A Fitness Granulation Approach for Large-Scale Structural Design Optimization

Mohsen Davarynejad, Jos Vrancken, Jan van den Berg, and Carlos A. Coello Coello

Abstract The complexity of large-scale mechanical optimization problems is partially due to the presence of high-dimensional design variables, the nature of the design variables, and the high computational cost of the finite element simulations needed to evaluate the fitness of candidate solutions. Evolutionary cycles are ruled by competitive games of survival and not merely by absolute measures of fitness, as well as exploiting the robustness of evolution against uncertainties in the fitness function evaluations. This paper takes up the complexity challenge of mechanical optimization problems by proposing a new fitness granulation approach that attempts to cope with many difficulties of fitness approximation approaches that have been reported in the specialized literature. The approach is based on adaptive fuzzy fitness granulation having as its main aim to strike a balance between the accuracy and the utility of the computations. The adaptation algorithm adjusts the number and size of the granules according to the perceived performance and level of convergence attained. Experimental results show that the proposed approach accelerates the convergence towards solutions, when compared to the performance of other, more popular approaches. This suggests its applicability to other complex finite element-based engineering design problems.

Mohsen Davarynejad, Jos Vrancken & Jan van den Berg
Faculty of Technology, Policy and Management, Delft University of Technology, NL-2600 GA, Delft, The Netherlands, e-mail: {m.davarynejad;j.l.m.vrancken;j.vandenberg}@tudelft.nl

Carlos A. Coello Coello
CINVESTAV-IPN, Departmento de Computación (Evolutionary Computation Group), Av. IPN No. 2508, Col. San Pedro Zacatenco, México D.F. 07300, México e-mail: ccoello@cs.cinvestav.mx

1 Introduction

Since the 1960s, and due to the significant developments in numerical methods and computing, the finite element analysis (FEA) became a frequent tool to solve engineering problems that arise in systems with several interacting components, complex geometries, and which are under the effect of different physical phenomena. These (complex) systems elude a thorough physical analysis with exact techniques which is made possible by means of a systematic discretization approach known as the finite element method (FEM) [1]. At the same time that the FEM was developed, efficient and fast optimization algorithms have arisen for solving various kinds of mathematical and optimization problems (OPs). Both trends contributed to the development of large-scale structural design and optimization problems (SDOPs) and to the discipline of structural optimization. The aim of structural optimization is to generate automated procedures for finding the best possible structure with respect to at least one criterion (the objective), and having to satisfy a set of constraints, by selecting from a set of geometrical dimensions, material properties and/or topological parameters [2].

Structural optimization problems are often challenging due to their high computational demands [1], multi-modality, non-convexity, high dimensionality, and multi-objectivity. Because of this, many structural optimization problems are weakly amenable to conventional mathematical programming approaches, which motivates the use of alternative solution methods.

Randomized search heuristics are among the simplest and most robust strategies that are applicable to a wide range of optimization problems including structural design (SD). While they can normally provide nearly optimal solutions, they cannot guarantee convergence to the optimum. However, their computational requirements are normally high. Among the randomized search heuristics currently available, evolutionary algorithms (EAs) have become very popular in the last few years, mainly because of their ease of use and efficacy. EAs are stochastic search techniques which operate on a set of solutions (the so-called population), that are modified based on the principles of the natural evolution (i.e., the survival of the fittest) [3]. EAs have been commonly adopted for solving complex SD problems. For example, Walker and Smith [4] combined the FEM and EAs to minimize a weighted sum of the mass and deflection of fiber-reinforced structures. Similarly, Abe et al. [5] used FEM and an EA for structural optimization of the belt construction of a tire. More recently, Giger and Ermanni [6] applied FEM and and EA to minimize the mass of composite fiber-reinforced plastic (CFRP) rims subject to strength and stiffness constraints. However, EAs may suffer from a slow rate of convergence towards the global optimum, which implies that they may be too (computationally) expensive for certain SD problems. Consequently,

[1] Finite element analysis is computationally costly and may require several days to complete its calculations, even for a relatively simple problem.

it is challenging to develop computationally efficient evolution-based search methods.

To alleviate the problem of converging time of computationally expensive optimization problems, a variety of techniques has been proposed in the literature. Perhaps the most obvious choice is to use parallelization techniques [7]. However, another alternative is to rely on fitness approximation techniques, which avoid evaluating every individual in the population of an EA. In order to do this, these approaches estimate the quality of some individuals, based on an approximate model of the fitness landscape. This is the sort of approach on which this chapter is focused. Section 4 provides a review of fitness approximation techniques in evolutionary computation. When using fitness approximation techniques, it is necessary to strike a balance between exact fitness evaluation and approximate fitness evaluation. In this chapter, with a view to reducing computational cost, we employ the concept of fuzzy granulation to effectively approximate the fitness function. The advantages of this approach over others is the fact that no training samples are required, and the approximate model is dynamically updated with no or negligible overhead cost.

The remainder of this chapter is organized as follows. The following section elaborates upon four SD optimization problems before explaining the genetic algorithm (GA) approach proposed here for the SD optimization task (see Section 3). This is followed by a review of the variety of fitness approximation approaches that have been proposed for EAs in Section 4. In order to accelerate the convergence speed of the GA with a minimum number of fitness function evaluations, a novel method is presented in Section 5. The approach is based on generating fuzzy granules via an adaptive similarity analysis. To illustrate the efficiency of the proposed method in solving the four SD problems introduced in Section 2, the performance results of different optimization algorithms are presented in Section 6. A further statistical analysis confirms that the proposed approach reduces the computational complexity of the number of fitness function evaluations by over 50% while reaching similar or even better final fitness values. Finally, in Section 8 we provide our conclusions.

2 Structural design optimization problems

Four SD optimization problems, with increasing complexity are investigated here. They are the following: (1) the design of a *3-layer composite beam* with *two* optimization variables, (2) the design of an *airplane wing* with *six* decision variables, (3) the design of a *2D truss* frame with *36* decision variables, and (4) the voltage/pattern design of piezoelectric actuators. We discuss in more detail the last problem, because of its complexity. Such a problem consists of finding the best voltage and pattern arrangement for static shape

control of a piezoelectric actuator with *200* design variables. Clearly, this is a more challenging and heavy optimization task from a fitness/computational perspective.

2.1 Easier/Smaller problems

The first three SD problems are covered in this section. The ultimate goal in these optimization problems is to maximize the first natural frequency [2] of the given structure. To allow more space for the last problem (described in Subsections 2.2 and 6.4), we limit ourselves here to a short description of the other problems.

2.1.1 3-Layer composite beam

A *multi-layered composite beam* is constructed from a combination of two or more layers of dissimilar materials that are joined together to act as a unit in which the resulting combination is lighter, stronger and safer than the sum of its parts. A finite element analysis model has been developed to analyze the multi-layer composite beams and plates. The objective is to raise the first natural frequency of the beam.

2.1.2 Airplane wing

An *airplane wing* is an elastic structure that, in the presence of aerodynamic loads, starts to vibrate. In this study, we treated the natural frequency as the design objective since it is quite intuitive and natural to raise the natural frequencies of the wing so that it is not easily excited by undesirable disturbances.

2.1.3 2D truss frame

Trusses are the most commonly used structure and in comparison to heavily-built structures, they have a relatively small *dead weight*. A truss consists of *bar-elements* (members) connected by *hinged joints* to each other and supported at the base. Truss design problems belong to the class of load-supporting structure design problems that are usually finite-dimension opti-

[2] Resonance occurs when the excitation frequency is the same as the natural frequency. For the same excitation energy, the resulting vibration levels at resonance frequency is higher than other exiting frequencies. The importance of maximizing the first natural frequency is to avoid the resonance phenomenon to occur.

mization problems. The design of load-supporting structures plays a key role in engineering dynamics. The objective (fitness) here is to raise the structure's *first natural frequency* to reduce the vibration domain and to prevent the resonance phenomenon (in dynamic response) of the structure.

2.2 Voltage and pattern design of a piezoelectric actuator

Piezoelectric materials exhibit both direct (electric field generation as a response to mechanical strains) and converse (mechanical strain is produced as a result of an electric field) piezoelectric effects. The direct effect is used in piezoelectric sensors while the converse effect is used in piezoelectric actuators.

Apart from ultrasound applications, energy harvesting, sensor applications (e.g., strain gauges and pressure sensors), and vibration/noise control domains, piezoelectric materials are widely used as actuators in smart structures. Smart structures with integrated self-monitoring, self-diagnosis and control capabilities have practical uses ranging from MEMS, biomedical engineering, control engineering, aerospace structures, ground transportation systems and marine applications. The smart structures' technology is widely used in biomechanics, i.e., to expand obstructed blood vessels or to prevent further enlargement of blood vessels damaged by aneurysms [8] which most commonly occurs in arteries. Another apparent practical use of smart and adaptive structural systems is to properly control the undesirable motions of geometry-changing structures.

Piezoelectric actuators are also found in an enormous range of applications for distributed actuation and control of mechanical structures for shape correction and modification. One example for this is their use in flexible aircrafts where they improve the aerodynamic performance and deformation control of conformal antennas [9], through their incorporation within the structure. For instance, in [10], an optimization algorithm is used to deal with the shape control of functionally graded material (FGM) plates which are actively controlled by piezoelectric sensor and actuator patches. A computational intelligence-based algorithm is used to derive the optimal voltage distribution, by adopting the elements of the gain control matrix as the design variables.

The optimal shape control and correction of small displacements in composite structures using piezoelectric actuators concern complex engineering problems. To achieve a predefined shape of the structure of the metal plate, in this chapter we will present a fast converging global optimization algorithm to find the optimal actuation voltages that need to be applied to the piezoelectric actuators and to the pattern of piezoelectric patches.

3 GAs in structural optimization problems

Genetic algorithms (GAs) are perhaps the most popular type of EAs nowadays and have been applied to a wide variety of problems [11]. The GA optimization procedure for solving SD problems begins with a set of randomly selected parents (design parameters). If any of these parents does not meet all the physical constraints, they are modified until they do. In subsequent generations, each offspring's phenotype is also checked for its feasibility. Furthermore, the fitness values of the parents and their offspring are compared and the worst individuals are rejected, preserving the remaining ones as parents of the new generation (known as steady-state population treatment). This procedure is repeated until a given termination criterion is satisfied.

Due to their robustness, GAs have been frequently used in a variety of real world optimization applications including optimizing the placement of actuators on large space structures [12], the design of a low-budget lightweight motorcycle frame with superior dynamic and mechanical properties [13], and the evolution of the structural configuration for weight minimization of a space truss structure [14]. The implementation of a GA can be summarized as follows:

1. **Initialization**: Initialize P design vectors $X = \{X_1, X_2, \ldots, X_i, \ldots, X_P\}$, where P is the population size.
2. **Constraints check**: If satisfied, continue, else modify X_i until the candidate solution becomes feasible.
3. **Evaluation (Analysis)**: Evaluate the fitness function $f(X_i),\ i = \{1, 2, \ldots, P\}$.
4. **Convergence check**:
   a. **if satisfied** stop,
   b. **else** select the next generation parent design vectors, apply genetic operators (mutation, recombination) and generate the next offspring design vectors X. Go to step 2.

