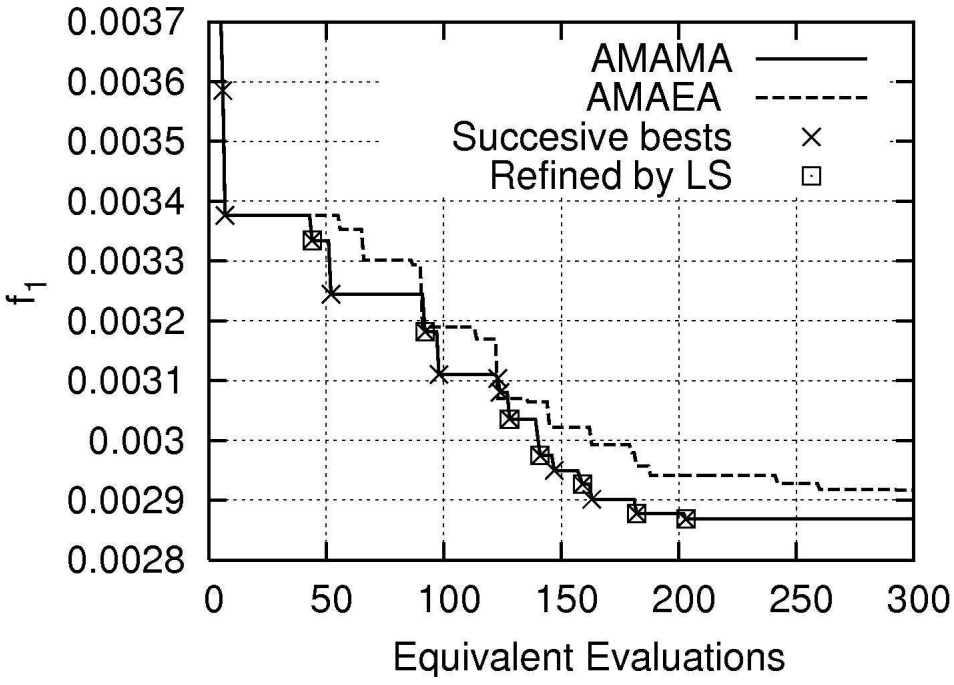
63x44mm (600 x 600 DPI)



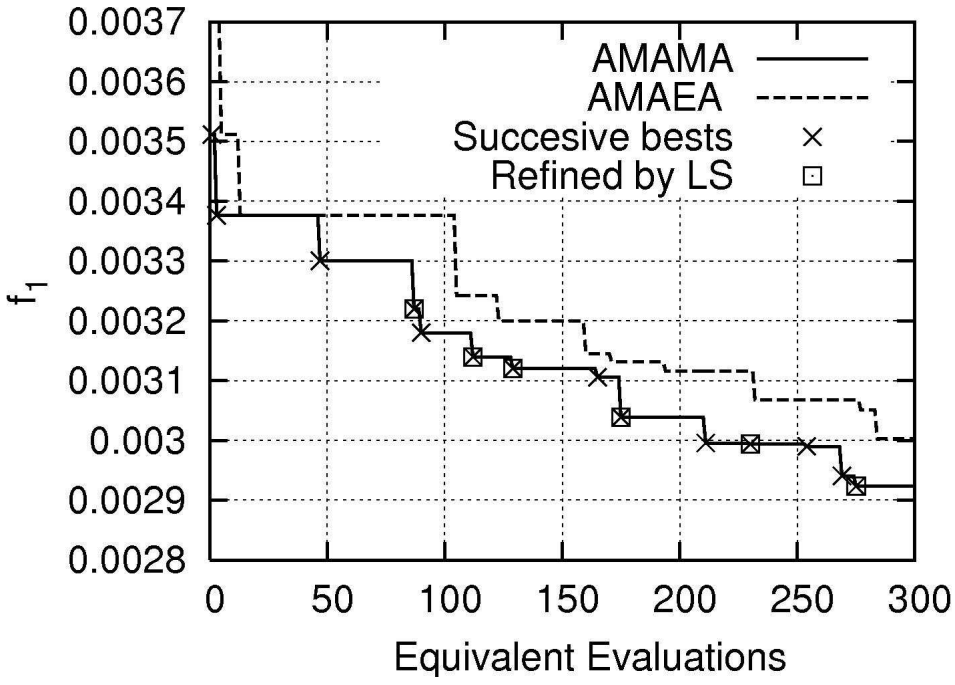
63x44mm (600 x 600 DPI)



