A Two-Level Parallelization Strategy for Genetic Algorithms Applied to Shape Optimum Design

N. Marco, Stephane Lanteri

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Two-Level Parallelization Strategy for Genetic Algorithms Applied to Shape Optimum Design

N. Marco and S. Lanteri

N° 3463

Juillet 1998

THÈME 4

ISSN 0249-6399

INRIA
SOPHIA ANTIPOLIS

A Two-Level Parallelization Strategy for Genetic Algorithms Applied to Shape Optimum Design

N. Marco* and S. Lanteri*

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet Sinus

Rapport de recherche n° 3463 — Juillet 1998 — 30 pages

Abstract: This report presents a two-level strategy for the parallelization of a genetic algorithm coupled to a compressible flow solver designed on unstructured triangular meshes. The parallel implementation is based on MPI and makes use of the process group features of this environment. The resulting algorithm is used for the optimum shape design of aerodynamic configurations. Numerical and performance results are presented for the optimization of two-dimensional airfoils for calulations performed on the following systems : a `SGI Origin 2000` and a `IBM SP-2` MIMD systems; a `Pentium Pro` (`P6/200 Mhz`) cluster where the interconnection is realized through a `FastEthernet` (100 Mbits/s) switch.

Key-words: Optimization - Genetic Algorithms - Euler Equations - Shape parametrization - MPI - Parallelization strategy

* INRIA, 2004 Route des Lucioles, BP. 93, 06902 Sophia Antipolis Cédex-France

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : 04 92 38 77 77 - International : +33 4 92 38 77 77 — Fax : 04 92 38 77 65 - International : +33 4 92 38 77 65

Une stratégie de parallélisation à deux niveaux pour des Algorithmes Génétiques appliqués à l'optimisation de forme en aérodynamique

Résumé :

Ce rapport présente une stratégie de parallélisation à deux niveaux d'un algorithme génétique appliqué à l'optimisation de forme en aérodynamique. L'écoulement compressible régi par les équations d'Euler est résolu surdes maillages triangulaires non structurés. L'implémentation en parallèle utilise la librairie MPI et utilise la notion de groupes de processus. Des résultats de performance sont présentés pour l'optimisation de profils d'ailes 2D sur les architectures parallèles suivantes : une `SGI Origin 2000`, un `IBM SP-2` et un cluster de `Pentium Pro` (`P6/200 Mhz`) interconnectés par un switch `FastEthernet` (100 Mbits/s).

Mots-clés : Optimisation - Algorithme génétique - Equations d'Euler - Paramétrisation de forme - MPI - Stratégie de parallélisation

1 Introduction

Genetic Algorithms (GAs) are search algorithms based on the mechanisms of natural selection. They lay on one of the most important principles of Darwin : *survival of the fittest*. John Holland, in the 1970's, thought that he could incorporate in a computer algorithm such a technique, to solve difficult problems through evolution. This technique, close to the laws of nature, uses a population of potential solutions represented by strings of binary digits (called *chromosomes* or *individuals*) which is submitted to many transformations (called genetic operations such that *selection, crossover* and *mutation*). The population is going to evolve during the generations according to the fitness value of the individuals; then, when a stationary state is reached, the population has converged to the solution of the given optimization problem. More recently, David Goldberg brought GAs in non convex optimization theory and introduced a decisive thrust in the GAs research field (see [5]).

GAs are different from normal optimization procedures (e.g. simple or conjugate gradient methods or steepest descent methods) in many ways :

- they work with a coding of the parameter set and not the parameters themselves,
- they work simultaneously with a population of potential binary coded solutions, not only with one solution,
- they use probabilistic rules (the genetic operators are applied with probabilities) and not deterministic ones,
- they investigate in a search space componing a database of the solutions, which implies that they cannot fall into a local optimum,
- two keywords are linked to GAs : *exploration* and *exploitation*. Exploration of the search space is important at the beginning of the GA process while exploitation is desirable when the GA process is close to the global optimum.

From about fifteen years, GAs have been introduced in aerodynamics shape design problems (see Kuiper *et al.* [8], Périaux *et al.* in [13], Quagliarella in [16] and Obayashi in [11] who presents 3D results for a transonic flow around a wing geometry). They are computationaly simple and they accomodate well to discontinuous or non linear environments. They are able to optimize problems involving simultaneously real, integer and boolean parameters. They count on both the exploration of the search space and the exploitation of the results. The exploration, which relies on a pseudo-random search, is wider than what is obtained with classical deterministic methods (such as

steepest descent methods). The latter methods are particularly well adapted to local (unimodal) optimization problems whereas GAs are much more robust for non-convex situations. Another attractive feature is that GAs can approach a multi-objective optimization problem. In that case, several criteria are taken into account simultaneously meaning that there exists a set of optimal solutions instead of a unique one (for more details, see Horn and Nafpliotis in [7], Goldberg and Richardson in [6] or Poloni in [15]). As a result, a data base of optimal solutions is created during the optimization process. Then, by choosing weights for each criteria, the different decision makers will obtain an optimal solution for their own problem.

The main concern related to the use of GAs for aerodynmic design is the computational effort needed for the accurate evaluation of a design configuration that, in the case of a crude application of the technique, might lead to unacceptable computer time if compared with more classical algorithms. In addition, hard problems need bigger populations and this translates directly into higher computational costs. It is a widely accepted position that GAs can be effectively parallelized and can in principal take full advantage of (massively) parallel computer architectures. This point of view is before all motivated by the fact that within a generation (iteration) of the algorithm, the fitness values associated to each individual of the population can be evaluated in parallel.

The present paper aims at presenting a parallelization strategy of GAs which is particularly well suited to optimization problems based on direct state calculations characterized by high CPU costs and large memory requirements. In the context of GAs, these characteristics translate into constraints on the evaluation of the fitness values associated to each individual of a population. In particluar, for the applications considered here involving an unstructured grid based CFD solver, we have observed that 80 to 90 % of the total CPU time is spent in the evaluation of fitness values. The basic motivation behind many early studies of Parallel Genetic Algorithms (PGAs) was to reduce the processing needed to reach an acceptable solution; later on, coarse grain PGAs based sub-populations models and migration operators were introduced and were shown to bring improvements on the convergence rate of classical GAs (see for example [10]). The present work lies in between the direct parallel approach which consists in concurrently evaluating the fitness values of each individual of a population within a given generation and, the more sophisticated sub-populations based PGAs will be motivated in the sequel. The authors refer to Cantù-Paz[1] for a recent review of PGAs.

In the next section we describe the genetic operators that form the building blocks of our GA. As this GA is used for the optimum design of aerodynamic shapes, we detail in section 3 the shape parametrization procedure. In section 4, we present the features of the underlying compressible flow solver. The flow solver is based on a mixed finite volume/finite element MUSCL method designed on unstructured triangular meshes. In section 5, we motivate and introduce a two-level strategy for the parallelization of a genetic algorithm coupled to a compressible flow solver. Finally, in section 6, numerical and performance results are presented for the optimization of 2D airfoils. Calculations have been performed on the following systems : a `SGI Origin 2000` and a `IBM SP-2` MIMD systems; a `Pentium Pro` (`P6/200 Mhz`) cluster where the interconnection is realized through `FastEthernet` (100 Mbits/s) switches.

2 The Genetic Algorithm

The basic structure of a Genetic Algorithm consists of the following steps : **(1)** initialize randomly a population of individuals, **(2)** evaluate the individuals following their fitness (cost functional) value, **(3)** apply genetic operators (crossover and mutation) to the population and goto **(2)** until the best individual is reached. The following subsections describe the genetic operators taking part in the binary coded GA used in our study (the individuals encoding is described is section 3).

2.1 Selection process (or reproduction)

The selection process is based on a tournament approach. Globally, the tournament consists of picking randomly two individuals (for a tournament of size 2 or "binary tournament") from the population and selecting the best according to its fitness. This best individual is retained for the new population. The two individuals are then put back in the population and the process is restarted unitil a new population is completed. For more details, see Oei *et al.*[12].