EAs in general are often expensive in the sense that they may require a high number of computationally costly objective function evaluations. As a result, it may be necessary to forgo an exact evaluation and use approximated fitness values that are computationally efficient. In the design of mechanical structures, for instance, each exact fitness evaluation requires the time-consuming stage of FEA which, depending on the size of the problem, may consume from several seconds up to several days. If we assume a conventional genetic algorithm with a fixed and modest population size of 100, a maximum of 100 generations, and a very small-scale structural problem that requires 10 seconds for each fitness evaluation, the total execution of the GA would require 30 hours! This should make evident the inhibiting role of the computational complexity associated to GAs (and EAs, in general) for more non-trivial and large-scale problems.

Since one of the crucial aspects for solving large-scale SD optimization problems using EAs is the computational time required, in the following section we outline a few existing strategies that have been proposed to deal with this issue.

4 Fitness Approximation in Evolutionary Computation

As indicated before, one possibility to deal with time-consuming problems using a GA is to avoid evaluating every individual and estimate instead the quality of some of them based on an approximate model of the search space. Approximation techniques may estimate individuals' fitness on the basis of previously observed objective function values of *neighboring* individuals. There are many possible approximation models [15]. Next, we will briefly review some of the most commonly adopted fitness approximation methods reported in the specialized literature.

4.1 Fitness Inheritance

This is a very simple technique that was originally introduced by Smith et al. [16]. The mechanism works as follows: when assigning fitness to an individual, some times we evaluate the objective function as usual, but the rest of the time, we assign fitness as an average (or a weighted average) of the fitness of the parents. This fitness assignment scheme will save us one fitness function evaluation, and operates based on the assumption of similarity between an offspring and its parents. Clearly, fitness inheritance cannot be applied all the time, since we require some true fitness function values in order to obtain enough information to guide the search. This approach uses a parameter called *inheritance proportion*, which regulates how many times do we apply fitness inheritance (the rest of the time, we compute the true fitness function values). As will be seen next, several authors have reported the use of fitness inheritance.

Zheng et al. [17] used fitness inheritance for codebook design in data compression techniques. They concluded that the use of fitness inheritance did not degrade, in a significant way, the performance of their GA.

Salami et al. [18] proposed a Fast Evolutionary Strategy (FES) in which a fitness and associated reliability value were assigned to each new individual. Considering two decision vectors $p_1^i = (X_1^i, F_1^i, r_1^i)$ and $p_2^i = (X_2^i, F_2^i, r_2^i)$ where X_1^i and X_2^i are the chromosomes 1 and 2 at generation i with fitness values F_1^i and F_2^i and reliabilities r_1^i and r_2^i, respectively. In this scheme, the true fitness function is only evaluated if the reliability value is below a certain

threshold. Otherwise, the fitness of the new individual and its reliability value is calculated from:

$$F^{i+1} = \frac{S_1 r_1^i F_1^i + S_2 r_2^i F_2^i}{S_1 r_1^i + S_2 r_2^i} \tag{1}$$

and

$$r^{i+1} = \frac{(S_1 r_1^i)^2 + (S_2 r_2^i)^2}{S_1 r_1^i + S_2 r_2^i} \tag{2}$$

where S_1 is the similarity between X_1^{i+1} and X_1^i and S_2 is the similarity between X_1^{i+1} and X_2^i. Also, they incorporated random evaluation and error compensation strategies. Clearly, this is another (more elaborate) form of fitness inheritance. Salami et al. reported an average reduction of 40% in the number of evaluations while obtaining similar solutions. In the same work, they presented an application of (traditional) fitness inheritance to image compression obtaining reductions ranging from 35% up to 42% of the total number of fitness function evaluations.

Pelikan et al. [19] used fitness inheritance to estimate the fitness for only part of the solutions in the Bayesian Optimization Algorithm (BOA). They concluded that fitness inheritance is a promising concept, because population-sizing requirements for building appropriate models of promising solutions lead to good fitness estimates, even if only a small proportion of candidate solutions is evaluated using the true fitness function.

Fitness inheritance has also been used for dealing with multi-objective optimization problems. Reyes-Sierra and Coello Coello [20, 21] incorporated the concept of fitness inheritance into a multi-objective particle swarm optimizer and validated it in several test problems of different degrees of difficulty. They generally reported lower computational costs, while the quality of their results improved in higher dimensional spaces. This was in contradiction with other studies (e.g., [22] as well as this chapter) that indicate that the performance of the parents may not be a good predictor for their children's composition in sufficiently complex problems, rendering fitness inheritance inappropriate under such circumstances.

4.2 Surrogates

A common approach to deal with expensive objective functions is to construct an approximation function which is much cheaper to evaluate (computationally speaking). In order to build such an approximation function which will be used to predict promising new solutions, several sample points are required. The meta-model built under this scheme aims to reduce the total number of

(true objective function) evaluations performed, while producing results of a reasonably good quality.

Evidently, the accuracy of the surrogate model depends on the number of samples provided (and their appropriate distribution) and on the approximation model adopted. Since surrogate models will be used very frequently, it is very important that the construction of such models is computationally efficient [15]. The following are examples of the use of surrogates of different types.

Sano et al. [23] proposed a genetic algorithm for optimization of continuous noisy fitness functions. In this approach, they utilized the history of the search to reduce the number of fitness function evaluations. The fitness of a novel individual is estimated using the fitness values of the other individuals as well as the sampled fitness values for it. So, as to increase the number of individuals adopted for evaluation, they not only used the current generation but also the whole history of the search. To utilize the history of the search, a stochastic model of the fitness function is introduced, and the maximum likelihood technique is used for estimation of the fitness function. They concluded that the proposed method outperforms a conventional GA in noisy environments.

Branke et al. [24] suggested the use of local regression for estimation, taking the fitness of neighboring individuals into account. Since in local regression it is very important to determine which solutions belong to the neighborhood of a given individual, they studied two different approaches for setting the value of the size of the local neighborhood (relative neighborhood and adaptive neighborhood). They concluded that local regression provides better estimations than previously proposed approaches. In more recent work [25], a comparison between two estimation methods, interpolation and regression, is done. They concluded that regression seems to be slightly preferable, particularly if only a very small fraction of the individuals in the population is evaluated. Their experiments also show that using fitness estimation, it is possible to either reach a better fitness level in a given time, or to reach a desired fitness level much faster (using roughly half of the original number of fitness function evaluations).

Ong et al. [26] proposed a local surrogate modeling algorithm for parallel evolutionary optimization of computationally expensive problems. The proposed algorithm combines hybrid evolutionary optimization techniques, radial basis functions, and trust-region frameworks. The main idea of the proposed approach is to use an EA combined with a feasible sequential quadratic programming solver. Each individual within an EA generation is used as an initial solution for local search, based on Lamarckian learning. They employed a trust-region framework to manage the interaction between the original objective and constraint functions and the computationally cheap surrogate models (which consist of radial basis networks constructed by using data points in the neighborhood of the initial solution), during local search. Extensive numerical studies are presented for some benchmark test functions and an

aerodynamic wing design problem. They show that the proposed framework provides good designs on a limited computational budget. In more recent work, Ong et al. [27] presented a study on the effects of uncertainty in the surrogate model, using what they call Surrogate-Assisted Evolutionary Algorithms (SAEA). In particular, the focus was on both the *curse of uncertainty* (impairments due to errors in the approximation) and *blessing of uncertainty* (benefits of approximation errors). Several algorithms are tested, namelly the Surrogated-Assisted Memetic Algorithm (SAMA) proposed in [26], a standard genetic algorithm, a memetic algorithm (considered as the standard hybridization of a genetic algorithm and the feasible sequential quadratic programming solver used in [26]), and the SAMA-Perfect algorithm (which is the SAMA algorithm but using the exact fitness function as surrogate model), on three multi-modal benchmark problems (Ackley, Griewank and Rastrigin). The conclusion was that approximation errors lead to convergence at false global optima, but turns out to be beneficial in some cases, accelerating the evolutionary search.

Regis and Shoemakes [28] developed an approach for the optimization of continuous costly functions that uses a space-filling experimental design and local function approximation to reduce the number of function evaluations in an evolutionary algorithm. The proposed approach estimates the objective function value of an offspring by means of a function approximation model over the k-nearest previously evaluated points. The estimated values are used to identify the most promising offspring per function evaluation. A Symmetric Latin Hypercube Design (SLHD) is used to determine initial points for function evaluation, and for the construction of the function approximation models. They compared the performance of an Evolution Strategy (ES) with local quadratic approximation, an ES with local cubic radial basis function interpolation, an ES whose initial parent population is obtained from a SLHD, and a conventional ES (in all cases, the They used a (μ, λ)-ES with uncorrelated mutations). The algorithms were tested on a groundwater bioremediation problem and on some benchmark test functions for global optimization (including *Dixon-Szegö*, *Rastrigin* and *Ackley*). The obtained results (which include analysis of variance to provide stronger and solid claims regarding the relative performance of the algorithms) suggest that the approach that uses SLHDs together with local function approximations has potential for success in enhancing EAs for computationally expensive real-world problems. Also, the cubic radial basis function approach appears to be more promising than the quadratic approximation approach on difficult higher-dimensional problems.

Lim et al. [29] presented a Trusted Evolutionary Algorithm (TEA) for solving optimization problems with computationally expensive fitness functions. TEA is designed to maintain good trustworthiness of the surrogate models in predicting fitness improvements or controlling approximation errors throughout the evolutionary search. In this case, the most interesting part was to predict search improvement as opposed to the quality of the

approximation, which is regarded as a secondary objective. TEA begins its search using the canonical EA, with only exact function evaluations. During the canonical EA search, the exact fitness values obtained are archived in a central database together with the design vectors (to be used later for constructing surrogate models). After some initial search generations (specified by the user), the trust region approach takes place beginning from the best solution provided by the canonical EA. The trust region approach uses a surrogate model (radial basis neural networks) and contracts or expands the trust radius depending on the ability of the approximation model in predicting fitness improvements, until the termination conditions are reached. An empirical study was performed on two highly multi-modal benchmark functions commonly used in the global optimization literature (*Ackley* and *Griewank*). Numerical comparisons to the canonical EA and the original trust region line search framework are also reported. From the obtained results, the conclusion was that TEA converges to near-optimum solutions more efficiently than the canonical evolutionary algorithm.

4.2.1 Kriging

A more elaborate surrogate model that has been relatively popular in engineering is the so-called Gaussian Process Model, also known as Kriging [30]. This approach builds probability models through sample data and estimates the function values at every untested point with a Gaussian distribution.

Ratle [31] presented a new approach based on a classical real-encoded genetic algorithm for accelerating the convergence of evolutionary optimization methods. A reduction in the number of fitness function calls was obtained by means of an approximate model of the fitness landscape using kriging interpolation. The author built a statistical model from a small number of data points obtained during one or more generations of the evolutionary method using the true fitness landscape. The model is updated each time a convergence criterion is reached.