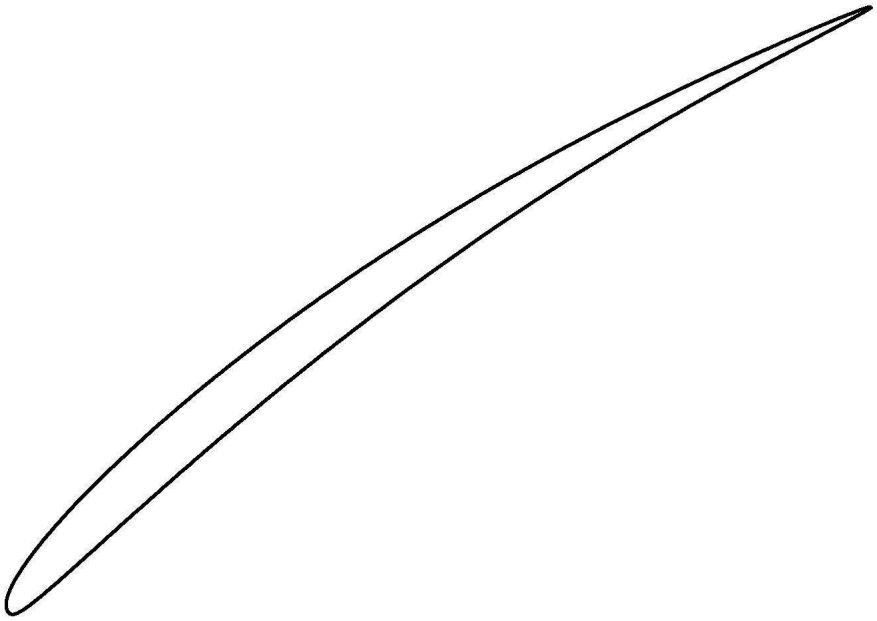
63x44mm (600 x 600 DPI)



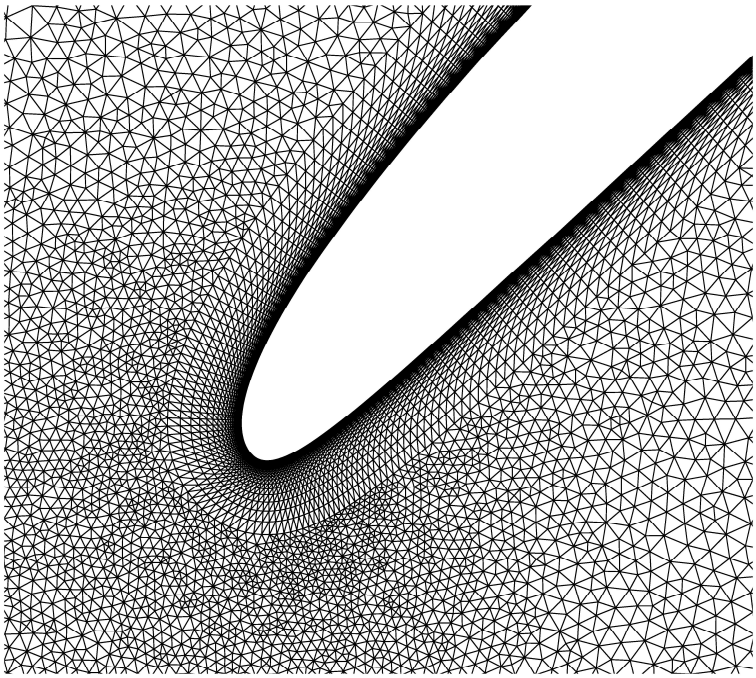
63x44mm (600 x 600 DPI)



