ABSTRACT

The design of the tubular frame of a chopper motorcycle is a challenging engineering problem due in part to the conflicting nature between the two main design criteria, namely, to minimize the frame mass and to minimize the maximum structural stress. The problem is even more complex under the presence of different kinds of decision parameters: discrete (e.g., standardized tube diameters available in the market) and continuous (e.g., angles and fillets). This project presents an optimization approach for the design of such frame topology based on evolutionary algorithms and employing finite element analysis for structural calculations. The proposed evolutionary algorithm deals with the two mentioned optimization objectives and the mixed decision parameters involved in the design. A communication routine between the structural simulation software and the evolutionary algorithm was developed to automate this hybrid technique. Frames obtained are compared to those of a previous study using a conventional design method, showing important progress in the frame’s performance.

1 INTRODUCTION

Finite element analysis (FEA) software is a powerful simulation tool, used to tackle a wide range of real world mechanical analysis problems. Unfortunately, the few commercial FEA programs that include optimization modules do not handle multiple criteria, and restrict the design parameters to a real domain.

Evolutionary algorithms (Michalewicz, 1996) are population-search techniques based on the principles of natural selection, used to solve complex optimization problems. Recently, some researchers have combined FEA and single-objective evolutionary algorithms to obtain good solutions in different engineering design problems. Abe et al. (2004) have used this procedure to improve the construction of tire reinforcements. Giger (2004) used a similar technique to design rims for GP1 racing motorcycles.

This project is part of an ongoing development. The main objective of this project is to design and build a low-cost chopper motorcycle. This work proposes a method based on the combination of ANSYS™ and a multiobjective evolutionary algorithm (MOEA) that focuses on the motorcycle’s frame design. The algorithm is based on a-priori articulation of preferences and is able to unveil motorcycle frame designs that balance the existing tradeoff between the frame mass and the maximum structural stress occurring during braking of the motorcycle. Our approach implements a mixed genotype that allows for real and integer design parameters, a single-point crossover operator, and a normally-distributed mutation operator. A communication routine between the FEA program and the MOEA has also been developed to automate the design process. This methodology could be easily extendable to many other multiobjective engineering design problems. Figure 1 briefly illustrates this process.

This paper is organized as follows. Section 2 describes the frame’s simulation using FEA. Section 3 deals with the description and implementation of the evolutionary algorithm and the interface between the FEA simulation and the MOEA. Section 4 presents the computational results. Section 5 outlines future research and highlights the main findings of the project.
2 FRAME’S FEA SIMULATION

An analysis of the frame’s components and their interactions is needed to obtain a lighter frame and improve its mechanical behaviour. Calderón (2004) obtained a first functional prototype of the motorcycle frame by using a traditional design approach, that is, to use manual intervention to fine tune the design parameters and run the simulation iteratively. However, this analysis can be done programmatically, simulating the performance of the frame using a FEA based software as a “black box” coupled with an optimization algorithm. In this study, we compared the latter approach with the traditional one based on manual “optimization” cycles.

The FEA simulation models the mechanical behavior of the frame under an extreme load case to evaluate its performance according to the previously mentioned criteria. The load case simulates the moments and forces transmitted to the frame, via the front fork and the rear suspension, present during abrupt braking (see Figure 2). Besides the inertial loads of the motorcycle produced by its deceleration, the analysis takes into account the weight and inertia of the pilot and engine (not a structural part of the frame). Since this scenario does not put the frame under considerable torsional loads, the torsional stiffness criterion is left out of the scope of the simulation. The complete finite element model and its analysis is done in ANSYS using its parametric design language APDL (ANSYS Inc., 2003), and follows the methodology used in a steady state structural analysis.