4.3 Artificial Neural Networks

In the last few years, artificial neural networks (ANNs), including multi-layer perceptrons [32] and radial basis function networks [33] have also been employed to build approximate models for design optimization. Due to their universal approximation properties, ANNs can be good fitness function estimators if provided with sufficient structural complexity and richness in their training data set. Next, some representative applications of the use of ANNs for building approximate models will be briefly reviewed.

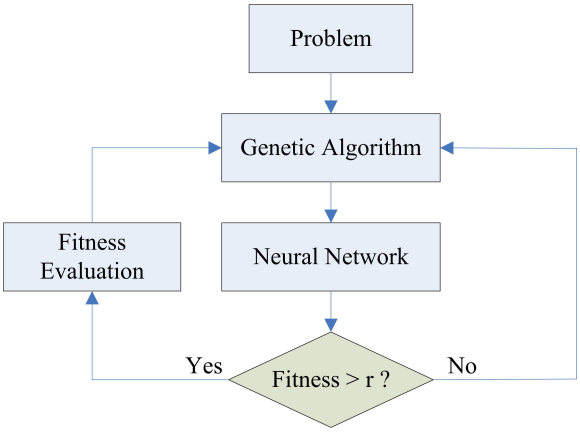

Fig. 1: The GA-ANN algorithm that is proposed in [34]. Only if the approximate fitness of an individual is better than the maximum fitness found in the last population, its fitness is re-evaluated in order to reflect its real fitness value.

Khorsand et al. [34] investigated structural design by a hybrid of neural network and finite element analysis that only selectively used the neuro-estimation when either interpolation was expected (interpolation is generally expected to be more accurate) or the individual was not deemed to be highly fit (error in estimation may not be important). The methodology used in [34] is presented in Figure 1 where r is considered as the maximum fitness of the individuals in the last generation. As with any other numerically driven approximation method, the performance of ANNs is closely related to the quality of the training data.

Jin et al. [35] investigated the convergence properties of an evolution strategy with neural network-based fitness evaluations. In this work, the concept of *controlled evolution* is introduced, in which the evolution proceeds using not only the approximate fitness function value, but also the true fitness function value. They also introduce two possibilities to combine the true with the approximate fitness function value: (1) the controlled individuals approach and (2) the controlled generation approach. Jin et al. defined "controlled" as evaluated with the true fitness function. Both approaches were studied and some interesting conclusions/recommendations for the correct use of such techniques are provided. A comprehensive survey of fitness approximation applied in evolutionary algorithms is presented in [36].

4.4 Final Remarks About Fitness Approximation

Lack of sufficient training data is the main problem in using most of the fitness approximation models currently available and hence the failure to reach a model with sufficient approximation accuracy. Since evaluation of the original fitness function is very time consuming and/or expensive, the approximate model may be of low fidelity and may even introduce false optima. Furthermore, if the training data does not cover all the domain range, large errors may occur due to extrapolation. Errors may also occur when the set of training points is not sufficiently dense and uniform. In such situations, a combination of methods may be more desirable. For example, Ong et al. [26] combined radial basis functions with transductive inference to generate local surrogate models.

Alternatively, if individuals in a population can be clustered into several groups as in [37], then only the individual that represents its cluster can be evaluated. The fitness value of other individuals in the same cluster will be estimated from the representative individual based on a distance measure. This is termed fitness imitation in contrast to fitness inheritance [15]. The idea of fitness imitation has been extended and more sophisticated estimation methods have been developed in [38]. A similarity based model is introduced in [39] and is applied to constrained and unconstrained optimization problems.

In multi-objective optimization problems (MOOP), the complexity of the problem is normally higher, compared to that of single-objective optimization problems (SOOP) [40]. In general, although the fitness approximation approaches used in SOOP can be simply extended and adapted for MOOP, such adaptation may require more elaborate mechanisms. One example of this is constraint-handling.[3] It is well-known that in real-world optimization problems there are normally constraints of different types (e.g., related to the geometry of structural elements to completion times, etc.) that must be satisfied for a solution to be acceptable. Traditionally, penalty functions have been used with EAs to handle constraints in SOOP [43]. However, because of the several problems associated to penalty functions (e.g., the definition of appropriate penalty values is normally a difficult task that has a serious impact on the performance of the EA), some researchers have proposed alternative constraint-handling approaches that require less critical parameters and perform well across a variety of problems (see for example [41, 44, 43]). However, when dealing with MOOPs, many of these constraint-handling techniques cannot be used in a straightforward manner, since in this case, the best trade-offs among the objectives are always located in the boundary between the feasible and the feasible region. This requires the development of different approaches specially tailored for MOOPs (see for example [45, 46]).

[3] Although constraint-handling techniques are very important in real-world optimization problems, their discussion is beyond the scope of this chapter, due to space limitations. Interested readers are referred to other references for more information on this topic (see for example [41, 42]).

A similar problem occurs when attempting to migrate single-objective fitness approximation models to MOOPs. For more details on this topic, see [47].

While the above methods aim to reduce computational cost by approximating the fitness function, the prevalent problems with interpolation in rough surfaces remains. If the assumption of smooth continuity is not valid, interpolation may even yield values that are not physically realizable. Furthermore, we may be blinded to the optimal solutions using interpolation as interpolation assumes a pattern of behavior that may not be valid around optimal peaks. The next section addresses this problem by introducing the concept of information granulation.

5 Adaptive Fuzzy Fitness Granulation

Fuzzy granulation of information is a vehicle for handling information, as well as a lack of it (uncertainty), at a level of coarseness that can solve problems appropriately and efficiently [48]. In 1979, the concept of fuzzy information granulation was proposed by Zadeh [49] as a technique by which a class of points (objects) are partitioned into granules, with a granule being a clump of objects drawn together by indistinguishability, similarity, or functionality. The fuzziness of granules and their attributes is characteristic of the ways by which human concepts and reasoning are formed, organized and manipulated. The concept of a granule is more general than that of a cluster, potentially giving rise to several conceptual structures in various fields of science as well as mathematics.

In this chapter, with a view to reducing computational cost, the concept of fitness granulation is applied to exploit the natural tolerance of EAs in fitness function computations. Nature's *survival of the fittest* is not about exact measures of fitness; rather it is about rankings among competing peers. By exploiting this natural tolerance for imprecision, optimization performance can be preserved by computing fitness only selectively and only to keep this ranking among individuals in a given population. Also, fitness is not interpolated or estimated; rather, the similarity and indistinguishability among real solutions is exploited.

In the proposed algorithm, an adaptive pool of solutions (fuzzy granules) with an exactly computed fitness function is maintained. If a new individual is sufficiently similar to a known fuzzy granule [49], then that granules' fitness is used instead as a crude estimate. Otherwise, that individual is added to the pool as a new fuzzy granule. In this fashion, regardless of the competitions' outcome, the fitness of the new individual is always a physically realizable one, even if it is a *crude* estimate and not an exact measurement. The pool size as well as each granules' radius of influence is adaptive and will grow/shrink depending on the utility of each granule and the overall population fitness. To encourage fewer function evaluations, each granule's radius of influence

is initially large and gradually shrinks at later stages of the evolutionary process. This encourages more exact fitness evaluations when competition is fierce among more similar and converging solutions. Furthermore, to prevent the pool from growing too large, not used granules are gradually replaced by new granules, once the pool reaches a certain maturity.

5.1 Algorithm Structure

Given the general overview in the preceding section, the concrete steps of the algorithm are as follows:

Step 1: Create a random parent population $P_1 = \{X_1^1,\ X_2^1,\ \ldots,\ X_j^1,\ \ldots,\ X_t^1\}$ of design variable vector, where, more generally, $X_j^i = \{x_{j,1}^i,\ x_{j,2}^i,\ \ldots\ , x_{j,r}^i,\ \ldots,\ x_{j,m}^i\}$ is the jth parameter individual in the ith generation, $x_{j,r}^i$ the rth parameter of X_j^i, m is the number of design variables and t is the population size.

Step 2: Define a multi-set G of fuzzy granules $(C_k,\ \sigma_k,\ L_k)$ according to $G = \{(C_k,\ \sigma_k,\ L_k) | C_k \in \Re^m,\ \sigma_k \in \Re,\ L_k \in \Re,\ k = 1, \ldots, l\}$. G is initially empty (i.e., $l = 0$). C_k is an m-dimensional vector of centers, σ_k is the width of membership functions (WMFs) of the kth fuzzy granule, and L_k is the granule's life index. A number of granules with different widths are shown in Figure 2.

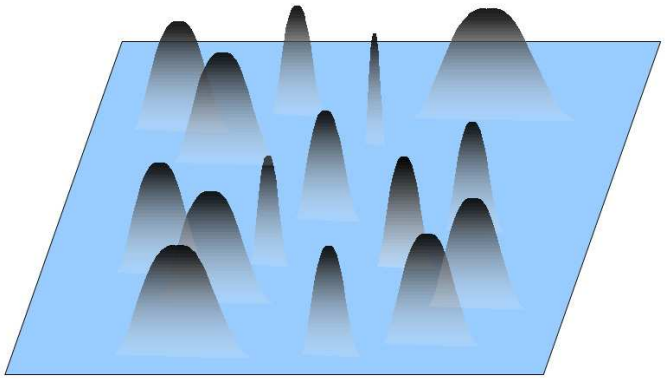

Fig. 2: A number of gaussian granules with different widths in a 2-D solution space. Once a new individual is sufficiently similar to a granule in the granule pool, then that granules' fitness is used instead as a crude estimate. Otherwise, that individual is added to the pool as a new fuzzy granule. Each granules' radius of influence is determined based on equation (4).

Step 3: Choose the phenotype of first chromosome ($X_1^1 = \{x_{1,1}^1,\ x_{1,2}^1,\ \ldots\ , x_{1,r}^1,\ \ldots,\ x_{1,m}^1\}$) as the center of the first granule ($C_1 = \{c_{1,1},\ c_{1,2},\ \ldots\ , c_{1,r},\ \ldots,\ c_{1,m}\} = X_1^1$).

Step 4: Define the membership $\mu_{k,r}$ of each $x_{j,r}^i$ to each granule member by a Gaussian similarity neighborhood function according to

$$\mu_{k,r}\left(x_{j,r}^i\right) = \exp\left(\frac{-\left(x_{j,r}^i - c_{k,r}\right)^2}{\left(\sigma_k\right)^2}\right),\ k = 1, 2, \ldots, l\ , \tag{3}$$

where l is the number of fuzzy granules.

Remark: σ_k is the distance measurement parameter that controls the degree of similarity between two individuals. Like in [50], σ_k is defined based on equation (4). According to this definition, the granules shrink or enlarge in reverse proportion to their fitness:

$$\sigma_k = \gamma \frac{1}{\left(e^{F(C_k)}\right)^\beta}\ , \tag{4}$$

where $\beta > 0$ is an emphasis operator and γ is a proportionality constant. The problem arising here is how to determine the parameters β and γ as design parameters. The fact is that these two parameters are problem dependent and, in practice, a number of trials is needed to adjust these parameters. This trial is based on a simple rule with respect to the acceleration of the parameter optimization procedure: high speed needs to have enlargement in the granule spread and, as a consequence of this, less accuracy is obtained in the fitness approximation, and viceversa. To deal with this rule, a fuzzy controller is presented in [50].