2.2 Crossover operator

As reproduction does not create new individuals, the crossover operator is needed to increase diversity among the population. Crossover proceeds in two steps. First, strings of the newly reproduced population are mated at random ; this crossover operation is made with a probability p_c. Second, each pair of strings undergoes crossing-over as follows : an integer position k along the string is selected uniformly at random in $[1, l-1]$, where l is the string length of the chromosomes and two new strings are

created by swapping all characters between positions $k+1$ and l inclusively (see in Fig. 1). These two new strings replace their parents in the population.

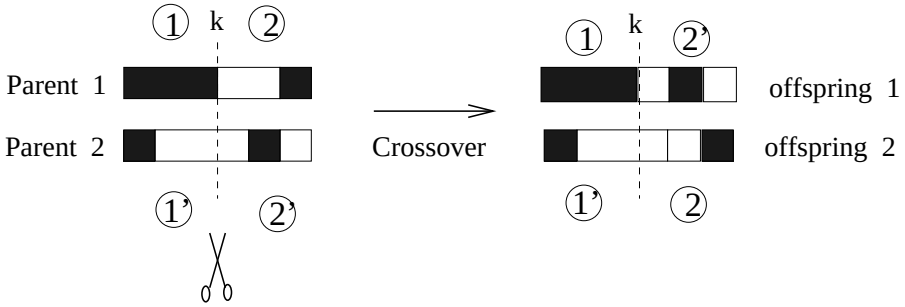

Figure 1: An example of crossover between two parents

2.3 Mutation operator

The mutation is needed because reproduction and crossover can occasionally loose some potentially useful genetic material (1's or 0's at particular locations). Mutation is then an insurance policy against premature loss of important notions. In a binary encoding context, this simply means changing a 1 into a 0 and vice versa, randomly, with a small probability p_m (see Fig. 2).

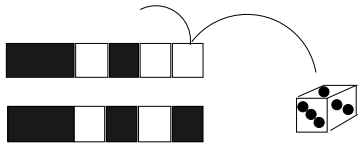

Figure 2: A schematic of simple mutation

3 Shape parametrization

In the 2D shape optimum design problems considered here, the population of individuals is represented by airfoil shapes. A natural parametrization of airfoils is a point by point one, based on the discretization of the shape. Such an approach presents two drawbacks :

- in order to obtain accurate results, the shape has to be defined by a lot of points and it is well known that the convergence of GAs depends on the number of

parameters (see [5]). It has been established (see [18]) that the time complexity of serial GAs is of $O(N \times l)$ or $O(N \log N + N \times l)$ depending on the selection scheme used, where N is the population size and l denotes the string length which is deduced from the number of parameters;

- the point to point representation is not fitted to the crossover operator. Indeed, the crossing-over of two individuals may create two new individuals that are non aerodynamically feasible (see Fig. 3).

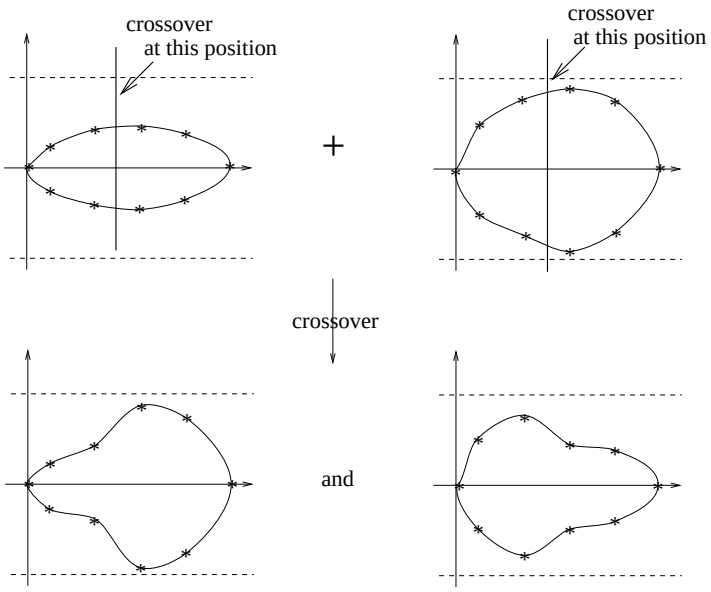

Figure 3: Crossing-over of two different individuals with a point-to-point representation

So, following a strategy already adopted by several authors[13]-[15]-[16], the shape parametrization procedure is based on Bézier curves. A few control points are then enough to represent the whole shape and, on the other hand, the smoothness properties of Bézier curves never imply the creation of non feasible shapes by the crossover operator. Bézier curves have a useful convex hull property that restricts the curve to never leave the bounding polygon of the control points. The convex hull property is derived from the fact that a Bézier curve is a convex combination of the data points (see Fig. 4).

A Bézier curve of order n is defined by the *Bernstein polynoms* $B_{n,j}$:

$$B(t) = \sum_{i=0}^{n} B_{n,i} P_i \quad \text{with} \quad B_{n,i} = C_n^i t^i (1-t)^{n-i} \quad , \quad C_n^i = \frac{n!}{(n-i)!}$$

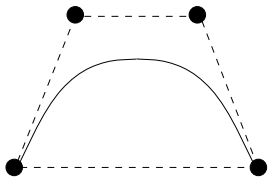

Figure 4: Creation of a Bézier curve from four control points

where $t \in [0,1]$, $P_i = (x_i, y_i)$ are the coordinates of the control points. In this work, a 8 order Bézier representation has been used (with 2 fixed points, P_0 and P_8, and 7 control points) :

$$x(t) = \sum_{i=0}^{8} C_8^i t^i (1-t)^{8-i}\ x_i \quad , \quad y(t) = \sum_{i=0}^{8} C_8^i t^i (1-t)^{8-i}\ y_i \tag{1}$$

Two Bézier curves are defined respectively for the extrados and the intrados; the points $P_0 = (0,0)$ and $P_8 = (1,0)$ are fixed because they correspond to the leading and trailing edges of the airfoil. The values $x_i \in [0,1]$ are fixed and the only parameters that vary are the ordinates $y_1, \cdots, y_7$ and $y_9, \cdots, y_{15}$. A chromosome is defined by a vector of $\mathbb{R}^{14}$ (see Fig. 5); the genes are the ordinates of the control points) :

$$chromosome = (\underbrace{y_1, \cdots, y_7}_{extrados}, \underbrace{y_8, \cdots, y_{14}}_{intrados})$$

In addition, we need to use a geometrical constraint : the y_i's vary on intervals $[min_i, max_i]$; this has a direct implication on the definition of the search space. The parameters constituting the chromosomes are then binary encoded.

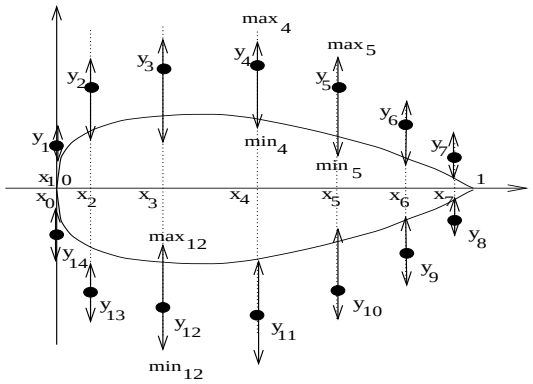

Figure 5: Representation of an airfoil using Bezier curves

Suppose that we choose an accuracy of 10^{-p} for each parameter y_i. Then, the interval $[min_i, max_i]$ is going to be discretized in $(max_i - min_i).10^p$ equally spaced intervals. Let m_i be the smallest integer such that :

$$(max_i - min_i).10^p \leq 2^{m_i} - 1$$

then m_i is the length of the binary string coding the parameter y_i. The real value of y_i is computed using the following expression :

$$y_i = min_i + (\text{decimal value})_i . \left(\frac{max_i - min_i}{2^{m_i} - 1} \right)$$

where :

$$(\text{decimal value})_i = \sum_{j=0}^{m_i - 1} 2^j \alpha_j$$

α_j being the bit values of the binary string coding y_i. The chromosome will then be represented by a string of length $l = \sum_{i=1}^{n} m_i$ where n denotes the number of parameters.