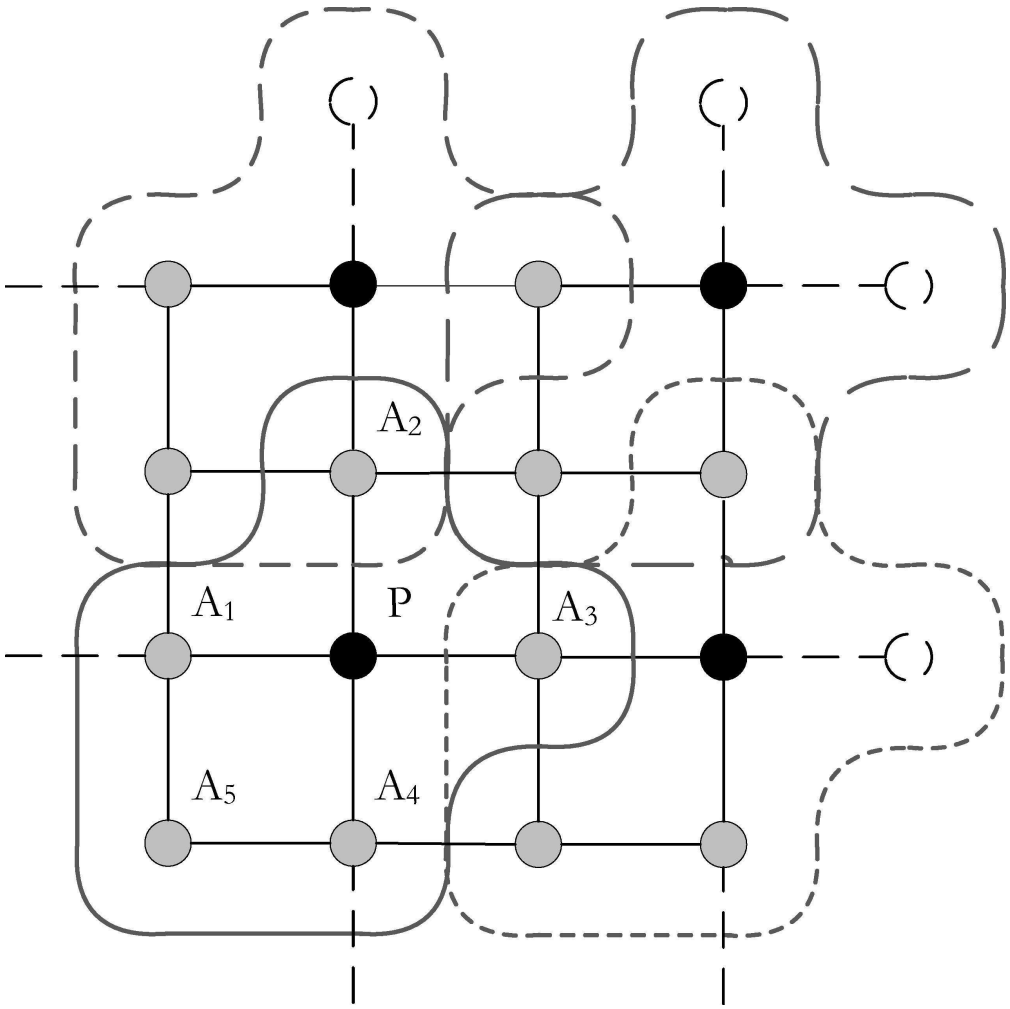
63x44mm (600 x 600 DPI)