Step 5: Compute the average similarity of every new design parameter $X_j^i = \{x_{j,1}^i,\ x_{j,2}^i,\ \ldots\ , x_{j,r}^i,\ \ldots,\ x_{j,m}^i\}$ to each granule G_k using equation (5)

$$\overline{\mu}_{j,k} = \frac{\sum_{r=1}^{m} \mu_{k,r}\left(x_{j,r}^i\right)}{m} \tag{5}$$

Step 6: Either calculate the exact fitness function of X_j^i or estimate the fitness function value by associating it to one of the granules in the pool in case there is a granule in the pool with higher similarity to X_j^i than a predefined threshold, i.e.

$$f\left(X_j^i\right) = \begin{cases} f\left(C_k\right) & \text{if } \max\limits_{k\in\{1,2,\ldots,l\}} \{\overline{\mu}_{j,k}\} > \theta^i\ , \\ f\left(X_j^i\right) & \text{otherwise.} \end{cases} \tag{6}$$

where $f(C_x)$ is the fitness function value of the fuzzy granule, $f(X_j^i)$ is the real fitness calculation of the individual, $\theta^i = \alpha(\max\{f(X_1^{i-1}), f(X_2^{i-1}), \ldots, f(X_t^{i-1})\}/\overline{f}^{i-1})$, $K = \text{argmax}\{\overline{\mu}_{j,k}\}$ when $k \in \{1, 2, \ldots, l\}$, $\overline{f}^i = \sum_{j=1}^{i} f(X_j^i)/t$

and $\alpha > 0$ is a proportionality constant that is usually set at 0.9 unless otherwise indicated. The threshold θ^i increases as the best individual's fitness at generation i increases. As the population matures and reaches higher fitness values (i.e., while converging more), the algorithm becomes more selective and uses exact fitness calculations more often. Therefore, with this technique we can utilize the previous computational efforts during previous generations. Alternatively, if

$$\max_{k\in\{1,2,\ldots,l\}}\{\overline{\mu}_{j,k}\} < \theta^i$$

X_j^i is chosen as a newly created granule.

Step 7: If the population size is not completed, repeat **Steps 5 to 7**.

Step 8: Select parents using a suitable selection operator and apply the genetic operators of recombination and mutation to create a new generation.

Step 9: When termination/evolution control criteria are not met, then update σ_k using equation (4) and repeat **Steps 5 to 9**.

In [48] and [51], additional details on the convergence speed of the algorithm on a series of mathematical testbeds are provided along with a simple example to illustrate the competitive granule pool update.

5.2 How to control the length of the granule pool?

As the evolutionary procedures are applied, it is inevitable that new granules are generated and added to the pool. Depending on the complexity of the problem, the size of this pool can be excessive and become a computational burden itself. To prevent such unnecessary computational effort, a *forgetting factor* is introduced in order to appropriately decrease the size of the pool. In other words, it is better to remove granules that do not win new individuals, thereby producing a bias against individuals that have low fitness and were likely produced by a failed mutation attempt. Hence, L_k is initially set to N and subsequently updated as below,

$$L_k = \begin{cases} L_k + M & \text{if } k = K\ , \\ L_k & otherwise\ , \end{cases} \tag{7}$$

where M is the life reward of the granule and K is the index of the winning granule for each individual at generation i. At each table update, only the N_G granules with the highest L_k index are kept, and the others are discarded. In [52], an example has been provided to illustrate the competitive granule pool update law. Adding a new granule to the granule pool and assigning a life index to it, is a simple way of controlling the size of the granule pool, since the granules with the lowest life index will be removed from the pool. However, it may happen that the new granule is removed, even though it was just inserted into the pool. In order to prevent this, the pool is split into

two parts with sizes εN_G and $(1-\varepsilon)N_G$. The first part is a FIFO (First In, First Out) queue and new granules are added to this part. If it grows above εN_G, then the top of the queue is moved to the other part. Removal from the pool takes place only in the $(1-\varepsilon)N_G$ part. In this way, new granules have a good chance to survive a number of steps. In all of the simulations that are conducted here, ε is set at 0.1.

The distance measurement parameter is completely influenced by the granule enlargement/shrinkage in the widths of the produced membership functions. As in [52], the combined effect of granule enlargement/shrinkage is in accordance with the granule fitness and it requires the fine-tuning of two parameters, namely β and γ. These parameters are problem dependent and it seems critical to have a procedure to deal with this difficulty. In [50] and [53], an auto-tuning strategy for determining the width of membership functions is presented which removes the need of exact parameter determination, without a negative influence on the convergence speed.

6 Numerical results

To illustrate the efficacy of the proposed granulation algorithm, the result of applying it to the problems introduced in Section 2 are studied and analyzed in the two following sections. The commercial FEA software ANSYS [54] is used during the analysis and numerical simulation study.

The GA routines utilize initially random populations, binary-coded chromosomes, single-point crossover for the first three problems and 15-point crossover for the piezoelectric actuator design problem, mutation, fitness scaling, and an elitist stochastic universal sampling selection strategy. Crossover rate $P_{XOVER} = 1$, $P_{MUTATION} = 0.01$ and the population size is set at 20. However, due to the number of parameters and complexity of the structural problems, the number of generations is set to 50 for the first three problems and 600 for the piezoelectric actuator design problem. These settings were determined during several trial runs to reflect the best performing set of parameters for the GA. Finally, the chromosome length varies depending on the number of variables in a given problem but each variable is still allocated 8 bits.

For performing the FES, a fitness and associated reliability value are assigned to each new individual that is truly evaluated if the reliability value is below a certain threshold T. The reliability value varies between 0 and 1 and depends on two factors: first is the reliability of parents, and second is how close parents and children are in the solution space, as explained in equation (2). Also, as mentioned in [18], $T = 0.7$ is used for the threshold as we empirically found that it generally produces the best results. The parameters of the GA-ANN are the same as in the GA alone. In the GA-ANN approach

for solving optimization problems, a two-layer neural network is used, having as input the design variables and as outputs the fitness values.

Furthermore, due to the stochastic nature of EAs, each of the simulations was repeated ten times, and a paired Mann-Whitney U test (also called the MannWhitneyWilcoxon) was performed except for the last optimization problem in which, for each algorithm, it was performed only once, due to the running time needed. The significance level α represents the maximum tolerable risk of incorrectly rejecting the null hypothesis H_0, indicating that the mean of the 1^{st} population is not significantly different from the mean of the 2^{nd} population. The p-value or the observed significance level of a statistical test is the smallest value of α for which H_0 can be rejected. If the p-value is less than the pre-assigned significance level α, then the null hypothesis is rejected. Here, the significance level α was assigned, and the p-value was calculated for each of the following applications.

The results are presented in Tables 1, 2, 3 and 5, in which *FFE* stands for the number of fitness function evaluations needed to perform the optimization task and the *training data* column presents the number of initial input/output pairs needed in order to build up the approximation model. Since the most computationally expensive part of an evolutionary algorithm is usually, by far, its fitness evaluation, the convergence *time improvement* of different algorithms, compared to the standard GA, can be estimated in terms of the number of fitness evaluations. So, the *time improvement* percentage column is calculated as one minus the difference between the sum of *FFE* and *training data* divided by the number of FFE of the standard algorithm, i.e., a GA, multiplied by 100.

6.1 3-Layer composite beam

A 3-layer composite beam has been modeled numerically by using the ANSYS program. The composite layout are the design variables that change in the region [0 - 180]. The objective here is to raise the first natural frequency by appropriately choosing 2 composite layers' angles. In this example, the Young's modulus [55] is EX = 210 GPa, EY = 25 GPa, EZ = 25 GPa, GXY = GYZ = GXZ = 30 GPa, Poisson's ratio $\nu = 0.2$ and density $\rho = 2100$ kg/m^3. There are two design variables (two degrees of freedom) for this optimization problem each varying between 0 and 180. For this case, a 2-100-1 ANN architecture is consequently chosen and used for the optimization runs. The proposed algorithm (called AFFG, for adaptive fuzzy fitness granulation) and other methods are compared in Table 1. Results indicate that while there is not a significant statistical difference between the three algorithms in terms of solution fitness, i.e., rigidity of the beam, the time savings provided by the proposed method is much higher than that of the GA-ANN. In particular, the proposed AFFG algorithm finds better solutions on the average with less

computational time as compared with the GA-ANN. Also, while FES seems to have found better solutions, the proposed GA-AFFG used less than half as many evaluations.

	FFEs	Training data	Time improvement (%)	Optimum S^{-1}	p-value
GA	1000	Not Needed		19.3722	
FES	228.1	Not Needed	77.19	19.369	0.0211
GA-ANN	155.9	100	74.41	19.3551	0.0026
GA-AFFG	97.5	Not Needed	90.25	19.3681	0.0355

Table 1: Performance of the optimization methods (average of 10 runs) for the 3-layer composite beam, $\alpha = 0.9$, $\beta = 0.1$, $\gamma = 30$, $M = 5$, $N_G = 250$, $T = 0.7$.

6.2 Airplane wing

Figure 3(a) shows the initial design of an airplane wing. The wing is of uniform configuration along its length, and its cross-sectional area is defined to be a straight line and a spline. It is held fixed to the body of the airplane at one end and hangs up freely at the other. The objective here is to maximize the wing's *first natural frequency* by appropriately choosing three key points of the spline. The material properties are: Young's modulus = 261.820 GPa, density $\rho = 11031$ kg/m^3, Poisson's ratio $\nu = 0.3$.

The optimized shape found by a simple GA is shown in Figure 3(b) and that found by GA-AFFG is shown in Figure 3(c). A 6-100-1 architecture is chosen for the ANN used as fitness approximator. Table 2 illustrates that while the GA-ANN finds inferior solutions as compared with the GA, the use of the ANN significantly reduces computational time. The application of AFFG shows an improvement in the search quality while maintaining a low computational cost. Specifically, the average 10-run performance of the AFFG solutions is higher than that of any of the competing algorithms including the GA, FES and GA-ANN. Furthermore, while the Mann-Whitney U test confirms that the proposed algorithm solutions are at least as good as those produced by the GA, the proposed algorithm is over 82% faster.