4 The flow solver

In the present study, the evaluation of a fitness value to be associated to an individual requires the calculation of the steady non-viscous compressible flow around an airfoil. The objective of this section is to describe briefly the main ingredients of the underlying flow solver for the solution of the two-dimensional Euler equations.

4.1 Mathematical model and approximation methods

The Euler equations describing 2D flows are given in conservative form by :

$$\frac{\partial W}{\partial t} + \vec{\nabla} . \vec{\mathbf{F}}(W) = 0 \quad , \quad W = \left(\rho \ , \ \rho \vec{U} \ , \ E \right)^T \quad , \quad \vec{\nabla} = \left(\frac{\partial}{\partial x} \ , \ \frac{\partial}{\partial y} \right)^T \tag{2}$$

where $\vec{\mathbf{F}}(W) = (F_1(W), F_2(W))^T$ is the vector of convective fluxes whose components are given by :

$$F_1(W) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{pmatrix} \quad , \quad F_2(W) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{pmatrix}$$

In the above expressions, ρ is the density, $\vec{U} = (u, v)^T$ is the velocity vector, E is the total energy per unit of volume and p denotes the pressure which is obtained using the perfect gas state equation $p = (\gamma_p - 1)(E - \frac{1}{2}\rho \parallel \vec{U} \parallel^2)$ where $\gamma_p = 1.4$ is the ratio of specific heats. The flow domain Ω is assumed to be a polygonal bounded region of $\mathbb{R}^2$. Let τ_h be a standard triangulation of Ω. A vertex of the mesh is denoted by S_i, and the set of its neighboring vertices by $N(S_i)$. At each vertex S_i, a control volume C_i is constructed by joining the middle of the edges $\{S_i, S_j\}$ for $S_j \in N(S_i)$ with the centroids of the triangles sharing S_i; this is depicted on Fig. 6. The boundary of C_i is denoted by ∂C_i, and the unit vector of the outward normal to ∂C_i by $\vec{\nu}_i$. The union of all these control volumes constitutes a dual discretization of Ω.

The spatial discretization method adopted here uses a finite volume upwind formulation. Briefly speaking, for each control volume C_i associated to a vertex S_i, one has to solve for n_i *one-dimensional Riemann problems* at the control volume boundary, n_i being the number of neighbors of S_i. The spatial accuracy of the Riemann solver depends on the accuracy of the interpolation of the physical quantities at the control volume boundary. For first order accuracy, the values of the physical quantities at the control volume boundary are taken equal to the values in the control volume itself. Extension to second order accuracy can be performed via a "Monotonic Upwind Scheme for Conservative Laws" (MUSCL) technique of Van Leer[19] . It consists in giving the Riemann solver a better interpolated value, taking into account some gradient of the physical quantities. One can use a *finite element gradient* (P_1-Galerkin) computed on a particular triangle, or an *averaged nodal gradient*, which is taken as a particular average of the finite element gradients on the set of triangles sharing a given vertex.

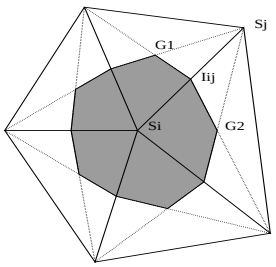

Figure 6: A 2D control volume C_i

Integrating Eq. (2) over C_i yields :

$$\begin{aligned} \iint_{C_i} \frac{\partial W}{\partial t}\vec{x} \;+\; \sum_{j\in N(S_i)} \int_{\partial C_{ij}} \vec{\mathbf{F}}(W).\vec{\nu_i}d\sigma \qquad & < 1 > \\ + \int_{\partial C_i \cap \Gamma_w} \vec{\mathbf{F}}(W).\vec{\nu_i}d\sigma \;+\; \int_{\partial C_i \cap \Gamma_\infty} \vec{\mathbf{F}}(W).\vec{\nu_i}d\sigma = 0 \quad & < 2 > \end{aligned} \tag{3}$$

where $\partial C_{ij} = \partial C_i \cap \partial C_j$; Γ_w and Γ_∞ respectively denote the wall boundary and the downstream/upstream boundary. A first order finite volume discretisation of $< 1 >$ is :

$$< 1 > = \; W_i^{n+1} - W_i^n + \Delta t \sum_{j\in N(S_i)} \Phi_{\mathbf{F}}(W_i^n, W_j^n, \vec{\nu_{ij}}) \tag{4}$$

where $\Phi_{\mathbf{F}}$ denotes a numerical flux function such that :

$$\Phi_{\mathbf{F}}(W_i, W_j, \vec{\nu_{ij}}) \approx \int_{\partial C_{ij}} \vec{\mathbf{F}}(W).\vec{\nu_i}d\sigma \quad , \quad \vec{\nu_{ij}} = \int_{\partial C_{ij}} \vec{\nu_i}d\sigma \tag{5}$$

Upwinding is introduced here in the computation of Eq. (5) by using Roe's[17] approximate Riemann solver thus computing $\Phi_{\mathbf{F}}$ as follows:

$$\Phi_{\mathbf{F}}(W_i, W_j, \vec{\nu_{ij}}) = \frac{\vec{\mathbf{F}}(W_i) + \vec{\mathbf{F}}(W_j)}{2}.\vec{\nu_{ij}} - \mid \mathbf{A}_R(W_i, W_j, \vec{\nu_{ij}}) \mid \frac{(W_j - W_i)}{2} \tag{6}$$

where $\mathbf{A}_R$ is Roe's mean value of the flux Jacobian matrix $\frac{\partial \vec{\mathbf{F}}(W)}{\partial W}.\vec{\nu}$. Following the MUSCL technique[19] , second order accuracy is achieved in Eq. (5) via a piecewise linear interpolation of the states W_{ij} and W_{ji} at the interface ∂C_{ij} (see Fig. 7) :

$$\tilde{W}_{ij} = \tilde{W}_i + \frac{1}{2}(\vec{\nabla}\tilde{W})_i.\vec{S_iS_j} \quad , \quad \tilde{W}_{ji} = \tilde{W}_j - \frac{1}{2}(\vec{\nabla}\tilde{W})_j.\vec{S_iS_j} \tag{7}$$

where $\tilde{W} = \left(\rho \;,\; \vec{U} \;,\; p\right)^T$. An *averaged nodal gradient* $(\vec{\nabla}\tilde{W})_i$ is obtained by averaging the P_1-Galerkin gradients computed on each triangle of C_i. Finally, the Van Albada limitation procedure[4] is introduced in the interpolation (7) in order to preserve the monotony of the approximation .

4.2 Time integration

As stated earlier, we are interested here in steady state solutions of the Euler equations. Therefore, time integration must be as efficient as possible regardless of time accuracy. For this purpose, an implicit formulation that allows the use of large pseudo-time steps

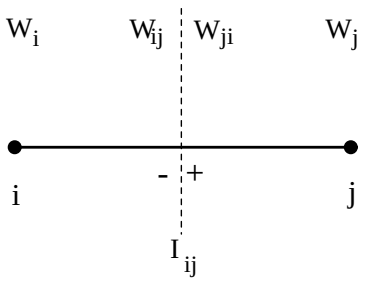

Figure 7: Interpolated physical states W_{ij} and W_{ji} on an edge $\{S_i, S_j\}$

is adopted. Assuming that $W(\vec{x}, t)$ is constant over C_i (in other words, a mass lumping technique is applied to the temporal term), we obtain :

$$\mathrm{vol}(C_i)\frac{dW_i}{dt} + \Psi(W)_i = 0 \quad , \quad i = 1, \cdots, N_v \tag{8}$$

where $W_i = W(\vec{x_i}, t)$, N_v is the number of control volumes, and :

$$\Psi(W)_i = \sum_{j \in N(S_i)} \Phi_{\mathbf{F}}(W_{ij}, W_{ji}, \vec{\nu_{ij}}) + \int_{\partial C_i \cap \Gamma} \vec{\mathbf{F}}(W).\vec{\nu_i} d\sigma \tag{9}$$

where $\Gamma = \Gamma_w \cup \Gamma_\infty$. Applying a first order linearization to the flux $\Psi(W^{n+1})$ yields the Newton-like formulation[4] :

$$\left(\frac{\mathrm{vol}(C_i)}{\Delta t^n} + J(W^n)\right)(W^{n+1} - W^n) = -\Psi(W^n) \tag{10}$$

where $J(W^n)$ denotes the associated Jacobian matrix. At each time step, the above linear system is approximately solved using Jacobi relaxations.