Table 1: Types of parameters for the simulation

<table>
<thead>
<tr>
<th>Discrete (10 parameters)</th>
<th>Continuous (12 parameters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Plates</td>
<td>- Angles</td>
</tr>
<tr>
<td>- Tubes</td>
<td>- Fillets</td>
</tr>
<tr>
<td></td>
<td>- Lengths</td>
</tr>
<tr>
<td></td>
<td>- Relative positions</td>
</tr>
</tbody>
</table>

Custom built geometries are cost prohibitive for the scope of this project. Therefore, plates and tubes are considered discrete parameters since these are commercially standardized. Eight plate thicknesses (6 mm, 8 mm, 10 mm, 12 mm, 16 mm, 20 mm, 25 mm, and 30 mm) (Avallon and Baumeister, 1997) are chosen as alternatives for four different parts of the frame, and eight tube commercial specifications (Avallon and Baumeister, 1997) are chosen for another seven parts (see Figure 3). Note that tubes are classified according to their outer diameter and schedule (standard measure of the tube’s wall thickness) (see Figure 4). Different orderings can be applied to the possible values these parameters can take (i.e., mass, length, first moment of inertia, polar moment of inertia, among others). Note that the distance between the possible values are not equally distributed. For instance, the distance in mass between the 6 mm and 8 mm plates is smaller than the distance in mass between the 25 mm and 30 mm plates. Even though it may not exist an absolute ordering, we indexed tubes according to their first moment of inertia and plates
to their mass. In each case, the possible values are represented by a list of integer numbers.

Figure 3: FEA model showing all discrete parameters and examples of continuous parameters

Figure 4: Commercial tubes selected for the various parts of the frame

We take advantage of the existing symmetry in both the frame geometry and the loads. These symmetries allow us to reduce in half the size of the FEA model (see Figure 3), causing significant savings in computational time for each simulation, without any loss in the mechanical analysis.

2.2 Meshing

Generating the finite element mesh involves selecting the element type and the level of refinement. Given the geometry’s complexity of the model, the domain is meshed with tetrahedrons using an APDL macro (divides lines of the model in proper segments) together with ANSYS’ automatic tool.

Table 2 summarizes the results for meshing tests using different elements and refinements. These tests are performed looking after a suitable combination producing reliable results. Structural percentage error in energy norm (ANSYS Inc., 2003) is taken as the convergence criterion (see column labeled SEPC). Note that the column labelled DOF corresponds to the degrees of freedom, which can be seen as a proxy of the FEA problem size. In accordance with this analysis, we chose element SOLID92 for the simulation.

Table 2: Meshing tests with different element types and level of mesh refinement

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Number of Elements</th>
<th>DOF</th>
<th>SEPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLID45</td>
<td>~ 100,000</td>
<td>~ 90,000</td>
<td>~ 32%</td>
</tr>
<tr>
<td>SOLID45</td>
<td>~ 300,000</td>
<td>~ 240,000</td>
<td>~ 27%</td>
</tr>
<tr>
<td>SOLID92</td>
<td>~ 80,000</td>
<td>~ 450,000</td>
<td>~ 17%</td>
</tr>
<tr>
<td>SOLID92</td>
<td>~ 100,000</td>
<td>~ 600,000</td>
<td>~ 13%</td>
</tr>
</tbody>
</table>

Once the model has been meshed, loads and boundary conditions are applied. Following Saint-Venant principle (Hibbeler, 1998), distributed loads are used to avoid extreme deformations inherent of point loads that may lead to incorrect results.

2.3 Solution

The main equations of a linear structural analysis are the stress-strain equations:

\[
\sigma = D \cdot \varepsilon
\]

where \( \sigma \) is the stress vector, \( D \) is the elastic stiffness matrix, and \( \varepsilon \) is the strain vector; and the principle of virtual work equations:

\[
\delta E_d = \delta W_1 + \delta W_2
\]

\[
\int_{Vol} \delta \varepsilon^T \cdot \sigma \, dVol = \int_{A_p} \delta \bar{u}^T \cdot \bar{P} \, dA_{pr} + \delta \bar{u}^T \cdot \bar{F}
\]

where \( E_d \) is the strain energy, \( W_1 \) is the external work produced by distributed loads, \( W_2 \) is the external work produced by point loads, \( d \) is the virtual operator, \( Vol \) is the structure’s volume, \( \bar{P} \) is the pressure vector, \( A_{pr} \) is the area over which the pressure acts, \( \bar{u} \) is the surface displacements vector, and \( \bar{F} \) is the force vector.