6.3 2D truss frame

A typical truss designed by an engineer is illustrated in Figure 4(a). The objective (fitness) here is to raise the structure's *first natural frequency* to

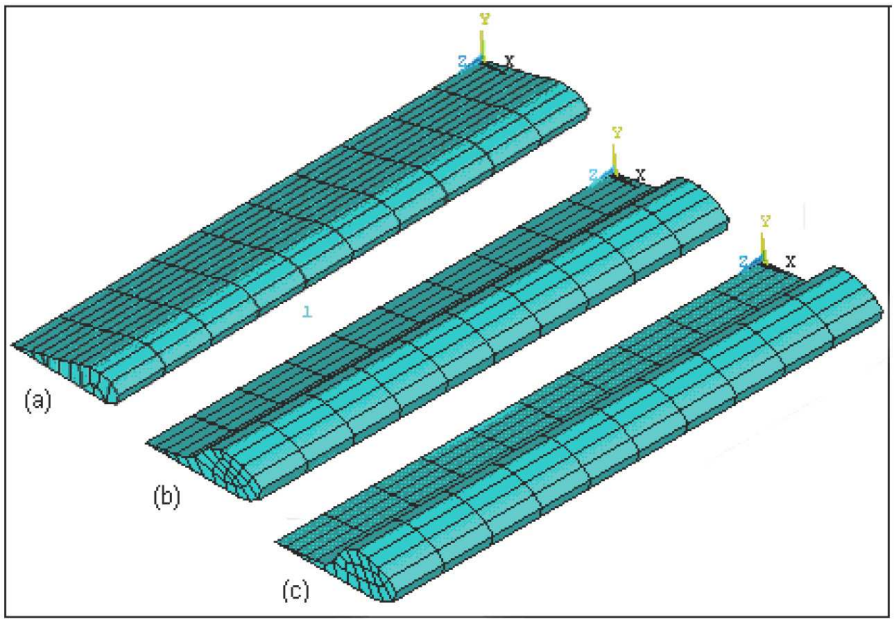

Fig. 3: Airplane wing: (a) initial shape, (b) GA optimized shape, and (c) GA-AFFG.

	FFEs	Training data	Time improvement (%)	Optimum S^{-1}	p-value
GA	1000			6.0006	
FES	481.6		51.84	5.9801	0.9698
GA-ANN	172.1	100	72.79	5.9386	0.4274
GA-AFFG	173.5		82.65	6.0527	0.3075

Table 2: Performance of the optimization methods (average of 10 runs) for Airplane wing, $\alpha = 0.9$, $\beta = 0.5$, $\gamma = 1$, $M = 5$, $N_G = 250$, $T = 0.7$.

reduce the vibration domain and to prevent the resonance phenomenon (in dynamic response) of the structure by appropriately choosing the 18 key point locations (our design variables) as illustrated in Figure 3(a).

In this benchmark, isotropic material properties are assumed (Young's modulus E = 210 GPa, Poisson's ratio $\nu = 0.3$ and density $\rho = 7800$ kg/m^3). The optimized shapes produced by the GA and the new proposed method AFFG are shown in Figures 4(b) and 4(c), respectively. The 36-100-1 ANN architecture is chosen and used for the optimization runs.

The search begins with an initial population. The maximum fitness in the initial population is nearly 9.32. Over several generations, the fitness gradually evolves to a higher value of 11.902. Figure 5 shows a plot of best,

average and worst fitness vs. generation number for one run of our GA-AFFG. This performance curve shows the learning activity or adaptation associated with the algorithm. The total number of generations is 50. For a population size of twenty, this requires 1000 (50×20) fitness evaluations for the GA while the proposed GA-AFFG required only 570.4 fitness evaluations. Figure 6 shows the plot of the number of FEA evaluations vs. generation number corresponding to one run [48].

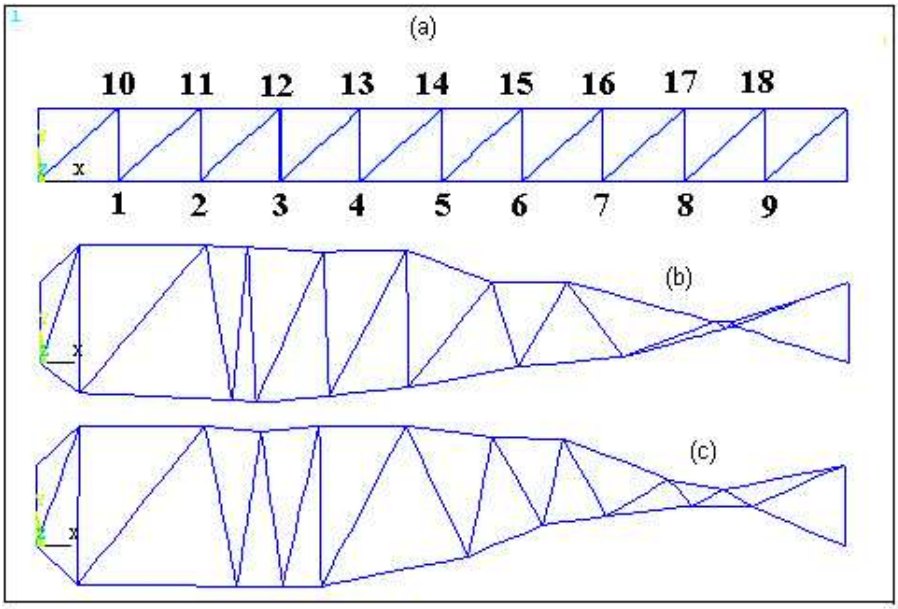

Fig. 4: 2D truss frame: (a) initial configuration, (b) GA optimized shape, and (c) GA-AFFG optimized shape.

	FFEs	Training data	Time improvement (%)	Optimum S^{-1}	p-value
GA	1000			12.1052	
FES	1000		0	11.8726	0.0058
GA-ANN	293	100	60.66	11.8697	0.0257
GA-AFFG	570.4		42.96	12.1160	0.9097

Table 3: Performance of the optimization methods (average of 10 runs) for the 2D truss, $\alpha = 0.9$, $\beta = 0.11$, $\gamma = 3.05$, $M = 5$, $N_G = 550$, $T = 0.7$.

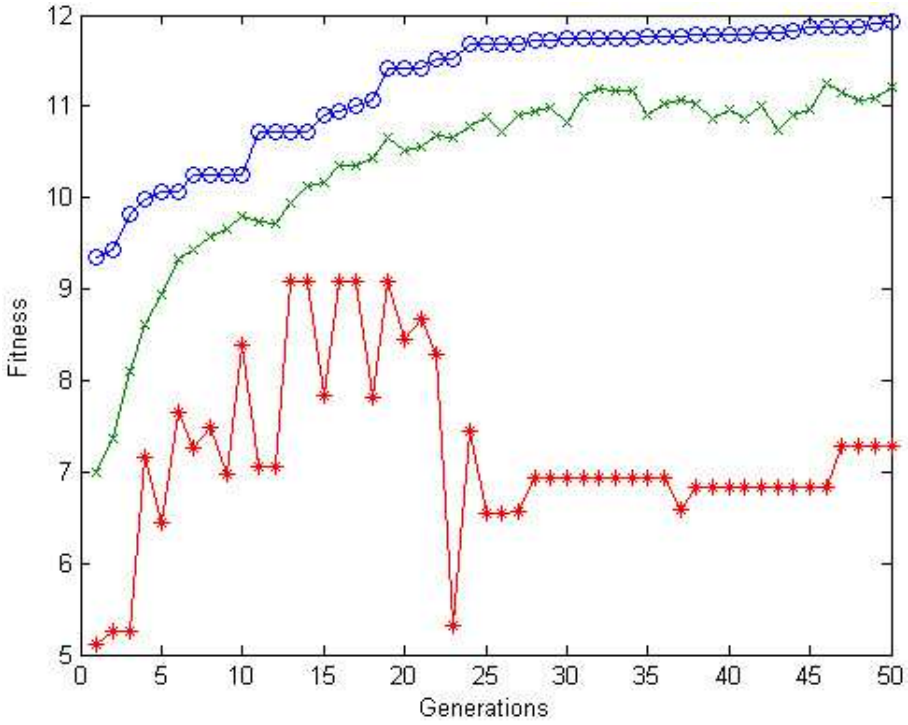

Fig. 5: Plot of generation number vs. fitness value for the 2D truss frame using GA-AFFG: best (circle), average (cross) and worst (asterisk) individuals at each generation.

6.4 Voltage and pattern design of piezoelectric actuator

Piezoelectric materials have coupled mechanical and electrical properties making them able to generate a voltage when subjected to a force or deformation (this is termed as the direct piezoelectric effect). Conversely, they exhibit mechanical deformation when subjected to an applied electric field (this is called the converse piezoelectric effect) [51]. Various applications of piezoelectric actuators/sensors have appeared in the literature. Lin et al. [56] investigated the modeling and vibration control of a smart beam by using piezoelectric damping-modal actuators/sensors. They presented theoretical formulations based on damping-modal actuators/sensors and numerical solutions for the analysis of a laminated composite beam with integrated sensors and actuators. A proof-of-concept design of an inchworm-type piezoelectric actuator of large displacement and force for shape and vibration control of adaptive truss structures is proposed by Li et al. in [57]. The applications of

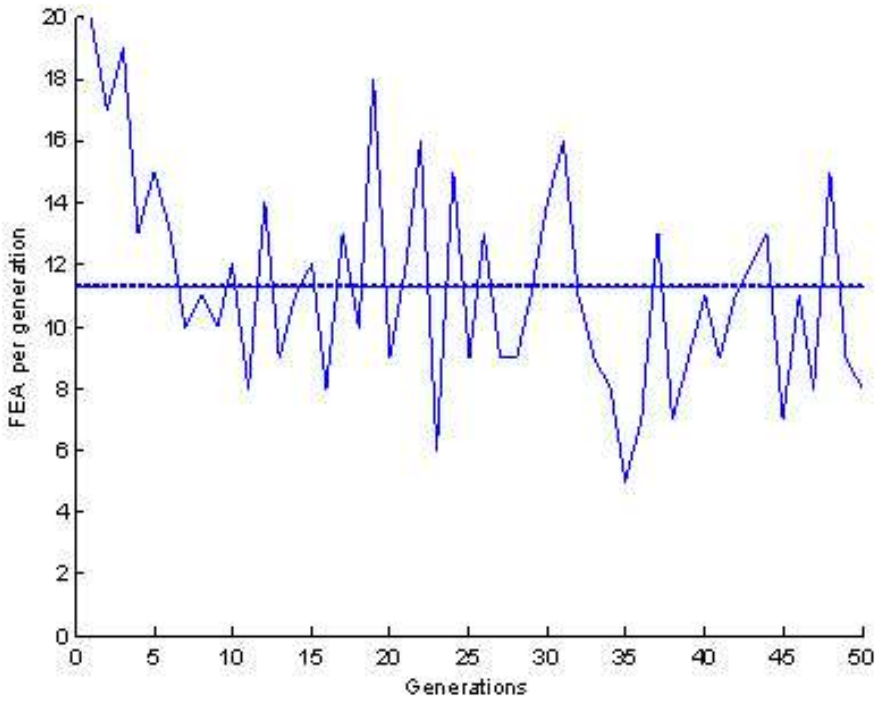

Fig. 6: Plot of the generation number vs. number of FEA evaluations for the 2D truss frame in a single run using GA-AFFG.

such actuators include smart or adaptive structural systems for the car and aerospace industries.

A fiber composite plate with initial imperfections and under in-plane compressive loads is studied by Adali et al. [58] with a view towards minimizing its deflection and optimizing its stacking sequence by means of the piezoelectric actuators and the fiber orientations. Krommer [59] studied a method to control the deformation of a sub-section of a beam. His intention was to apply a distributed control by means of self-stresses within the sub-section to keep the sub-section in its non-deformed state. In practical applications such as deformation control of conformal antennas, this strategy is highly valuable.