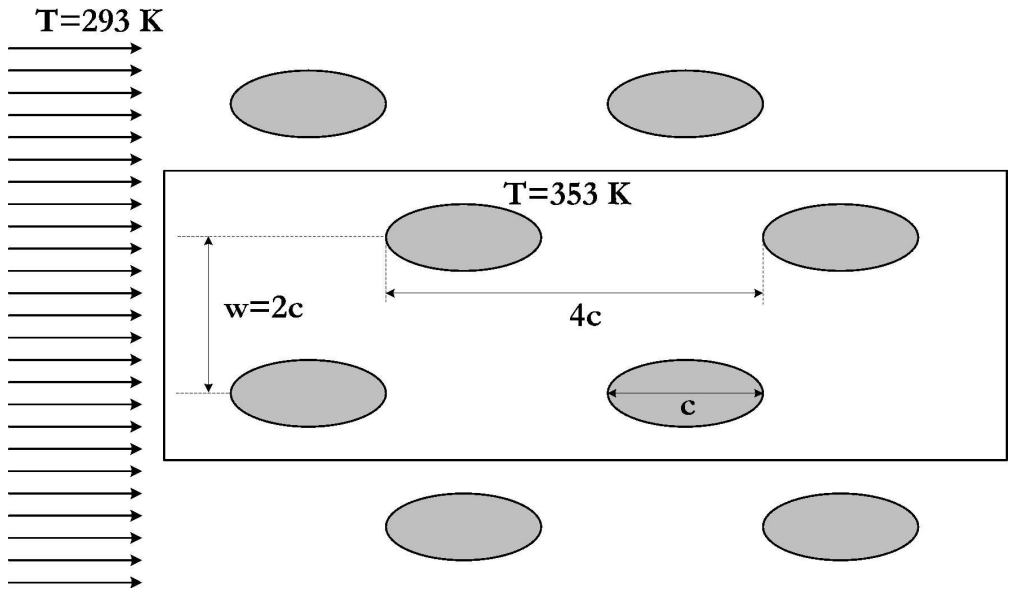
63x44mm (600 x 600 DPI)



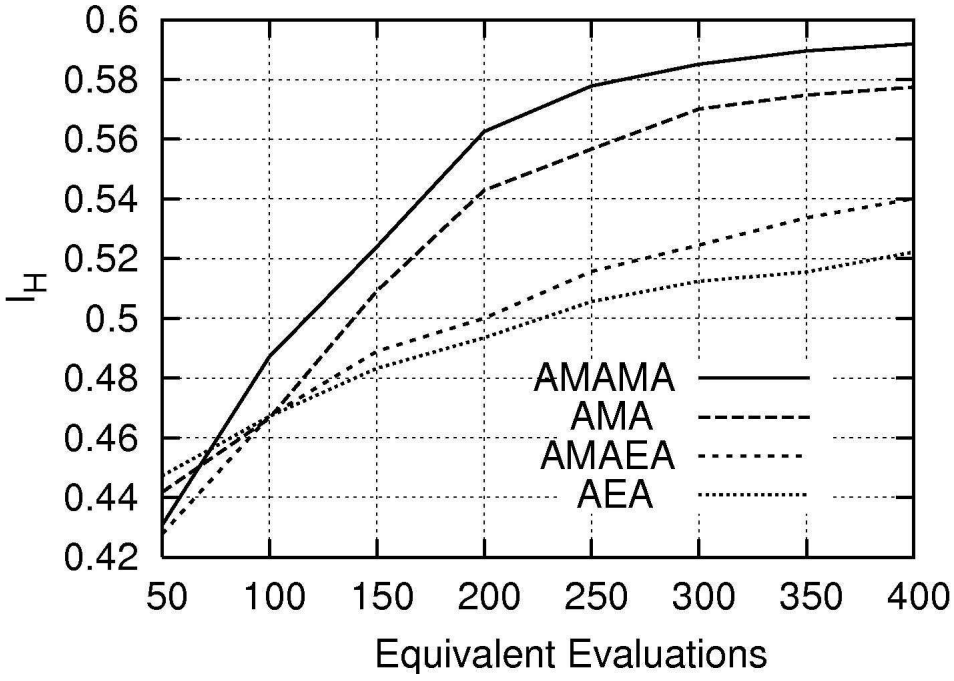
229x203mm (600 x 600 DPI)