The FEA methodology generates a system of linear equations that approximates the structural behavior of the model described by Equations 1 and 2 (Reddy, 1993). The iterative PCG (Preconditioned Conjugate Gradient) solution (ANSYS Inc., 2003), is used to solve the system of linear equations. For each FEA solution, we calculate the
total mass of the frame (m) and the maximum Von Mises stress (s). These design criteria are later used in the fitness function of the evolutionary algorithm.

3 THE EVOLUTIONARY ALGORITHM

Evolutionary algorithms (EAs) are a family of stochastic search methods based on the principles of natural selection. EAs are used to solve a wide range of complex optimization problems. These algorithms gradually improve the fitness of a population of solutions by applying the principle of survival of the fittest. It is expected that the algorithm finds the global optimum solution as generations evolve. In EAs, potential solutions are encoded into an array that simulates a chromosome. Several evolutionary operators, such as recombination and mutation, are applied to the chromosomes. The purpose of these operators is to explore a vast solution space and exploit promising solutions. The population of the next generation is selected from the current population formed by parents and their children. Evolutionary algorithms in use today have evolved from the pioneering work by Holland (1975), Rechenberg (1973), Schwefel (1975), and Fogel, Owens and Walsh (1966). A good introductory material on evolutionary algorithms can be found in Michalewicz (1996). The reader is referred to Coello et. al. (2002) for an excellent introduction to multiobjective evolutionary optimization.

Figure 5 shows the evolutionary algorithm used in this work. The algorithm starts by randomly initializing the first generation of solutions \( P(1) \) and evaluating it (steps 2 and 3). Next, the main loop repeats until the maximum number of generations is reached. In the main loop, the mutation and crossover operators are applied to the parent population \( P(t) \) to create the children population \( Cm(t) \) (step 5) and \( Cc(t) \) (step 6), respectively. The total children population \( C(t) \) is evaluated (step 8). The selection operator is applied to the expanded population \( E(t) \) to create the parent population for the next generation \( P(t+1) \) (steps 9 y 10). This algorithm has been implemented using a flexible and extensible library called JGA (Medaglia and Gutiérrez, 2004).

1: \( t \leftarrow 1 \)
2: Initialize \( P(t) \)
3: Evaluate \( P(t) \)
4: while \( t \leq T \) do
5: Mutate \( P(t) \) and Generate \( Cm(t) \)
6: Recombine \( P(t) \) and Generate \( Cc(t) \)
7: \( C(t) \leftarrow Cm(t) \cup Cc(t) \)
8: Evaluate \( C(t) \)
9: \( E(t) \leftarrow P(t) \cup C(t) \)
10: Select \( P(t+1) \) from \( E(t) \)
11: \( t \leftarrow t + 1 \)
12: end while

**Figure 5:** Evolutionary algorithm

3.1 Data Structure (Genotype)

A vector of real and integer values (i.e., the chromosome) is used to codify the frame’s geometric parameters. Note that this structure allows for a direct abstraction of the problem posed.

\[
x^p = [x_{p}^0, x_{2}^p, ..., x_{2}^p, ..., x_{L}^p]^{T}, \quad i \in \{I_{R} \cup I_{Z}\}, \quad p = 1,...,N \quad (3)
\]

where \( x_{p}^p \) is the gene \( i \) of individual (chromosome) \( p \), \( L \) is the length of the chromosome, \( N \) is the population size, and \( I_{R} \) and \( I_{Z} \) are the index sets of real and integer parameters, respectively.

Based on an a-priori knowledge of interactions between parameters (i.e., nonlinearities), highly interactive genes are placed closed to each other in the chromosome. This careful positioning of genes along the chromosome reduces the possible disruption of well-performing building blocks produced by epistasis (Whitley, 1994).

3.2 Crossover Operator

Crossover (i.e., recombination) produces new individuals combining the information contained in two or more parents. Normally, this operator is applied only to a fraction of the parent population (i.e., crossover probability \( p_c \sim 60\%-100\% \)). The proposed algorithm uses a one-point crossover operator. This operator takes two individuals and cuts their chromosome at one random point, producing two “head” and two “tail” segments. The “tail” segments are interchanged to produce two new chromosomes, so the children inherit some of the genes of each parent.