Global optimization algorithms [60] along with a finite element formulation are widely used in shape control. For instance in [10], a computational intelligence based optimization algorithm along with a modified finite element formulation is used to deal with the shape control of functionally graded material (FGM) plates that contain piezoelectric sensor and actuator patches. In this study, an optimal voltage distribution or a gain control matrix are used as design variables for the shape control of smart structures. Numerical simulations have been successfully carried out on the shape control of the

FGM plates by optimizing the voltage distribution for the open loop shape control or gain values for the closed loop shape control. Finite element formulation with non-rectangular shaped actuators for laminated smart composite structure is studied in [61]. For smart cantilever plates, the actuated deflections are measured and are used to validate the present formulation. They also investigated the effect of actuator pattern on the optimum values of the applied voltages and the shape match factors. Numerical results shown that the actuator patterns may have an important influence on the values of the optimum voltages applied to each individual actuator and the final shape match factor.

6.4.1 Piezoelectric equations (constitutive equations)

In this study, the assumption is that the thermal effect is negligible. The piezoelectric constitutive relationships describe how two piezoelectric mechanical and electrical quantities (stress, strain, electric displacement, and electric field) interact and it is expressed by the direct and the converse piezoelectric equations respectively [62]:

$$\{D\} = [e]\{\varepsilon\} + [\varepsilon]\{E\} \,, \tag{8}$$

$$\{\sigma\} = [Q]\{\varepsilon\} + [e]^T\{E\} \,, \tag{9}$$

where $\{\sigma\}$ is the stress vector, $[Q]$ is the elastic stiffness matrix, $\{\varepsilon\}$ is the strain vector, $[e]$ is the piezoelectric constant matrix, $\{E\} = -\nabla\varphi$ is the electric field vector. Also, φ is the electrical potential, $\{D\}$ is the electric displacement vector and $[\varepsilon]$ is the permittivity coefficient matrix. Equations (8) and (9) describe the electromechanical coupling. Assuming that a laminated beam consists of a number of layers and each layer possesses a plane of material symmetrically parallel to the x-y plane, the constitutive equations for the k^{th} layer can be written as [63]:

$$\begin{Bmatrix} D_1 \\ D_3 \end{Bmatrix}_k = \begin{bmatrix} 0 & e_{15} \\ e_{31} & 0 \end{bmatrix}_k \times \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_5 \end{Bmatrix}_k + \begin{bmatrix} \varepsilon_{11} & 0 \\ 0 & \varepsilon_{33} \end{bmatrix}_k \times \begin{Bmatrix} E_1 \\ E_3 \end{Bmatrix}_k \tag{10}$$

$$\begin{Bmatrix} \varepsilon_1 \\ \varepsilon_3 \end{Bmatrix}_k = \begin{bmatrix} Q_{11} & 0 \\ 0 & Q_{55} \end{bmatrix}_k \times \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_5 \end{Bmatrix}_k + \begin{bmatrix} 0 & \varepsilon_{31} \\ \varepsilon_{15} & 0 \end{bmatrix}_k \times \begin{Bmatrix} E_1 \\ E_3 \end{Bmatrix}_k \tag{11}$$

where

$$Q_{11} = \frac{E_{11}}{1 - v_{12}v_{21}}, Q_{55} = G_{13}$$

and are the reduced elastic constants of the k^{th} layer, E_{11} is the Young's modulus and G_{13} is the shear modulus. The piezoelectric constant matrix $[e]$ can be expressed in terms of the piezoelectric strain $[d]$ as:

$$[e] = [d][Q]$$

where

$$[d] = \begin{bmatrix} 0 & d_{15} \\ d_{31} & 0 \end{bmatrix}$$

Using the above piezoelectricity analysis and formulation, finite element model (FEM) of piezoelectric patches and metal plate [64] was built by ANSYS [54]. Also, a small deflection and thin plate theory are assumed for the FEM of the plate.

To validate the software, a clamped free aluminum plate with 4 piezoelectric patches is modeled and the results are compared with the experimental model of reference [65]. A close agreement between our model and our experimental results is observed. Also, in order to achieve an acceptable mesh density, mesh sensitivity [4] is performed.

6.4.2 Piezoelectric design for static shape control

The shape control problem considered here is to find the optimal actuator pattern design vector P and exiting voltage vector V as design variables. The (quasi-) static shape control problem can be defined, in the context of an optimization formulation, as follows:

Find $S = [P, V]^T$ to minimize:

$$f(S) = \sum_{j=1}^{N_x} \sum_{i=1}^{N_y} \frac{\left| d_{i,j}^d - d_{i,j}^f \right|}{\left| max\left(d_{i,j}^d\right) \right|} / \left(N_x \times N_y\right) \tag{12}$$

S is the design variable vector with two components: i) the pattern variable vector P, and ii) the applied voltage variable vector V. Here, $f(S)$ is the objective function. P is the distribution of active piezoelectric actuator material (pattern variable) whereas the voltage variables in vector V are the electrical potentials applied across the thickness direction of each actuator. The objective function $f(S)$ in equation (12) is a weighted sum of all the absolute differences between the desired and designed shapes at all nodes. $d_{i,j}^d$ and $d_{i,j}^f$ are the desired and designed (calculated by the FE model) displacements of the i^{th} location, respectively, and $max(d_i^d)$ is the maximum displacement in the desired structural shape. Plates are considered to be in the (x, y) Cartesian plane, d_i^d and d_i^f are the transversal displacements or z in Cartesian plane which is the displacement component in the global displacement vector. As the displacement is small here, there is no need to consider stress or strain constraint variables for the shape control problem.

[4] Mesh sensitivity is performed to reduce the number of elements and nodes in the mesh while ensuring the accuracy of the finite element solution [66].

6.4.3 Model description

A cantilever plate clamped at its left edge and subjected to a non-applied mechanical load is assumed here. The plate has a length of 154 mm; width of 48 mm and consists of one layer of 0.5 mm in thickness. The piezoelectric actuators (thickness of 0.3 mm each) are attached to the top surfaces of the plate (Figure 7). The desired pre-defined surface [65] is defined as:

$$d^d_{i,j} = \left(1.91x^2 + 0.88xy + 0.19x\right) \times 10^{-4} \,. \tag{13}$$

The piezoelectric electro-mechanical properties shown in Table 4 according to PX5-N from Philips Components. After a careful mesh sensitivity analysis, a FEM is built as illustrated in Figure 8.

$C^E_{11}(N\ m^{-2})$	13.11×10^{10}	$d_{15}(m\ V^{-1})$	515×10^{-12}
$C^E_{12}(N\ m^{-2})$	7.984×10^{10}	$d_{31}(m\ V^{-1})$	-215×10^{-12}
$C^E_{13}(N\ m^{-2})$	8.439×10^{10}	$d_{33}(m\ V^{-1})$	500×10^{-12}
$C^E_{33}(N\ m^{-2})$	12.31×10^{10}	$\varepsilon^t_{11}/\varepsilon_0$	1800
$C^E_{44}(N\ m^{-2})$	2.564×10^{10}	$\varepsilon^t_{33}/\varepsilon_0$	2100
$C^E_{66}(N\ m^{-2})$	2.564×10^{10}	$\rho(kg\ m^{-3})$	7800

Table 4: Material properties for the PX5-N piezoelectric material [65].

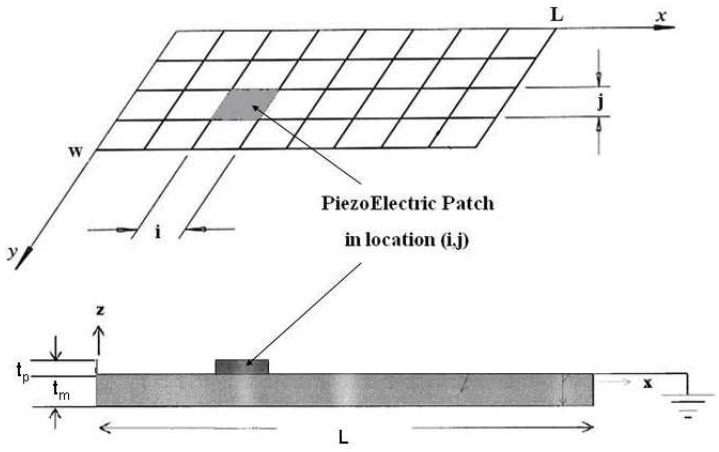

Fig. 7: Geometrical model of the piezoelectric patch adopted here.

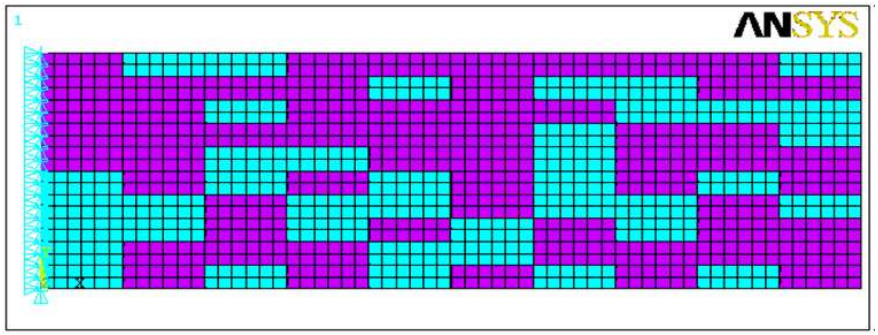

Fig. 8: Finite element model built by ANSYS.

For this SD and optimization problem, there are 200 design variables. Half of these design variables belong to actuation voltage of piezoelectric patches which vary between -10 and 20 V and the rest of the design variables are Boolean, indicating whether or not the voltage should be applied to the piezoelectric patches. When the $i^{th}(i = 1, \ldots, 100)$ piezoelectric pattern variable is zero, the piezoelectric patch is not built so that there is no actuation voltage, and viceversa. Figure 9 shows the graph of best, average and worst fitness vs. generation number and Figure 10 shows the number of FEA evaluations vs. generation number for a single GA-AFFG run while Table 4 presents the results of the four optimization algorithms corresponding to one run.

	FFEs	Training data	Time Improved (%)	Error (%)
GA	12000			7.313
FES	12000		0	12.82
GA-ANN	2617	5000	36.52	8.093
GA-AFFG	5066	Not needed	57.64	7.141

Table 5: Piezoelectric actuator performance of the optimization methods, α = 0.9, $\beta = 0.11$, $\gamma = 3.05$, $M = 5$, $N_G = 550$, $T = 0.7$.