4.3 Mesh update procedure

Once a new shape has been determined using the procedure described in section 3, the overall computational mesh has to be updated prior to a new flow calculation. In this work, an unstructured dynamic fluid mesh is represented by a pseudo structural model where a fictitious linear spring is associated with each edge connecting two fluid mesh vertices S_i and S_j. The vertices located on the downstream and upstream boundaries are held fixed while the new positions of those points located on the wall boundary are determined from shape parametrization procedure. Then, the new position of the interior vertices is determined from the solution of a displacement driven pseudo structural problem via a Jacobi iterative procedure on appropriate static equilibrium

equations. This procedure is adapted from the work presented in Farhat and Lanteri[3] for unsteady flow simulation on dynamic meshes.

5 Parallelization strategy

Several possible strategies can be considered for the parallelization of the GA based shape optimum design methodology described in the previous sections :

- a first strategy stems from the following remark : within a given generation of the genetic algorithm, the evaluation of the fitness values associated to the population individuals defines independent processes. This makes GAs particularly well suited to massively parallel systems; we also note that a master/slave approach is a standard candidate for the implementation of this first level of parallelism, especially when the size of the populations is greater than the available number of processors;

- a second strategy consists in concentrating the parallellization efforts on the process underlying a fitness value evaluation i.e. on the flow solver. This approach finds its main motivation in the fact that, when complex field analysers are used in conjonction with a genetic algorithm, then the aggregate cost of fitness values evaluations can represent between 80 to 90% of the total optimization time. A SPMD paradigm is particularly well suited to the implementation of this strategy;

- the third option combines the two above approaches and clearly yields a two-level parallelization strategy which has been considered here and which is detailed in the sequel. Our choice has been motivated by the following remarks : (1) a parallel version of the two-dimensional flow solver was available and adapted to the present study, (2) we have targetted a distributed memory SPMD implementation and we did not want the resulting optimization tool to be limited by memory capacity constraints, especially because the present study will find its sequel in its adaptation to 3D shape optimization problems, based on more complex aerodynamic models (Navier-Stokes equations coupled to a turbulence model) and, (3) we believe that the adopted parallelization strategy will define a good starting-point for the construction and evaluation of sub-populations based PGAs.

5.1 Parallelization of the flow solver

The parallelization strategy adopted for the flow solver combines domain partitioning techniques and a message-passing programming model. The underlying mesh is as-

sumed to be partitioned into several submeshes, each defining a subdomain. Basically the same "old" serial code is going to be executed within every subdomain. Applying this parallelization strategy to the previously described Euler flow solver results in modifications occuring in the main time-stepping loop in order to take into account one or several assembly phases of the subdomain results, depending on the order of the spatial approximation and on the nature of the time advancing procedure (explicit/implicit).

This approach enforces data locality, and therefore is suitable for all parallel hardware architectures. For the partitioning of the unstructured mesh, two basic strategies can be considered. The first one is based on the introduction of an overlapping region at subdomain interfaces and is well suited to the mixed finite volume/element formulation considered herein. Mesh partitions with overlapping have a main drawback : they incur redundant floating-point operations. The second possible strategy is based on non-overlapping mesh partitions and incur no more redundant floating-point operations. While updated nodal values are exchanged between the subdomains in overlapping mesh partitions, partially gathered quantities are exchanged between subdomains in non-overlapping ones. It has been our experience that both the programming effort and the performances are maximized when considering non-overlapping mesh partitions[9]. In the present study we will consider one triangle wide overlapping mesh partitions for second order accurate implicit computations.

5.2 Overall optimization loop

The coordination of subdomain calculations through information exchange at artificial boundaries is implemented using calls to functions of the MPI library. Therefore, the paralellization described above aims at reducing the cost of the fitness function evaluation for a given individual. However, another level of parallelism can clearly be exploited here and is directly related to the binary tournament approach and the crossover operator. In practice, during each generation, individuals of the current population are treated pairwise; this applies to the selection, crossover, mutation and fitness function evaluation steps. Here, the main remark is that for this last step, the evaluation of the fitness functions associated to the two selected individuals, defines independent operations. We have chosen to exploit this fact using the notion of process groups which is one of the main features of the MPI environment. Two groups are defined, each of them containing the same number of processes; this number is given by the number of subdomains in the partitioned mesh. Now, each group is responsible for the evaluation of the fitness function for a given individual. We note in passing that such an approach based on process groups will be also interesting in the context of sub-populations based PGAs; this will be considered in a future work.

5.3 Parallel algorithms

The general structure of the parallelized genetic algorithm considered in this study is depicted on Fig. 8 while Fig. 9 visualizes the evaluation step using two process groups.