3.3 Mutation Operator

The mutation operator is independently applied to every individual of the parent population, and alters each of their genes with probability \( p_m \). Several researchers (Bäck, 1993; Ochoa et al., 1999) agree that the optimum mutation rate depends basically on \( 1/L \) (i.e., the reciprocal of the chromosome’s length) and report good results using \( p_m = 1/L \) for a wide range of problems with binary genotypes. Although this problem uses a mixed real-integer genotype, it follows the same approach. Given the high computational cost of each FEA simulation (see Section 4), finding the optimum \( p_m \) via performing various experiments is unfeasible in practice.

The mutation operator is often designed to perform exploration (i.e., sampling unknown regions of the solution space). However, a mutation operator can also perform exploitation (i.e., improving the best individual found so far). We have designed the mutation operator to achieve a balance between exploration and exploitation.
For real-valued genes, we propose a normally-distributed change $\beta$ in $x_i^p$, with mean 0 and standard deviation given by a percentage of the gene’s domain. Note that the value of the changing gene $y_i^p$ is forced to remain in its allowed range of values as follows:

$$
y_i^p = \begin{cases} 
  x_i^p + \beta \text{ if } x_i \leq x_i^p + \beta \leq \bar{x}_i, & \chi \in \mathcal{X} \\
  x_i \text{ if } x_i^p + \beta > \bar{x}_i, & \chi \in \mathcal{X} \\
  x_i \text{ if } x_i^p + \beta < \underline{x}_i, & \chi \in \mathcal{X}
\end{cases}
$$

(4)

where $\beta \sim N(0, \sigma)$, $\sigma = r \cdot (\bar{x}_i - \underline{x}_i)$, $\bar{x}_i = \max_{\chi \in \mathcal{X}} \{x_i\}$, $\underline{x}_i$ is the domain of gene $x_i$, and $r$ is the range of mutation (e.g., $\sim 10\%$). This operator can cover a wide range of the gene’s domain in a single step, achieving good results quickly while preserving population’s diversity. At the same time, a large portion of the changing genes $y_i^p$ will be near the original gene $x_i^p$, which is highly desirable when the mutating individual is already well adapted (i.e., exploitation).

For integer-valued genes a similar approach is followed. In addition, considerations to allow the operator to work on an integer set are taken into account. The absolute value and the ceiling functions are applied to $\beta$, and an auxiliary variable $a$ is used to obtain both positive and negative changes in $x_i^p$:

$$
y_i^p = \begin{cases} 
  x_i^p + a \cdot \lceil |\beta| \rceil \text{ if } x_i \leq x_i^p + a \cdot \lceil |\beta| \rceil \leq \bar{x}_i, & \chi \in \mathcal{X} \\
  x_i \text{ if } x_i^p + a \cdot \lceil |\beta| \rceil > \bar{x}_i, & \chi \in \mathcal{X} \\
  x_i \text{ if } x_i^p + a \cdot \lceil |\beta| \rceil < \underline{x}_i, & \chi \in \mathcal{X}
\end{cases}
$$

(5)

where $\beta \sim N(0, \sigma)$; $a$ is a Bernoulli-distributed variable taking values of -1 and 1 with equal probability; $\sigma = r \cdot (\bar{x}_i - \underline{x}_i)$, $\bar{x}_i = \max_{\chi \in \mathcal{X}} \{x_i\}$, $\underline{x}_i = \min_{\chi \in \mathcal{X}} \{x_i\}$, $\bar{x}_i$ is the domain of gene $x_i$, and $r$ is the range of mutation (e.g., $\sim 10\%$).