7 Analysis of results

Tables 1, 2, 3 and 5, illustrate the performance of the proposed GA-AFFG method in comparison with a GA, FES and GA-ANN [48] in terms of computational efficiency and performance for the 3-layer composite beam, the airplane wing, and the 2D truss design problems as well as for the piezo-

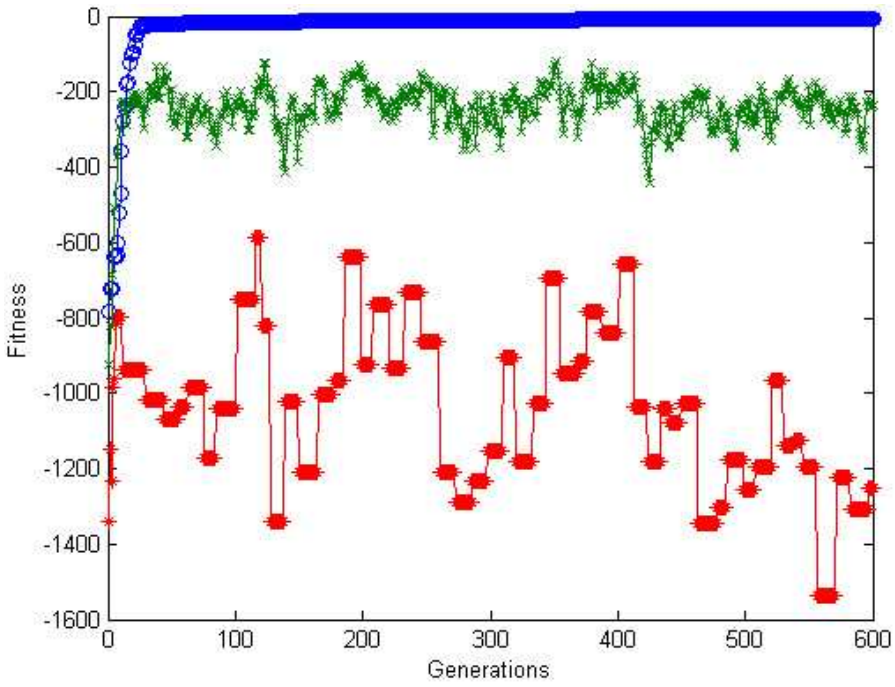

Fig. 9: Generation number vs. fitness for the piezoelectric actuator using our proposed GA-AFFG for a single run: best (circle), average (cross) and worst (asterisk) of individuals at each generation.

electric actuator problem. Due to the stochastic nature of the GA, the first three design simulations are repeated 10 times and a statistical analysis is performed. However, for the piezoelectic actuator we could not run the GA that many times, because of its high computational cost.

The second column in these tables makes a comparison of the three algorithms in terms of the number of FEA evaluations as compared to those of the GA, while the fourth column makes a comparison in terms of performance. Results indicate that our proposed GA-AFFG finds statistically equivalent solutions by using more than 90%, 82%, 42% and 57% fewer finite element evaluations. The GA-ANN also significantly reduces the number of FEA evaluations, but its average performance is inferior when compared with our proposed GA-AFFG due to the ANNs approximation error. It must be noted that the GA-ANN's improved time includes the number of initial training data.

For the piezoelectric actuator design problem, Table 5 illustrates a comparison of the GA, FES and GA-ANN [51] with respect to our proposed GA-AFFG in terms of computational efficiency and performance. The second column of this table makes a comparison of the four algorithms in terms

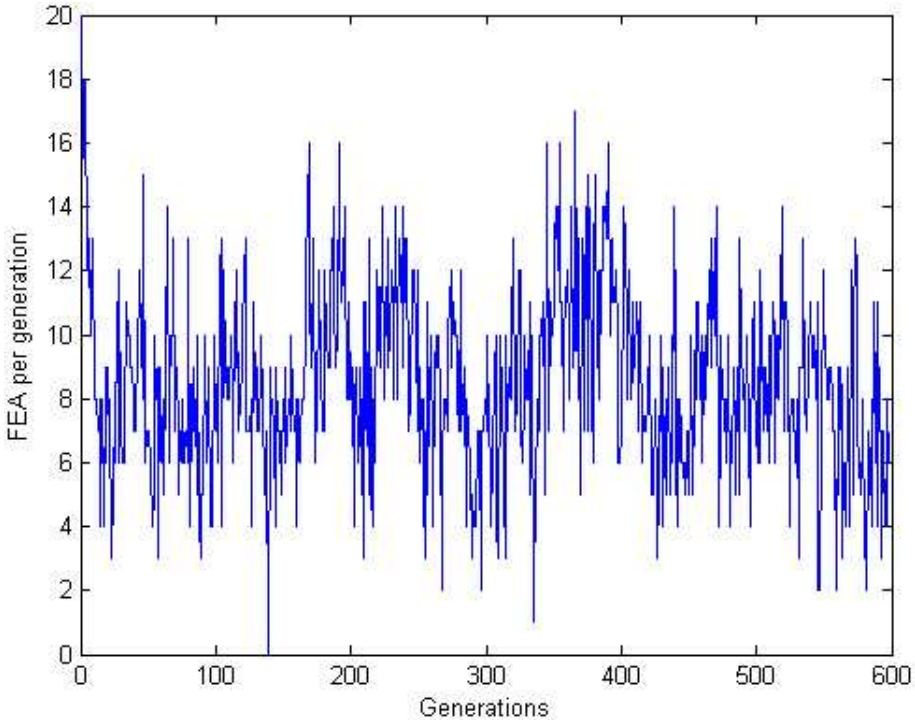

Fig. 10: Generation number vs. number of FEA evaluations, for the piezoelectric actuator, using our proposed GA-AFFG for a single run.

of the number of FEA evaluations as compared with a GA, while the fifth column makes a comparison in terms of the quality of the optimal solutions. Results indicate that GA-AFFG finds at least equivalent solutions by using 57% fewer finite element evaluations as compared to GA. Also, when compared with the GA-ANN, the proposed algorithm finds better solutions while being more computationally efficient. The main difference here is ANN's need for pre-training. Trying various sizes of initial training sets and considering the 200 design parameters, the ANN required at least 5000 training data pairs for comparable performance, see Table 5.

Overall, when compared with a GA, the two sets of applications indicate that FES, GA-ANN and GA-AFFG improve the computational efficiency of their problem by reducing the number of exact fitness function evaluations. However, the neuro-approximation as well as fitness inheritance fail with a growing size of the input-output space. Consequently, the utility of AFFG becomes more significant in larger and more complex design problems. Furthermore, our statistical analysis confirms that fitness inheritance is more comparable in terms of performance when the size of the search space is

smaller (Tables 1 and 2), but its performance deteriorates as the complexity of the problem increases (Tables 3 and 5).

A comparison of the number of exact fitness function evaluations in terms of mean and variance that presents the improved computational time is presented in Tables 6, 7 and 8 for the first three mechanical optimization problems described before. A Mann-Whitney U test is also performed to study the significance of lower computation cost. Since the fourth optimization problem (piezoelectic actuator design) could not be repeated due to the its FEA time consuming nature, a Mann-Whitney U test could not be performed in that case.

	Simulation results		
3-layer composite beam	Mean	Var	p-Value
FES	228.1	4601.2	6.39×10^{-05}
GA-ANN	155.9	511.9	6.34×10^{-05}
GA-AFFG	97.5	406.7	6.39×10^{-05}

Table 6: A Mann-Whitney U test of the number of real fitness calculations for the 3-layer composite beam (10 runs).

	Simulation results		
Airplane wing	Mean	Var	p-Value
FES	481.6	38648	6.39×10^{-05}
GA-ANN	172.1	6392.1	6.39×10^{-05}
GA-AFFG	173.5	1600.3	6.39×10^{-05}

Table 7: A Mann-Whitney U test of the number of real fitness calculations for the airplane wing (10 runs).

	Simulation results		
2D truss	Mean	Var	p-Value
FES	100	0	Not available
GA-ANN	293	2394.2	6.39×10^{-05}
GA-AFFG	570.4	18477	6.39×10^{-05}

Table 8: A Mann-Whitney U test of the number of real fitness calculations for the 2D truss (10 runs).

8 Conclusions

In this chapter, we have proposed a systematic and robust methodology for solving complex structural design and optimization problems. The proposed methodology relies on the use of finite element analysis and adaptive fuzzy fitness granulation. As we saw, adaptive fuzzy fitness granulation provides a method to selectively reduce the number of actual fitness function evaluations performed by considering the similarity/indistinguishability of an individual to a pool of fuzzy information granules. Since the proposed approach does not use approximation or online training, it is not caught in the pitfalls of such techniques such as false peaks, large approximation error due to extrapolation, and time consuming online training.

The effectiveness and functionality of the proposed approach was verified through four structural design problems. In the first three of them, the objective was to increase the first natural frequency of the structure. In the last problem, a piezoelectric actuator was considered for the purposes of shape control and/or active control for correction of static deformations. The design variables were the voltage and the actuator locations and the performance index was considered as the square root of the error between the nodal predefined displacement and the observed displacement.

Acknowledgements The work in this chapter is in the context of the *Con4Coord* project, supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. INFSO-ICT-223844, the Next Generation Infrastructures Research Program of Delft University of Technology and the Mexican CONACyT Project No. 103570.

References

1. J. Reddy. *Introduction to the Finite Element Method*. McGrawHill, New York, 1993. 2
2. M. Papadrakakis, N.D. Lagaros, and G. Kokossalakis. Evolutionary Algorithms Applied to Structural Optimization Problems. *High Performance Computing for Computational Mechanics*, pages 207–233, 2000. 2
3. Z. Michalewicz. *Genetic algorithms + data structures = evolution programs*. Springer-Verlag New York, Inc., New York, NY, USA, 1994. 2
4. M. Walker and R. E. Smith. A technique for the multiobjective optimisation of laminated composite structures using genetic algorithms and finite element analysis. *Composite Structures*, 62(1):123–128, 2003. 2
5. A. Abe, T. Kamegawa, and Y. Nakajima. Optimization of construction of tire reinforcement by genetic algorithm. *Optimization and Engineering*, 5(1):77–92, 2003. 2
6. M. Giger and P. Ermanni. Development of CFRP racing motorcycle rims using a heuristic evolutionary algorithm approach. *Structural and Multidisciplinary Optimization*, 30(1):54–65, 2005. 2
7. E. Alba and M. Tomassini. Parallelism and Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462, 2002. 3