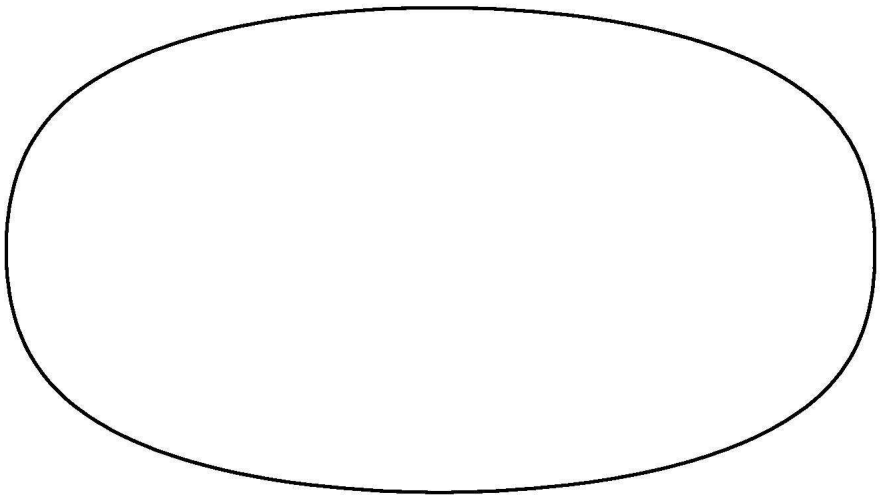
108x107mm (600 x 600 DPI)



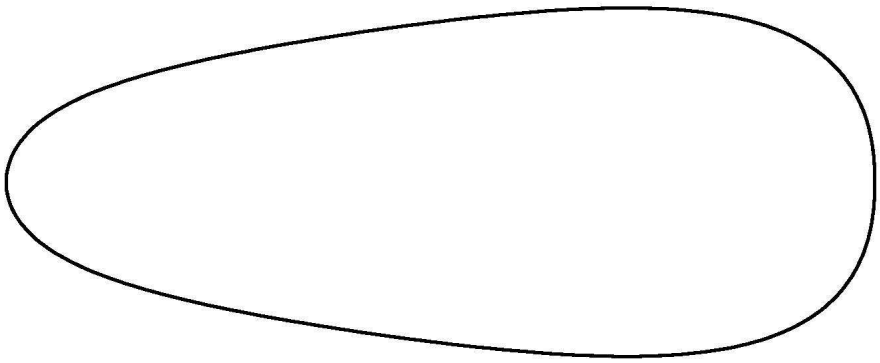
225x134mm (600 x 600 DPI)



63x44mm (600 x 600 DPI)

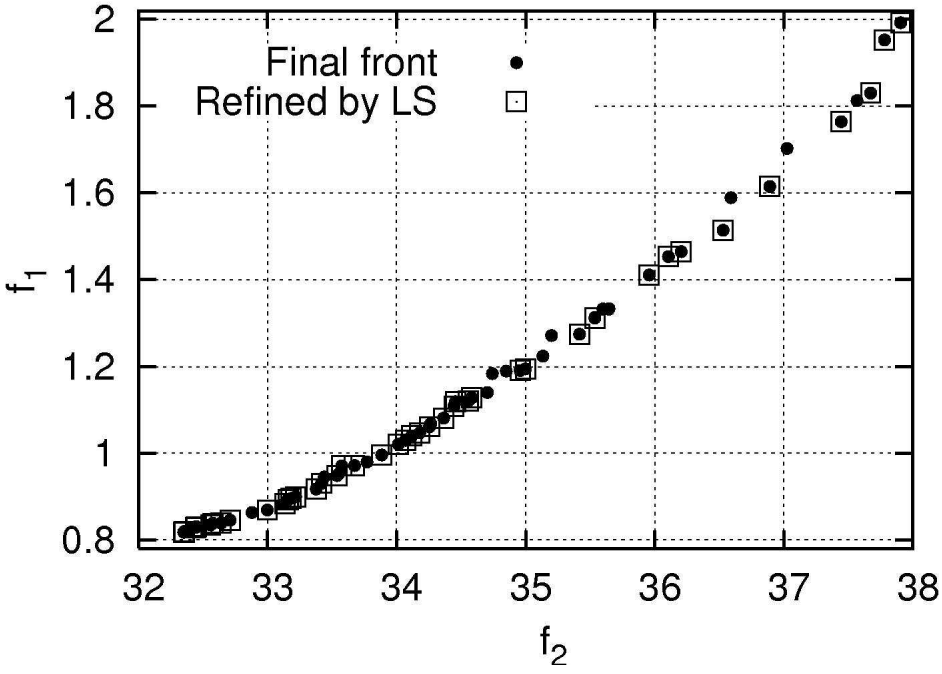


63x44mm (600 x 600 DPI)