### 3.4 Fitness Function

The problem described in this paper considers the optimization of two design criteria, usually in conflict. For instance, if mass $s$ increases it is expected that the maximum structural stress $\sigma$ decreases. This fact clearly suggests the use of a multiobjective optimization approach. Even though there exist stronger approaches to multiobjective evolutionary optimization (Coello et al., 2002), we have used a-priori articulation of preferences for its simplicity, ease of implementation, and computational efficiency. In a-priori articulation of preferences, the fitness function is calculated by means of a weighted sum that aggregates all the objectives into a single one. This method transforms the multiobjective optimization problem into a single-objective optimization problem. The fitness function for the $p$-th individual is calculated as follows:

$$
\tilde{f}(\bar{x}^p) = \lambda_1 \cdot f_1(\bar{x}^p) + \lambda_2 \cdot f_2(\bar{x}^p)
$$

(6)

where $\tilde{f}(\bar{x}^p)$ is the scalar fitness function; $f_i(\bar{x}^p)$ is the normalized mass function defined by:

$$
f_i(\bar{x}^p) = m(\bar{x}^p)/(|\max \{m\} - \min \{m\})), \ m \in \mathcal{M}
$$

(7)

where $\mathcal{M}$ is the domain of the mass $m$; $f_i(\bar{x}^p)$ is the normalized maximum stress function defined by:

$$
f_2(\bar{x}^p) = s(\bar{x}^p)/(|\max \{s\} - \min \{s\})), \ s \in \mathcal{S}
$$

(8)

where $\mathcal{S}$ is the domain of the stress $s$, and $\gamma_1$ and $\gamma_2$ are the weighting coefficients for each normalized function satisfying $\gamma_1 + \gamma_2 = 1$; $\gamma_1, \gamma_2 = 0$.

The algorithm’s output is a chromosome $\bar{x}^*$, which is expected to be Pareto optimum. Pareto optimality means that there exists no other feasible chromosome $\bar{x}^p$ in the solution space that could improve one criterion without causing a deterioration in the other criterion.

An important concern of the weighted sum approach is that it only produces Pareto optima when used in convex criteria spaces. However, problems solved with FEA are precisely those which have very complex criteria spaces and often lack of analytical mathematical formulations. This fact makes the formal proof of the convexity of the criteria space very difficult, forcing us to rely on experimental evidence to prove this hypothesis (see Section 4).

Maximum and minimum values for the design criteria domains (i.e., $\max \{m\}$, $\max \{s\}$, $\min \{m\}$ and $\min \{s\}$) are found experimentally using each design criteria as the sole fitness function for the evolutionary algorithm. For example, to find $\max \{s\}$ the fitness function is set to maximize $s(\bar{x}^p)$, and to find $\min \{s\}$ the fitness function is set to minimize $s(\bar{x}^p)$.

### 3.5 Implementation

The evolutionary algorithm and its interface with ANSYS were implemented in Java™. The evolutionary algorithm’s implementation uses the JGA, a generic tool implemented in Java for solving optimization problems using evolutionary algorithms (Medaglia and Gutierrez, 2004).
3.5.1 JGA

JGA allows the user to implement a particular evolutionary algorithm, based on a previously built architecture and a set of components found in the library. JGA considerably reduces the programming effort due to the fact that the main components of the evolutionary algorithm are already available in the library.

Figure 6 illustrates the JGA architecture. The specific algorithm components implemented for this project are:

- Class FrameFitness: this class extends the library class FitnessFunction. It also evaluates the performance of each design by connecting to the frame’s simulation in ANSYS (via class Frame described in Section 3.5.2).
- Class RZGenotype: this class extends the library class Genotype. It implements the mixed data structure that can hold both real and integer parameter values.
- Classes RZMutation and SinglePointRZCrossover: they extend classes MutationOperator and CrossoverOperator, respectively. They implement the proposed evolutionary operators outlined in Sections 3.2 and 3.3.
- Class RZMain: This is the main class for the evolutionary algorithm. It loads the configuration data and executes the algorithm calling its handler provided by the library.
- File RZSettings.ini: This file contains the type (either R or Z) and the range for each of the 22 controlled parameters in the frame design.