8. J. Mackerle. Smart materials and structuresa finite element approachan addendum: a bibliography (1997 2002). *Modelling and Simulation in Materials Science and Engineering*, 11(5):707–744, 2003. 5
9. L. Joseffsson and P. Persson. *Conformal Array Antenna Theory and Design (IEEE Press Series on Electromagnetic Wave Theory)*. Wiley-IEEE Press, 2005. 5
10. K.M. Liew, X.Q. He, and T. Ray. On the use of computational intelligence in the optimal shape control of functionally graded smart plates. *Computer Methods in Applied Mechanics and Engineering*, 193(42-44):4475–4492, 2004. 5, 24
11. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Co., Reading, Massachusetts, 1989. 6
12. H. Furuya and R. T. Haftka. Locating actuators for vibration suppression on space trusses by genetic algorithms. *ASME-PUBLICATIONS-AD*, 38, 1993. 6
13. J. E. Rodríguez, A. L. Medaglia, and C. A. Coello Coello. Design of a motorcycle frame using neuroacceleration strategies in MOEAs. *Journal of Heuristics*, 15(2):177–196, 2009. 6
14. A. Lemonge, H. Barbosa, and L. Fonseca. A genetic algorithm for the design of space framed structures. In *XXIV CILAMCE–Iberian Latin-American Congress on Computational Methods in Engineering, Ouro Preto, Brazil*, 2003. 6
15. Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12, 2005. 7, 9, 13
16. R. Smith, B. Dike, and S. Stegmann. Fitness inheritance in genetic algorithms. In *Proceedings of ACM Symposiums on Applied Computing*, pages 345–350. ACM, 1995. 7
17. X. Zhang, B. Julstrom, and W. Cheng. Design of vector quantization codebooks using a genetic algorithm. In *Proceedings of the IEEE Conference on Evolutionary Computation*, pages 525–529. IEEE, 1997. 7
18. M. Salami and T. Hendtlass. A fast evaluation strategy for evolutionary algorithms. *Applied Soft Computing*, 2:156–173, 2003. 7, 18
19. M. Pelikan and K. Sastry. Fitness inheritance in the Bayesian optimization algorithms. In *Genetic and Evolutionary Computation Conference*, pages 48–59. Springer, 2004. 8
20. M. Reyes Sierra and C. A. Coello Coello. Fitness Inheritance in Multi-Objective Particle Swarm Optimization. In *2005 IEEE Swarm Intelligence Symposium (SIS'05)*, pages 116–123, Pasadena, California, USA, June 2005. IEEE Press. 8
21. M. Reyes Sierra and C. A. Coello Coello. A Study of Fitness Inheritance and Approximation Techniques for Multi-Objective Particle Swarm Optimization. In *2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, volume 1, pages 65–72, Edinburgh, Scotland, September 2005. IEEE Service Center. 8
22. E. I. Ducheyne, B. De Baets, and R. De Wulf. Is Fitness Inheritance Useful for Real-World Applications? In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 31–42, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632. 8
23. Y. Sano and H. Kita. Optimization of noisy fitness functions by means of genetic algorithms using history. In M. Schoenauer et al, editor, *Parallel Problem Solving from Nature (PPSN)*, volume 1917 of *Lecture Notes in Computer Science*. Springer, 2000. 9
24. J. Branke, C. Schmidt, and H. Schmeck. Efficient fitness estimation in noisy environment. In L. Spector et al, editor, *Proceedings of Genetic and Evolutionary Computation Conference (GECCO)*, pages 243–250, San Francisco, CA, July 2001. Morgan Kaufmann. 9
25. J. Branke and C. Schmidt. Fast convergence by means of fitness estimation. *Soft Computing Journal*, 9(1):13–20, 2005. 9

26. Y.S. Ong, P.B. Nair, and A.J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *American Institute of Aeronautics and Astronautics Journal*, 41(4):687–696, 2003. 9, 10, 13
27. Y. S. Ong, Z. Zhu, and D. Lim. Curse and blessing of uncertainty in evolutionary algorithm using approximation. In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC'2006)*, pages 2928–2935, 2006. 10
28. R.G. Regis and C.A. Shoemaker. Local function approximation in evolutionary algorithms for the optimization of costly functions. *IEEE Transactions on Evolutionary Computation*, 8(5):490–505, 2004. 10
29. D. Lim, Y. S. Ong, Y. Jin, and B. Sendhoff. Trusted evolutionary algorithm. In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC'2006)*, pages 149–156, 2006. 10
30. J. Sacks, W. Welch, T. Mitchell, and H. Wynn. Design and analysis of computer experiments (with discussion). In *Statistical Science*, volume 4, pages 409 – 435, 1989. 11
31. A. Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In A. Eiben, Th. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature*, volume V, pages 87–96, 1998. 11
32. Y.-S. Hong, H.Lee, and M.-J. Tahk. Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks. *Engineering Optimization*, 35(1):91–102, 2003. 11
33. K. S. Won, T. Ray, and K. Tai. A framework for optimization using approximate functions. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 1077–1084, 2003. 11
34. A.-R. Khorsand and M. Akbarzadeh. Multi-objective meta level soft computing-based evolutionary structural design. *Journal of the Franklin Institute*, pages 595–612, 2007. 12
35. Y. Jin, M. Olhofer, and B. Sendhoff. On evolutionary optimization with approximate fitness functions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 786–792. Morgan Kaufmann, 2000. 12
36. L. Shi and K. Rasheed. A survey of fitness approximation methods applied in evolutionary algorithms. In L. M. Hiot, Y. S .Ong, Y. Tenne, and C. K. Goh, editors, *Computational Intelligence in Expensive Optimization Problems*, volume 2 of *Adaptation Learning and Optimization*, pages 3–28. Springer Berlin Heidelberg, 2010. 12
37. H.-S. Kim and S.-B. Cho. An efficient genetic algorithms with less fitness evaluation by clustering. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 887–894. IEEE, 2001. 13
38. M. Bhattacharya and G. Lu. A dynamic approximate fitness based hybrid ea for optimization problems. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 1879–1886, 2003. 13
39. L. G. Fonseca and H. J. C. Barbosa. A similarity-based surrogate model for enhanced performance in genetic algorithms. *OPSEARCH*, 46:89107, 2009. 13
40. Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, second edition, September 2007. ISBN 978-0-387-33254-3. 13
41. E. Mezura-Montes, editor. *Constraint-Handling in Evolutionary Optimization*. Springer, Berlin, Germany, 2009. ISBN 978-3-642-00618-0. 13
42. T. P. Runarsson. Constrained Evolutionary Optimization by Approximate Ranking and Surrogate Models. In Xin Yao, Edmund Burke, José A. Lozano, Jim Smith, , Juan J. Merelo-Guervós, John A. Bullinaria, Jonathan Rowe, Peter Tiňo, Ata Kabán, and H.-P. Schwefel, editors, *Proceedings of 8th Parallel Problem Solving From Nature (PPSN VIII)*, pages 401–410, Heidelberg, Germany, September 2004. Birmingham, UK, Springer-Verlag. Lecture Notes in Computer Science Vol. 3242. 13

43. C. A. Coello Coello. Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, January 2002. 13
44. T. P. Runarsson and X. Yao. Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, September 2000. 13
45. Y. G. Woldesenbet, G. G. Yen, and B. G. Tessema. Constraint Handling in Multiobjective Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 13(3):514–525, June 2009. 13
46. H. Kumar Singh, T. Ray, and W. Smith. C-PSA: Constrained Pareto simulated annealing for constrained multi-objective optimization. *Information Sciences*, 180(13):2499–2513, July 1 2010. 13
47. Luis V. Santana-Quintero, Alfredo Arias Montaño, and Carlos A. Coello Coello. A Review of Techniques for Handling Expensive Functions in Evolutionary Multi-Objective Optimization. In Yoel Tenne and Chi-Keong Goh, editors, *Computational Intelligence in Expensive Optimization Problems*, pages 29–59. Springer, Berlin, Germany, 2010. ISBN 978-3-642-10700-9. 14
48. Mohsen Davarynejad. Fuzzy Fitness Granulation in Evolutionary Algorithms for Complex Optimization. Master's thesis, Ferdowsi University of Mashhad, June 2007. 14, 17, 22, 28
49. L. A. Zadeh. Fuzzy sets and information granularity. *Advances in Fuzzy Set Theory and Applications*, pages 3–18, 1979. 14
50. M. Davarynejad, C. W. Ahn, J. L. M. Vrancken, J. van den Berg, and C. A. Coello Coello. Evolutionary hidden information detection by granulation-based fitness approximation. *Applied Soft Computing*, 10(3):719–729, 2010. 16, 18
51. M.R. Akbarzadeh-T, M. Davarynejad, and N. Pariz. Adaptive fuzzy fitness granulation for evolutionary optimization. *International Journal of Approximate Reasoning*, 49(3):523–538, 2008. 17, 23, 29
52. M. Davarynejad, M.-R. Akbarzadeh-T, and N. Pariz. A novel general framework for evolutionary optimization: Adaptive fuzzy fitness granulation. In *IEEE Congress on Evolutionary Computation*, pages 951–956. IEEE, 2007. 17, 18
53. M. Davarynejad, M.R. Akbarzadeh-T, and Carlos A. Coello Coello. Auto-tuning fuzzy granulation for evolutionary optimization. In *CEC 2008, IEEE World Congress on Evolutionary Computation*, pages 3572–3579, Hong Kong, June 2008. 18
54. I. Ansys. ANSYS users manual. *ANSYS Inc., Southpointe*, 275, 2004. 18, 26
55. J. Freudenberger, J. Gllner, M. Heilmaier, G. Mook, H. Saage, V. Srivastava, and U. Wendt. Materials science and engineering. In K. H. Grote and E. K. Antonsson, editors, *Springer Handbook of Mechanical Engineering*. Springer Berlin Heidelberg, 2009. 19
56. J. Lin and M. Nien. Adaptive control of a composite cantilever beam with piezoelectric damping-modal actuators/sensors. *Composite Structures Journal*, 70:170–176, 2005. 23
57. J. Li, R. Sedaghati, J. Dargahi, and D. Waechter. Design and development of a new piezoelectric linear Inchworm actuator. *Mechatronics Journal*, 15:651–681, 2005. 23
58. S. Adali, I. Sadek, J. Bruch Jr., and J. Sloss. Optimization of composite plates with piezoelectric stiffener-actuators under in-plane compressive loads. *Composite Structures Journal*, 71:293–301, 2005. 24
59. M. Krommer. Dynamic shape control of sub-sections of moderately thick beams. *Computers & Structures*, 83(15-16):1330–1339, 2005. 24
60. T. Weise. Global Optimization Algorithms–Theory and Application. *URL: http://www. it-weise. de, Abrufdatum*, 1, 2008. 24
61. Q. Nguyen and L. Tong. Shape control of smart composite plate with non-rectangular piezoelectric actuators. *Composite Structures*, 66(1-4):207–214, 2004. 25

62. F. Aryana, H. Bahai, R. Mirzaeifar, and A. Yeilaghi. Modification of dynamic characteristics of FGM plates with integrated piezoelectric layers using first-and second-order approximations. *International Journal for Numerical Methods in Engineering*, 70(12):1409–1429, 2007. 25
63. A.-R. Khorsand, M.-R. Akbarzadeh-T, and H. Moin. Genetic Quantum Algorithm for Voltage and Pattern Design of Piezoelectric Actuator. In *IEEE Congress on Evolutionary Computation, CEC 2006*, pages 2593–2600, 2006. 25
64. V. Piefort. *Finite element modelling of piezoelectric active structures*. PhD thesis, Université Libre de Bruxelles, 2001. 26
65. S. da Mota Silva, R. Ribeiro, J. D. Rodrigues, M. A. P. Vaz, and J. M. Monteiro. The application of genetic algorithms for shape control with piezoelectric patches-an experimental comparison. *Smart Materials and Structures*, 13:220–226, 2004. 26, 27
66. D. W. Kelly, J. P. De S. R. Gago, O. C. Zienkiewicz, and I. Babuska. A posteriori error analysis and adaptive processes in the finite element method: Part ierror analysis. *International Journal for Numerical Methods in Engineering*, 19:15931619, 1983. 26