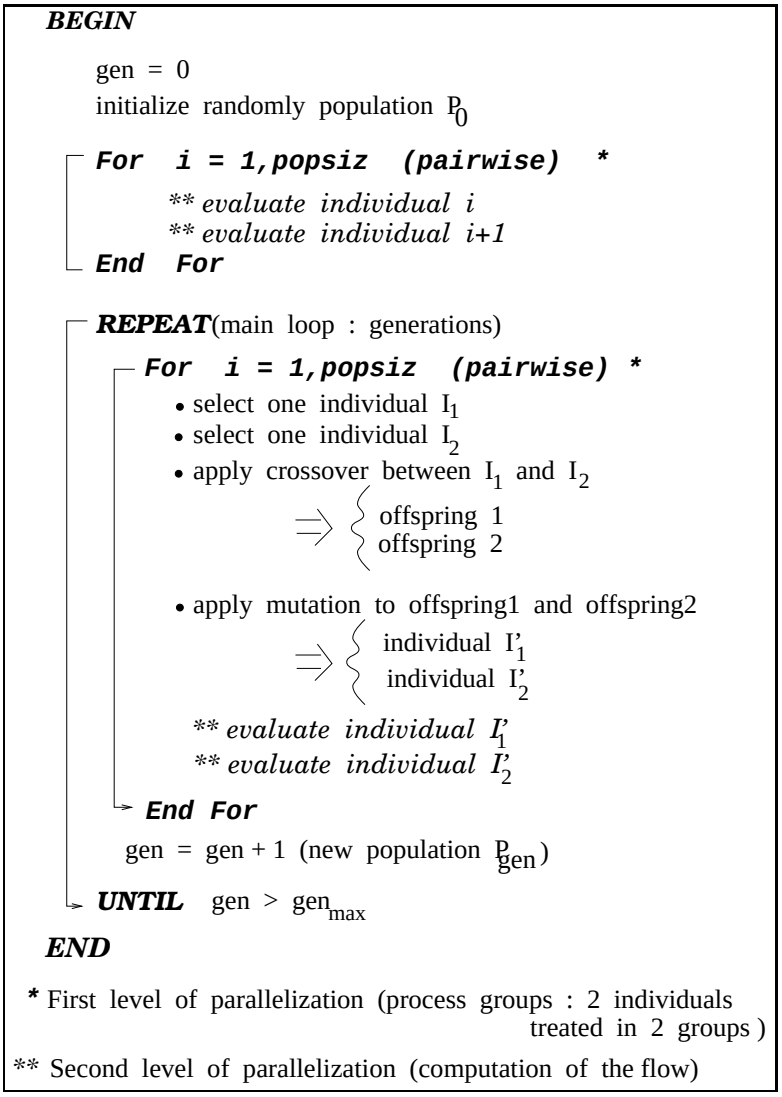

Figure 8: General structure of the parallelized GA

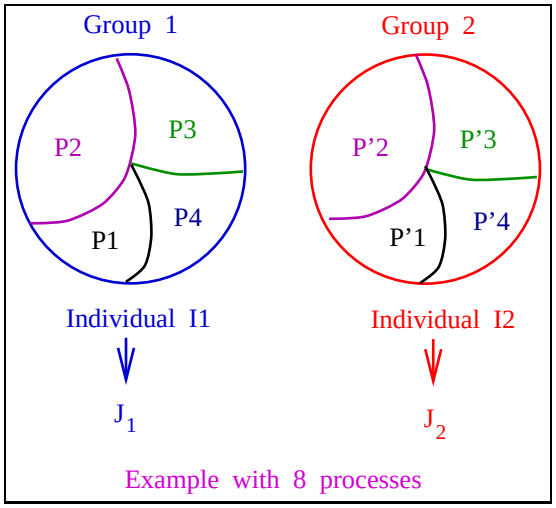

Figure 9: The process group approach for the evaluation of two individuals

6 Numerical and performance results

6.1 A shape optimum design application

This section is mainly devoted to a detailed evaluation of the performances of the proposed parallelization strategy for GAs. For this reason, we have considered a single test case which is a direct optimization problem; the goal is to obtain the shape of a `RAE2822` type airfoil corresponding to the minimization of the shock-induced drag. In order to do so, we consider the external Euler flow around a `RAE2822` airfoil (2° incidence, freestream Mach number fixed to 0.73). The non-linear convergence tolerance has been fixed to 10^{-6}. The cost functional is defined by :

$$j(\gamma) = C_D + 10(C_L - C_L^{comput})^2$$

where γ represents the airfoil shape, C_D is the drag coefficient, C_L is the lift coefficient, C_L^{comput} is the computed lift coefficient on the `RAE2822`. The aim is to reduce C_D while preserving C_L^{comput}. The computational mesh consists of 14747 vertices (160 vertices on the airfoil) and 29054 triangles (see Fig. 10). Here, we do not enter into the details of the GA parameters as the objective is not to assess the complete optimization process for various configurations. In addition to the geometrical constraint stated in section 3, we have also taken into account a constraint about the thickness of the airfoil : the thickness of the `RAE2822` airfoil is preserved with a 5% maximum variation. We will see in the following that the latter constraint poses a challenging problem for the parallelization. The angle of attack is also taken as a parameter to be optimized,

function of the computed thickness and the constraint on the lift coefficient. After 50 generations, the shape has been modified and the shock has been notably reduced (see Fig. 11). The initial and final flows (iso-Mach values) are presented on Figure 13. Finally, the initial and final values of C_D and C_L are :

$C_L^{comput} = 0.8068$	$C_L^{opt} = 0.8062$
$C_D^{init} = 0.0089$	$C_D^{opt} = 0.0048$

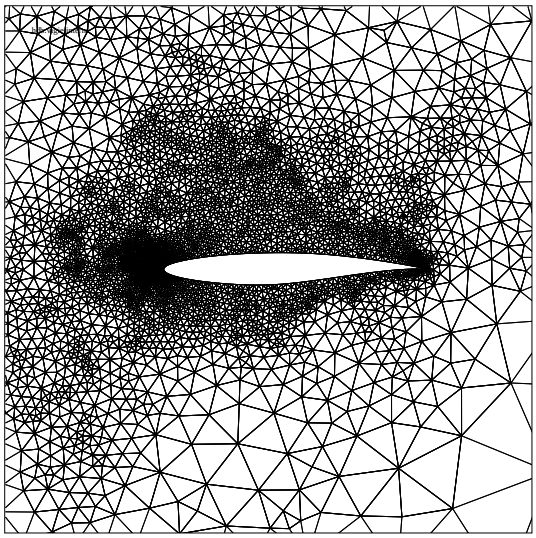

Figure 10: Computational mesh for a `RAE2822` airfoil

6.2 Parallel performance results

6.2.1 General remarks

Calculations have been performed on the following systems : a `SGI Origin 2000` (equipped with `Mips R10000/195 Mhz` processors), a `IBM SP-2` (equipped with `Risc P2SC/120 Mhz` processors) MIMD systems and an experimental `Pentium Pro` (`P6/200 Mhz`, running the LINUX system) cluster where the interconnection is realized through `FastEthernet` (100 Mbits/s) switches. The native MPI implementations have been used on the `SGI Origin 2000` and `IBM SP-2` systems while MPICH 1.1 has been used on the `Pentium Pro` cluster. Performance results are given for 64 bit arithmetic computations. We compare timing measures for the overall optimization using one and two process groups. Timings are given for a fixed number of generations (generally, 5

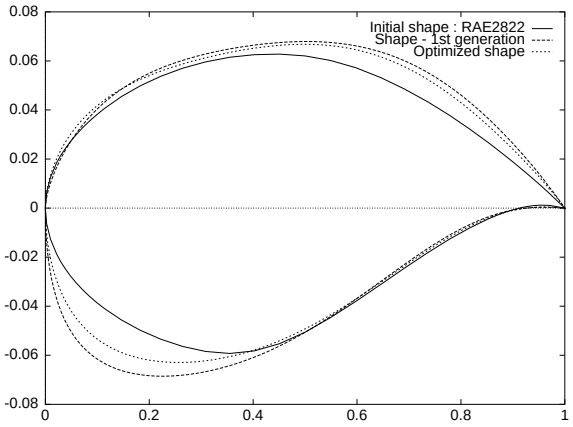

Figure 11: Drag reduction : initial, first generation and optimized airfoils (up)

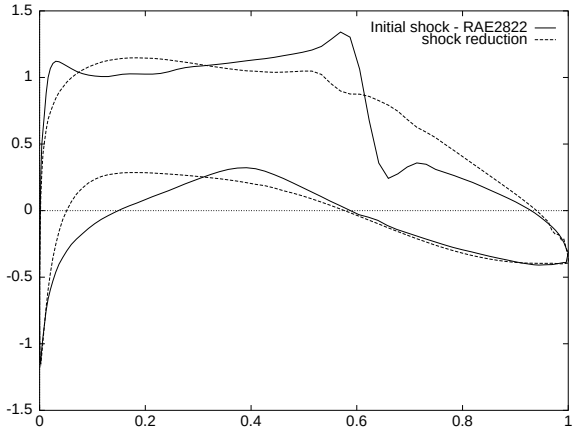

Figure 12: Drag reduction : initial and optimized pressure coefficient

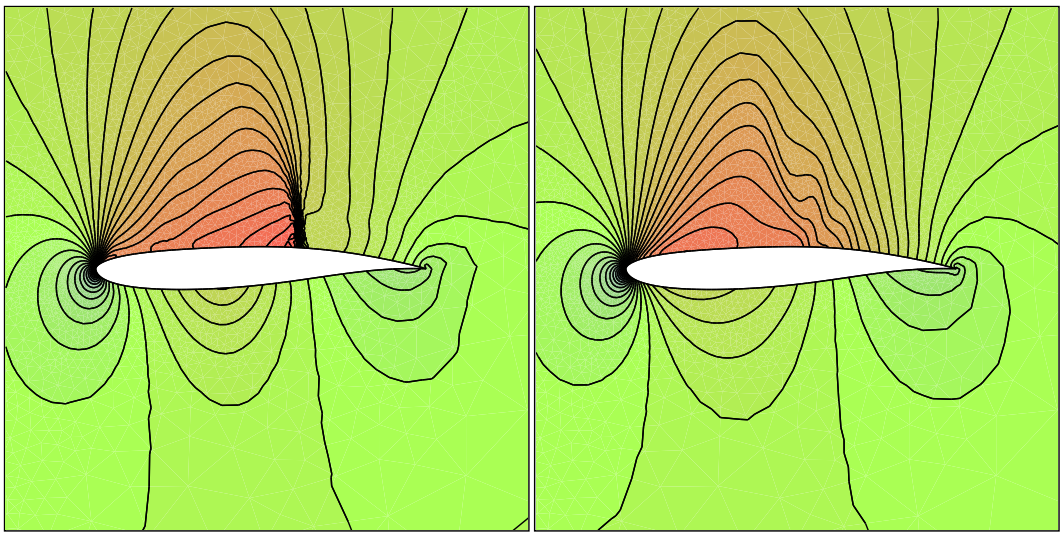

Figure 13: Drag reduction : initial and optimized flows (steady iso-Mach lines)

optimization iterations). In the tables below, N_g and N_p respectively denote the number of process groups and the total number of processes ($N_g = 2$ and $N_p = 4$ means 2 processes for each of the two groups), "CPU" is the total CPU time, "Flow" is the accumulated flow solver time. "Elapsed" is the total elapsed time (the distinction between the CPU and the elapsed times is particularly relevant for the `Pentium Pro` cluster). On the other hand, $S(N_p)$ is the parallel speed-up. For the multiple process cases, the given timing measures ("CPU" and "Flow") always correspond to the maximum value over the per-process measures.