3.5.2 Interface

The interface between the evolutionary algorithm and ANSYS is implemented within class Frame. Instances of class Frame are used within class FrameFitness. Through a series of methods, each individual’s chromosome is converted into ADPL language and stored in a file. Then, an ANSYS simulation is executed in batch mode and a plain text file with the individual’s performance (i.e., m and s) is produced. Finally, the file with the individual’s performance is read and class FrameFitness calculates the proposed fitness function. Class Frame also includes a method that keeps a record of the population’s evolution as generations pass by. This information is later used to study the convergence of the evolutionary algorithm.

4 COMPUTATIONAL RESULTS

Extensive experimental tuning of the various algorithm parameters is not practical, given that each FEA simulation is time consuming (~8 minutes in computer with an Intel Pentium IV CPU running at 2.8GHz with 1GB of RAM). For this reason, parameter selection is based on results of previous studies (Deb, 2003; Coello et al., 2002) that have used evolutionary algorithms similar to the one proposed in this project. These studies also use evolutionary algorithms with elitist selection (see BestIndividualSelection in Figure 6 for our JGA implementation) and inter-generational competition, that is, after applying the evolutionary operators, the best N individuals among the combined parent and children population E(t) are selected for the next generation P(t+1). The parameters used in the proposed evolutionary algorithm are summarized in Table 3. We found that for T=50 the great majority of genes have converged (i.e., approximately 90% of the population share the same genes).

Table 3: Evolutionary algorithm parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size (N)</td>
<td>40</td>
</tr>
<tr>
<td>Maximum number of generations (T)</td>
<td>50</td>
</tr>
<tr>
<td>Crossover rate (p_c)</td>
<td>0.6</td>
</tr>
<tr>
<td>Mutation rate (p_m)</td>
<td>1/22</td>
</tr>
</tbody>
</table>

Table 4 shows a set of solutions for the frame’s design using the evolutionary approach and Calderón’s conventional method (Calderón, 2004). Designs A, B and C correspond to the solutions obtained with the evolutionary algorithm. Therefore, these solutions are expected to be on the Pareto optimal front or at least very close to it. The experimental evidence does not suggest that the criteria space is non convex. Recall that in the absence of convexity, an aggregated objective expressed by the weighted sum in the fitness function may lead to some problems (see Section 3.4).

Figure 5 graphically illustrates how Calderón’s design is dominated by solutions A and B. Compared to Calderón’s design, design B presents a 15% mass (m) reduction and a 26% stress (s) reduction; whereas design A presents a 33% mass reduction and a 5% stress reduction. It is in the hands of the designer to decide which is the best
frame among designs A, B and C. If a lighter frame and savings in manufacturing materials are desired, a higher priority would be given to reductions in mass ($m$). On the other hand, if a higher safety factor is more important, then a higher priority would be given to reductions in stress ($s$).

Figure 6 illustrates the ANSYS output of the Von Mises stress distribution for design B. Note that in Figure 6, zones in the frame with cold colors (blue or black) mean lower stress levels than zones with warmer colors (red or white).

Table 4: Computational results

<table>
<thead>
<tr>
<th>Solution</th>
<th>$\omega_1$</th>
<th>$\omega_2$</th>
<th>$s$ (MPa)</th>
<th>$m$ (Kg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calderón</td>
<td>-</td>
<td>-</td>
<td>176</td>
<td>23.6</td>
</tr>
<tr>
<td>A</td>
<td>0.50</td>
<td>0.50</td>
<td>166</td>
<td>17.8</td>
</tr>
<tr>
<td>B</td>
<td>0.37</td>
<td>0.63</td>
<td>130</td>
<td>20.2</td>
</tr>
<tr>
<td>C</td>
<td>0.25</td>
<td>0.75</td>
<td>106</td>
<td>26.5</td>
</tr>
</tbody>
</table>

Figure 5: Criteria space showing performance of designs produced with the MOEA vs. a conventional method.

It is important to remember that the motorcycle will perform under dynamic loads in its life cycle. For this reason, a fatigue analysis for the selected frame design has to be conducted in order to find its endurance limit. This can be done using conventional fatigue analysis methods (see Shigley and Mischke, 2001) or via an additional FEA study.