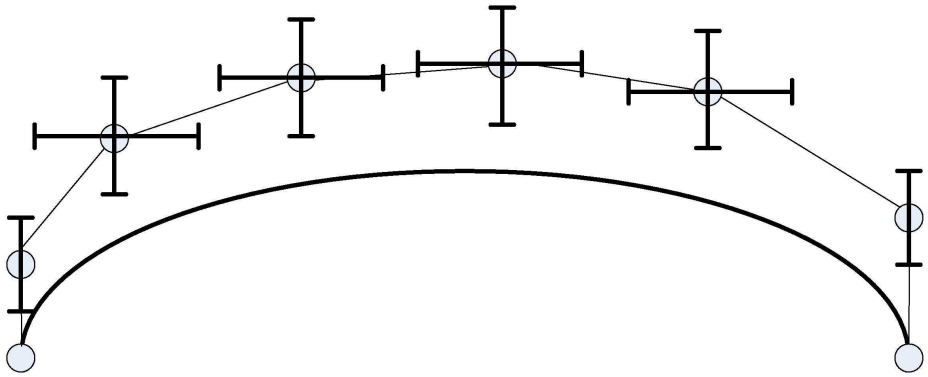


63x44mm (600 x 600 DPI)

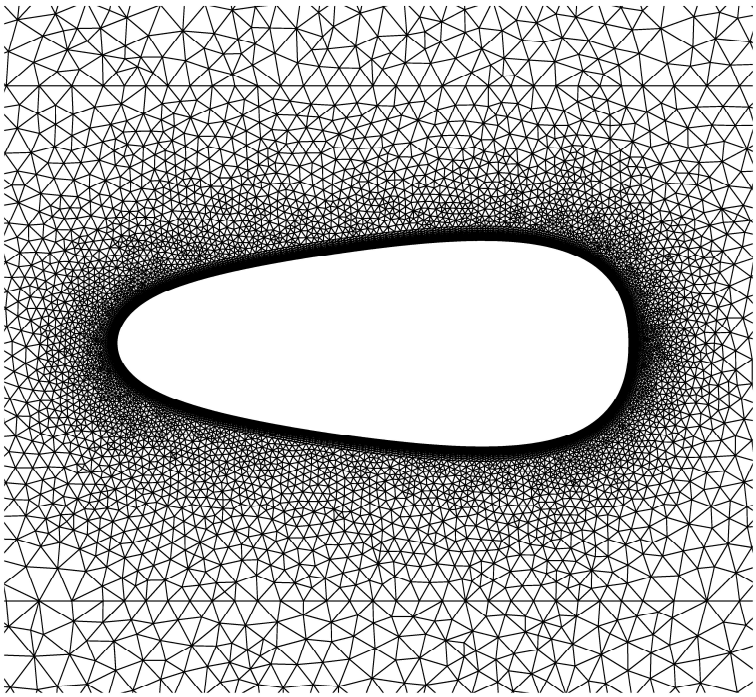
63x44mm (600 x 600 DPI)