6.2.2 Numerical experiment parameters

A typical numerical experiment on the optimization problem stated in paragraph 6.1 requires the setting of a number of execution parameters concerning the Genetic Algorithm and the compressible flow solver. Some of them have been fixed for all the numerical experiments that have been considered here; others characterize the differences between the various numerical experiments and have been selected with regard to their influence on the parallel performances of the overall optimization process. As stated above, the geometrical constraint on the airfoil thickness poses an interesting problem from the parallelization point of view. In practice, when two individuals are selected, mated and mutated, valid values of the fitness function for the resulting offsprings can be obtained only if the corresponding airfoils do verify the constraint on the thickness. In particular, when a given airfoil does not verify the latter condition,

an artificially high value is attributed to the fitness function in order to reduce the chances to select the corresponding individual in the next generation. This verification step is performed for each offspring prior to their evaluation such that flow calculations are actually performed for valid airfoils. Therefore, when two process groups are used and an odd number of valid airfoils has to be evaluated, a situation of computational load imbalance can be obtained; here we have chosen to force the corresponding flow calculation to be performed by each group. However, there are other situations that result in computational load imbalance. A typical example is concerned with the flow solutions associated with the acceptable individuals (i.e. verifying the geometrical constraint on the airfoil thickness) within a given optimization iteration. The non-linear convergence history is of course strongly dependent on the shape of an airfoil that is, different airfoil shapes result in different number of iterations (pseudo-time steps) for obtaining the steady-state. For some airfoils, the maximum allowable number of iterations is reached without a reasonable level of convergence; in such cases, we again set an artificially high value for the corresponding fitness function.

For the numerical experiments considered below, the following parameters have been fixed :

- *for the GA* : the size of the population has been set to 30; the crossover operator probability $p_c = 0.6$; the mutation operator probability $p_m = 0.064$ (this probability is reduced by a factor 2 every ten generations).

- *for the flow solver* : the tolerance on the non-linear residual for the obtention of the steady state has been set to 10^{-4} (this value is relative to the initial normalized residual; several values for the maximum number of time steps are considered in the following numerical experiments); in the implicit time integration scheme, the local pseudo time steps are deduced from the law $CFL = \max(10 \times it, 10^6)$ where it is the non-linear iteration number; for the linear system solution phases, the residual tolerance is 10^{-3} with a maximum of 100 Jacobi relaxations; finally, for the mesh deformation procedure, the tolerance is 10^{-4} with a maximum of 300 Jacobi relaxations.

6.2.3 Parallel performances

We have chosen to assess the performances of the proposed parallelization strategy for GAs by considering several situations : one versus two process groups and various numbers of submeshes within one process group; moreover, the influence of a computational load imbalance on the parallel efficiencies has been evaluated by considering several values for the maximum number of time steps for a flow calculation. For all the numerical experiments, timings are given for 5 optimization iterations.

Table 1: Computational load per generation in terms of flow calculations

#it	Init	Gen 1	Gen 2	Gen 3	Gen 4	Gen 5
50	4	7	10	10	11	12
40	4	5	7	8	11	13
35	4	4	7	7	5	8

Table 2: Parallel perfomance results on the `SGI Origin 2000`

N_g	N_p	*#it*	Elapsed	CPU	Flow		$S(N_p)$
					Min	Max	
1	5	50	2187 sec	2173 sec	1934 sec	1995 sec	1.0
2	10	50	1270 sec	1261 sec	1031 sec	1118 sec	1.7
1	10	50	1126 sec	1115 sec	900 sec	953 sec	1.9
1	5	40	1962 sec	1948 sec	1735 sec	1783 sec	1.0
2	10	40	1145 sec	1135 sec	963 sec	1006 sec	1.7
1	10	40	988 sec	978 sec	798 sec	841 sec	2.0
1	5	35	1671 sec	1640 sec	1306 sec	1345 sec	1.0
2	10	35	806 sec	797 sec	683 sec	713 sec	2.0
1	10	35	687 sec	678 sec	554 sec	582 sec	2.4

6.2.4 Discussion

The results presented in Tab. 1 to 4 are very instructive, especially with regards to the following points.

Parallel speed-up : on each computing platform, the parallel speed-up $S(N_p)$ has been evaluated using the total elapsed time. The measure obtained for five processing nodes and one process group was taken as the reference time (i.e. the parallel speed-up is relative to the case $N_g = 1$ and $N_p = 5$). In general, the achieved efficiencies are acceptable knowing that the global problem size (the size of the underlying mesh) is relatively small. In particular, the local problem size resulting from the 10 submeshes decomposition is responsible for the superlinear speed-up observed on the `SGI Origin`

Table 3: Parallel perfomance results on the `IBM SP-2`

N_g	N_p	$\#it$	Elapsed	CPU	Flow		$S(N_p)$
					Min	Max	
1	5	50	4186 sec	4094 sec	3566 sec	3659 sec	1.0
2	10	50	2414 sec	2344 sec	1892 sec	2026 sec	1.7
1	10	50	2539 sec	2466 sec	1941 sec	2080 sec	1.65
1	5	40	3713 sec	3625 sec	3144 sec	3237 sec	1.0
2	10	40	2171 sec	2126 sec	1746 sec	1842 sec	1.7
1	10	40	2243 sec	2178 sec	1710 sec	1838 sec	1.65
1	5	35	2594 sec	2526 sec	2188 sec	2238 sec	1.0
2	10	35	1531 sec	1505 sec	1244 sec	1304 sec	1.7
1	10	35	1560 sec	1515 sec	1190 sec	1275 sec	1.65

Table 4: Parallel perfomance results on the `Pentium Pro` cluster

N_g	N_p	$\#it$	Elapsed	CPU	Flow		$S(N_p)$
					Min	Max	
1	5	50	18099 sec	14974 sec	13022 sec	14387 sec	1.0
2	5	50	9539 sec	8945 sec	7291 sec	8744 sec	1.85
1	10	50	10764 sec	8866 sec	7000 sec	7947 sec	1.7
1	5	40	15757 sec	13835 sec	11426 sec	13378 sec	1.0
2	5	40	9228 sec	8118 sec	6377 sec	7891 sec	1.65
1	10	40	8570 sec	7357 sec	5366 sec	6607 sec	1.85
1	5	35	10386 sec	9555 sec	8199 sec	9206 sec	1.0
2	5	35	6492 sec	5499 sec	4078 sec	5223 sec	1.6
1	10	35	5909 sec	4927 sec	3750 sec	4544 sec	1.75

2000; on this platform the R10000 processor possesses a 4 Mb cache memory, a rather large value that contributes to the obtention of high computational rates in the cases ($N_g = 2$, $N_p = 10$) and ($N_g = 1$, $N_p = 10$).