5 CONCLUSIONS AND FUTURE RESEARCH

Some mechanical design conclusions can be elicited from studying the three frame designs found by the evolutionary algorithm. Tubes combining large outer diameters and thin walls seem to produce better results. Tubes perpendicular to the yaw and roll axes of the motorcycle (tubes 3, 5, and 7 in Figure 3) greatly contribute to the structural performance of the frame. Not only do they support loads produced by the engine and pilot, but lower the stress levels in other parts of the frame. The supporting plate proposed by Calderón (2004) (plate 1 in Figure 3) can be removed if the spacing at the frame’s head between tubes 1 and 4 in Figure 3 is increased. A large enough tube spacing at the frame’s head significantly reduces stress in this spot.

Current computer hardware capabilities allow the use of population-search techniques for solving fairly complex design optimization problems involving FEA. However, problems with millions of degrees of freedom (DOF) are common in today’s FEA applications. The long computer time these FEA problems demand (even using parallel architectures) puts a practical limit to the combined use of EAs and FEA.

Due to the high computational cost of each FEA simulation, the construction of a much quicker approximate evaluation function is of particular interest. We are currently exploring the possibility of designing an evolutionary algorithm that combines FEA simulations with approximate evaluations. This approach could possibly find a better chromosome in a limited computer time. Possible methods for constructing an approximate evaluation function are linear regression using ordinary least-squares or artificial neural networks.

The main drawback of the evolutionary algorithm used in this paper is that it only produces one Pareto optimal solution per run. In addition, the convexity requirement for the criteria space (given by the use of a weighted sum in the fitness function) is always a concern. Recent evolution-
ary approaches (Coello et al., 2002) are able to produce an entire set of approximate Pareto optimal solutions in a single run, without imposing any restrictions on the shape of the criteria space. Among these new approaches is the NSGA II proposed by Deb et al. (2002). This algorithm has been successfully used in various shape optimization problems (e.g., Deb, 2003).

As an alternative approach to the issue of tuning the algorithm’s parameters (i.e., high computational cost of manual adjustment), a dynamic parameter control could be devised. This would involve an automatic tuning of the key parameters (usually \(p_c\), \(p_m\) and \(N\)) while the algorithm is running. Over the last years there has been extensive research on this topic and different methods have been suggested (Eiben et al., 1999; Eiben et al, 2004). These include control techniques based on algorithm’s feedback, auto-adaptation (by embedding algorithm parameters in the chromosome) and heuristics (including EAs).

REFERENCES


AUTHOR BIOGRAPHIES

JORGE ENRIQUE RODRIGUEZ received a B.Sc. (2004) in Mechanical Engineering and a B.Sc. (2005) in Industrial Engineering from Universidad de los Andes. He is currently a M.Sc. student in Industrial Engineering and a member of the Centro para la Optimización y Probabilidad Aplicada (COPA) at Universidad de los Andes. He can be contacted by e-mail at jorg-rod@uniandes.edu.co.

ANDRÉS MEDAGLIA received a B.Sc. (1992) in Industrial Engineering from Universidad Javeriana and a M.Sc.
(1995) in Industrial Engineering from Universidad de los Andes. He holds a Ph.D. (2001) in Operations Research from North Carolina State University. In 2001-2002, he worked as a postdoctoral fellow in the Industrial Engineering Department at N.C. State, sponsored by SAS Institute Inc. (Cary, NC). In 2002, he was appointed Assistant Professor of the Industrial Engineering Department at Universidad de los Andes. He is a founding member of the Centro para la Optimización y Probabilidad Aplicada (COPA). He can be contacted by e-mail at amedagli@uniandes.edu.co and his web address is http://www.prof.uniandes.edu.co/~amedagli/.

JUAN PABLO CASAS received a B.Sc. (1997) in Mechanical Engineering and a M.Sc. (1999) in Mechanical Engineering from Universidad de los Andes. In 2002, he was appointed Instructor and Research Associate of the Mechanical Engineering Department at Universidad de los Andes. He can be contacted by e-mail at jcasas@uniandes.edu.co and his web address is http://mecanica.uniandes.edu.co/manager.php?id=153