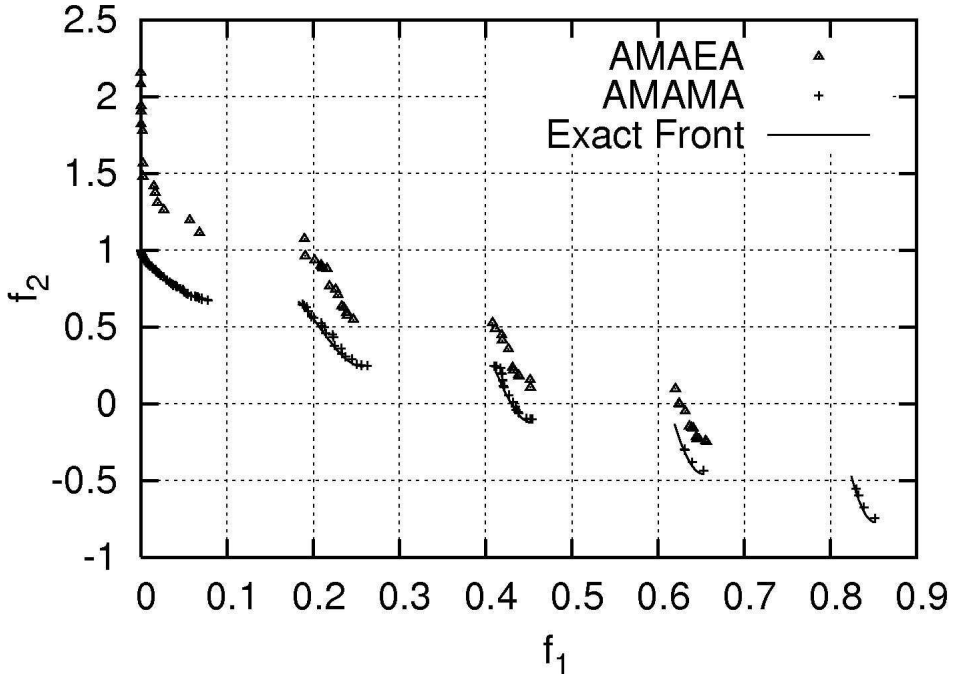
63x44mm (600 x 600 DPI)



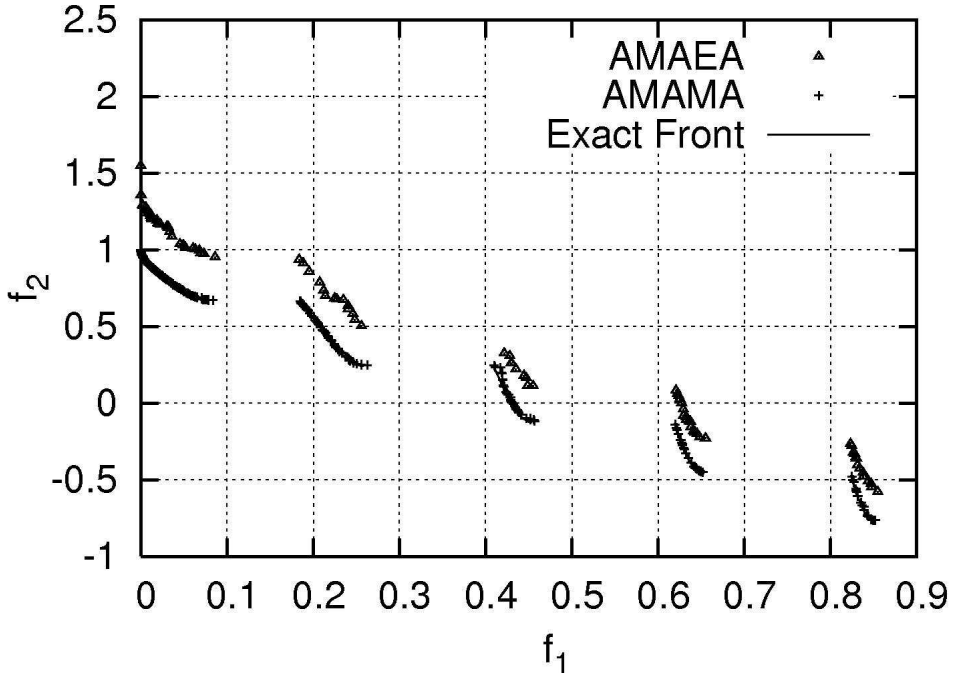
107x43mm (600 x 600 DPI)



229x203mm (600 x 600 DPI)



63x44mm (600 x 600 DPI)



63x44mm (600 x 600 DPI)