Computational load imbalance : here, we are interested in assessing the influence of a computational load imbalance on the parallel efficiencies achieved with the two process groups strategy. First of all, we notice that the aggregate time spent in solving flow problems represents between 77% to 92% of the total elapsed time for all the tested configurations. When $N_g = 2$, two flow problems can be simultaneously computed within a given generation (see Tab. 1 for a statistic of the number of flow solutions in each case) and it is particularly important that approximately the same time is spent in both flow solutions. Fig. 14 depicts the effective number of time steps for each flow solution (i.e. for the five generations of the GA). Therefore, we have chosen to evaluate the ratio $R_{balc} = \dfrac{(\text{Flow}_{max} - \text{Flow}_{min})}{\text{Flow}_{min}}$ in the case $N_g = 2$ and $N_p = 10$ for each value of $\#it$ (i.e. the maximum number of time steps allowable in a flow solution). On the SGI Origin 2000 we see that this ratio decreases from 8.5% to 4.5% and the parallel speed-up increases from 1.7 to 2.0; on the IBM SP-2 the corresponding figures for the above ratio are 7.0% and 4.8% and the parallel speed-up does not change. These figures clearly confirm our expectations however the present assessment needs to be complemented by a detailed study of the influence of the quality of the flow solution (which depends on the level on reduction of the non-linear residual) on the convergence and the optimal solution given by the GA. A partial answer is given by Fig. 15 and 16 where we visualize the reduction of the average and minimum value of the fitness function versus the GA generation.

Finally, a totally opposite behaviour is observed on the Pentium Pro cluster : R_{balc} increases from 20% to 28% and the corresponding values of the parallel speed-up are 1.85 (case $\#it = 50$) and 1.6 (case $\#it = 35$).

Parallel performance of the Pentium Pro cluster. The objective of this paragraph is twofold : to assess the benefits of the cluster computing approach for the present optimization application and, to try to give an explanation to the degradation of the parallel speed-up with the two process groups strategy. A measure that is of particular importance here, is given by the ratio of the CPU time to the total elapsed time. In a dedicated mode (which is the case here), the difference between the total elapsed time and the CPU time is a good measure of the communication time (I/O operations are limited and we have verified that the requested memory is well below the available memory on each node). Looking at the entries of Tab. 4 corresponding to

$N_g = 2$ and $N_p = 10$, we see that this ratio is respectively equal to 93% ($\#it = 50$), 88% ($\#it = 40$) and 85% ($\#it = 35$) which demonstrates that parallel computing using a `Pentium Pro` cluster based on a high performance interconnection is a viable alternative to truly MIMD architectures. Furthermore, if we compare the measures obtained for $\#it = 35$, we deduce that the `SGI Origin 2000` system is $6492/806 \approx 8$ times faster (and that the `IBM SP-2` is $6492/1531 \approx 4$ times faster) than the `Pentium Pro` cluster on this particular case. These would clearly be lower with the most recent generation of `Pentium Pro` processor (`P6/333 Mhz` for instance) or with improved switching technologies (`Myrinet` for instance).

Finally, we can verify from the measures obtained on the `Pentium Pro` cluster, that the aggregate flow solution times represent 91% ($\#it = 50$), 85% ($\#it = 40$) and 80% ($\#it = 35$) of the corresponding total elapsed times. For comparison, on the `IBM SP-2`, this ratio is relatively stable when varying the maximum number of times steps for a steady flow solution (around 84%); the same can be said for the measures obtained on the `SGI Origin 2000` system. In our application, the ratio between communication operations to purely arithmetic ones is rather small, that is, once updated data are available at the external side of subdomain interfaces, a large number of arithmetic operations are performed in parallel for the evaluation of convective fluxes (see Eq. 9) and the assembly of Jacobian matrices (see Eq. 10); additional communication steps are induced by the iterative solution of the resulting linear system. The degradation of the parallel speed-up that is observed on the `Pentium Pro` cluster can be explained as follows : the reduction of the maximum number of time steps ($\#it$) induces both a decrease in the number of communication steps and in the number of arithmetic operations; the fact is that this coupled reduction is not in favour of the communication cost while the time spent in purely arithmetic operations is notably reduced. This behaviour suggests that, for the present application, the `SGI Origin 2000` and `IBM SP-2` MIMD systems are better equilibrated than the tested `Pentium Pro` cluster; in particular, we can expect improved parallel speed-ups on the latter platform with the use of more performant switching technologies.

7 Concluding remarks and future works

A two-level strategy has been proposed for the parallelization of genetic algorithms applied to aerodynamic shape optimization applications. At the lower level (a fitness function evaluation), an unstructured grid based flow solver is parallelized by combining mesh partitioning techniques and a message passing programming model. MPI has been adopted for the parallel implementation. At the higher level (the generation loop)

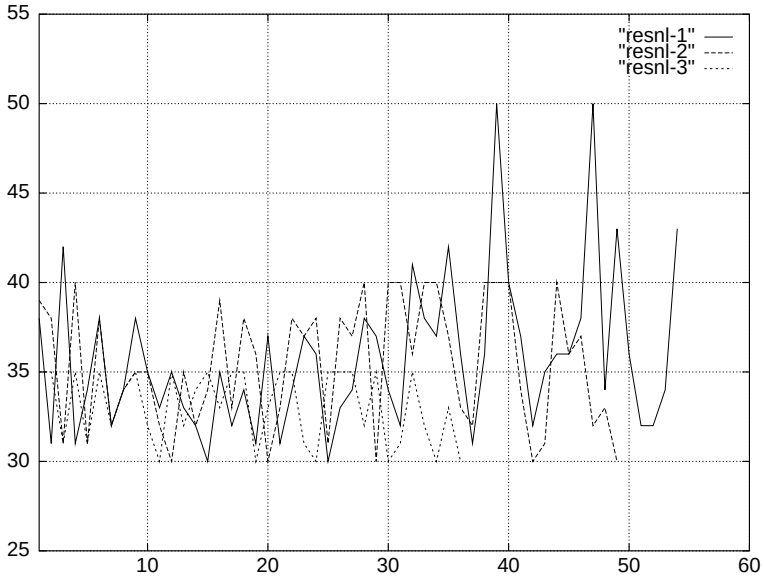

Figure 14: Effective number of non-linear iterations (time steps) per flow calculation
X-coordinate : index of flow calculation - Y-coordinate : number of time steps
resnl-1 : $\#it = 50$ - resnl-2 : $\#it = 40$ - resnl-3 : $\#it = 35$

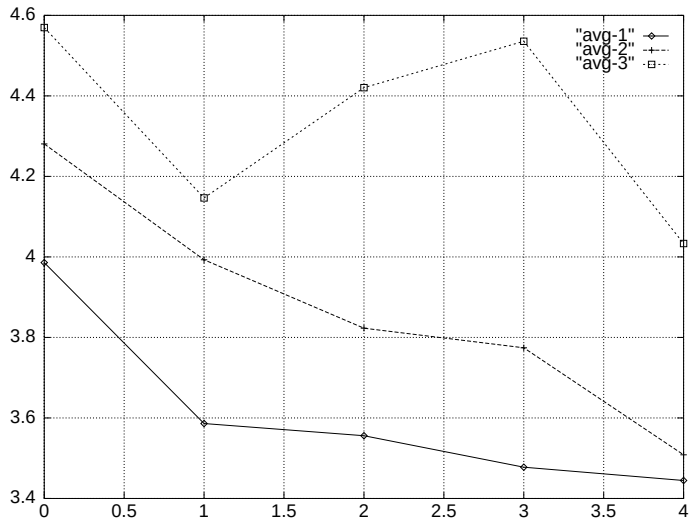

Figure 15: Evolution of the average value of the fitness function
X-coordinate : index of GA generation - Y-coordinate : average value of the fitness function
avg-1 : $\#it = 50$ - avg-2 : $\#it = 40$ - avg-3 : $\#it = 35$

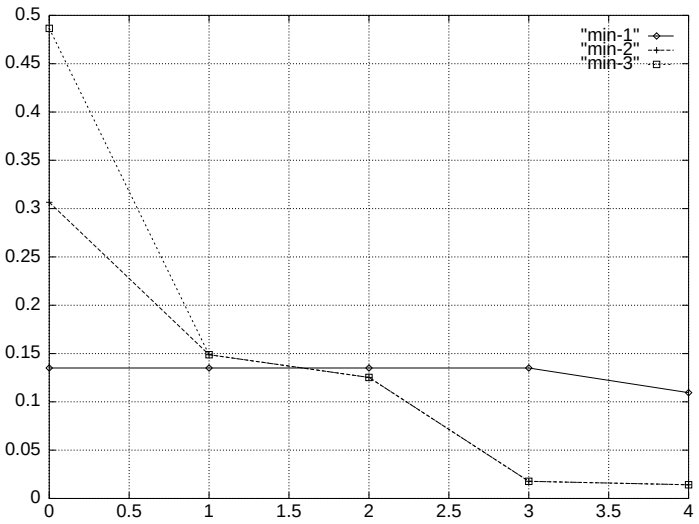

Figure 16: Evolution of the minimum value of the fitness function
X-coordinate : index of GA generation - Y-coordinate : minimum value of the fitness function
min-1 : $\#it = 50$ - min-2 : $\#it = 40$ - min-3 : $\#it = 35$

the MPI process group features has been used : two individuals are treated simultaneously using two MPI process groups; the same number of processes is attributed to each group for the calculation of the underlying flows.

The resulting optimization tool has been ported and evaluated on the following systems : a `SGI Origin 2000` and a `IBM SP-2` MIMD systems; a `Pentium Pro` (`P6/200 Mhz`) cluster where the interconnection is realized through `FastEthernet` (100 Mbits/s) switches.

An ongoing work is concerned with the extension of the present strategy to three-dimensional shape optimization (e.g. wings). In this study, one important step is the parametrization of the shape; the general goal is to work with chromosomes defined by short length strings (i.e. using a small number of parameters) and which result in smooth shapes. A natural choice which extends the parametrization procedure used in the present study, relies on Bezier surfaces[14].

Future research directions will target sub-populations based PGA's[10] and hybrid GA/local (gradient based) optimization techniques. These two topics will be considered by the authors in the context of the `ESPRIT DECISION` (*HPCN-Integrated Optimization Strategies for Increased Engineering Design Complexity*) project. In a sub-populations based PGA, one is going to work with several goups of individuals ; for a given group (sub-population), individuals are going to evolve by means of the genetic operators, during a few generations and without interaction with the other groups. Then, the best individuals of each sub-populations are going to migrate (introduction of a new operator) to some other sub-populations, and so on. This approach generally results in improved convergence rates of the GA (as it has been demonstrated experimentally, see [1] and [10]) and more accurate results. The hybridization of GAs consists in coupling a GA with a gradient type method or a hill-climbing method (see [2]). Several approaches are set up in the litterature including the following : **(1)** the best solution found by the GA is taken as a starting point for the local method, **(2)** the gradient method can be incorporated in the GA as a new operator.

Finally, a number of extensions and improvements of the present work are concerned with the flow solver. First, a more accurate representation of the physical features of the flow will be achieved by solving the Navier-Stokes equations coupled to a turbulence model. Second, the efficiency of the flow solution could be notably improved (thus reducing the overall optimization cost) through the use of appropriate linear solvers such as multigrid methods.

Acknowledgements : the calculations presented in this paper have been conducted on the `IBM SP-2` system installed at the CNUSC (*Centre National Universitaire Sud de Calcul*) site of the IDRIS (*Institut du Développement et des Ressources en Informatique Scientifique*) institute and on the `SGI Origin 2000` system installed at the *Centre Charles Hermite* located in Nancy. The `Pentium Pro` cluster is an experimental platform located at INRIA Sophia Antipolis. For the work presented here, N. Marco was partially supported by DASSAULT-AVIATION under grant $\#AGT-7061/P-41606$. The authors thank B. Mantel and J. Périaux from DASSAULT-AVIATION for their help in the realization of this work.

References

[1] E. Cantù-Paz. A summary of research on parallel genetic algorithms. Technical Report 95007, IlliGAL Report, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, 1995.

[2] H.-V. Cao and G.-A. Blom. Navier-stokes/genetic optimization of multi-element airfoils. In *AIAA-96-2487, 14th AIAA applied aerodynamics conference*, New Orleans, 1996.

[3] C. Farhat and S. Lanteri. Simulation of compressible viscous flows on a variety of mpps: computational algorithms for unstructured dynamic meshes and performance results. *Comp. Meth. in Appl. Mech. and Eng.*, 119:35–60, 1994.

[4] L. Fezoui and B. Stoufflet. A class of implicit upwind schemes for Euler simulations with unstructured meshes. *J. of Comp. Phys.*, 84:174–206, 1989.

[5] D.E. Goldberg. *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley Company Inc., 1989.

[6] D.E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Second international conference on genetic algorithms : genetic algorithms and their applications*, pages 41–49. Morgan Kaufmann, 1987.

[7] J. Horn and N. Nafpliotis. Multiobjective optimization using the niched Pareto Genetic Algorithm. In *First IEEE conference on evolutionary computation, IEEE world congress on computational intelligence*, volume 1 (ICEC '94), 1994.

[8] H. Kuiper, A.J. Van der Wees, C.F. Hendriks, and T.E. Labrujere. Application of genetic algorithms to the design of airfoil pressure distribution. NLR Technical publication TP95342L for the ECARP European Project.

[9] S. Lanteri. Parallel solutions of compressible flows using overlapping and non-overlapping mesh partitioning strategies. *Parallel Comput.*, 22:943–968, 1996.

[10] H. Muhlenbein, M. Schomisch, and J. Born. The parallel genetic algorithm as function optimizer. *Parallel Comput.*, 17:619–632, 1991.

[11] S. Obayashi and A. Oyama. Three-dimensional aerodynamic optimization with genetic algorithm. In J.-A. Désidéri, C. Hirsch, P. Le Tallec, M. Pandolfi, and J. Périaux, editors, *Third ECCOMAS Computational Fluid Dynamics Conference and Second ECCOMAS Conference on Numerical Methods in Engineering*, volume Computational Fluid Dynamics '96, pages 420–424. John Wiley & Sons, 1996.

[12] C.K. Oei, D.E. Goldberg, and S.J. Chang. Tournament selection, niching and the preservation of diversity. Technical Report 91011, IlliGAL Report, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, 1992.

[13] J. Périaux, M. Sefrioui, B. Stoufflet, B. Mantel, and E. Laporte. Robust genetic algorithms for optimization problems in aerodynamic design. In M. Galàn G. Winter, J. Périaux and P. Cuesta, editors, *Genetic algorithms in engineering and computer science*, pages 371–396. John Wiley & Sons, 1995.

[14] L. Piegl and W. Tiller. *The NURBS book*. Springer Verlag, Berlin, 1997.

[15] C. Poloni. Hybrid GA for multi-objective aerodynamic shape optimization. In M. Galàn G. Winter, J. Périaux and P. Cuesta, editors, *Genetic algorithms in engineering and computer science*, pages 397–415. John Wiley & Sons, 1995.

[16] D. Quagliarella. Genetic algorithms applications in computational fluid dynamics. In M. Galàn G. Winter, J. Périaux and P. Cuesta, editors, *Genetic algorithms in engineering and computer science*, pages 417–442. John Wiley & Sons, 1995.

[17] P.L. Roe. Approximate Riemann solvers, parameters vectors and difference schemes. *J. of Comp. Phys.*, 43:357–371, 1981.

[18] P. Spiessens and B. Manderick. A massively parallel Genetic Algorithm implementation and first analysis. In Morgan Kaufmann, editor, *4th ICGA Intenational Conference on Genetic Algorithms*, pages 279–285, San Mateo (CA), 1991.

[19] B. Van Leer. Towards the ultimate conservative difference scheme V : a second-order sequel to Godunov's method. *J. of Comp. Phys.*, 32:361–370, 1979.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399