ANNALS OF THE ORADEA UNIVERSITY
Fascicle of Management and Technological Engineering
ISSUE #1, MAY 2013, http://www.imtuoradea.ro/auo.fmte/

STRUCTURAL OPTIMIZATION WITH GENETIC ALGORITHMS AND PARTICLE SWARM OPTIMIZATION

Razvan CAZACU[1], Lucian GRAMA[2]
[1] Industrial Engineering and Management Department, Petru Maior University, Nicolae Iorga Street, No. 1, Targu Mures, Romania, razvan.cazacu@ing.upm.ro
[2] Industrial Engineering and Management Department, Petru Maior University, Nicolae Iorga Street, No. 1, Targu Mures, Romania, lucian.grama@ing.upm.ro

***Abstract*—**This is the second paper in a series intended to highlight the present possibilities offered by the evolutionary computing techniques for structural optimization. In this second part, the two most prominent methods, the genetic algorithm and the particle swarm optimization, are discussed in detail. We will present their classical formulation, main aspects, implementation details, newly proposed enhancements, advantages and drawbacks. The final section also lists available implementations of these algorithms which can be used to develop original computer programs to test improvements of these methods.

***Keywords*—**structural optimization, particle swarm optimization, genetic algorithms

I. INTRODUCTION

STRUCTURAL optimization is a key element in the functional design of mounted structures and cast parts subjected to important loading. The engineer is faced with the challenge of designing a component considering objectives that are often times contradicting, like minimizing the mass and total cost, maximizing the strength and stiffness and easing the manufacturability.

There are 3 types of structural optimization: topology, shape and size. While topology optimization deals with all the design space, trying to find the optimum mass distribution in this space, shape and size optimization assume a certain parameterized layout and try to find the best values for the parameters. Topology optimization has the biggest optimization potential, size and shape optimization being usually used for the fine-tuning of the model [1]. However, the models resulting from topology optimization need to be interpreted and if this interpretation is not done properly, the procedure fails to achieve its aim. In an effort to improve shape optimization, new approaches suggest geometry-based optimization using splines and varying their specification tree in commercial CAD programs [2]. Other researches use a morphological representation of the geometry for the same reason [3]. Some works tackle all three types of optimization at once [4], [5], although this approach is still in its infancy and only tried for simple problems.

Classic optimization is carried out using analytical or empirical methods [1], [6]. This paper presents the possibilities offered by two metaheuristic methods, the Genetic Algorithm (GA) and the Particle Swarm Optimization (PSO). There are also some efforts to mix analytical and evolutionary techniques, like the hybrid GA and Taguchi method presented in [7]. Another mixing possibility is between PSO and GA, which are compared in [8], the author concluding that the best results are obtained by incorporating concepts from one method into the other.

GA and PSO have been proven to be the most efficient for the structural optimization of trusses [9]. Still, according to the "no free lunch" theorem [10] no optimization method is more efficient than any other for all types of problems, so there is a need for further research to study if the two remain as efficient for the optimization of other types of structures.

II. GENETIC ALGORITHMS

The original Genetic Algorithm was proposed in 1975 [11] but it has suffered many adaptations and enhancements over the years, in an effort to make it more efficient and applicable to a wider set of problems.

Belonging to the family of evolutionary programming, it simulates natural evolution concepts. The main idea is to evolve a population of possible solutions towards the optimum. Each individual in the population represents a solution to the given problem and is coded in the form of a chromosome. The chromosome is a collection of genes and each gene is a parameter of the problem. The initial population is randomly generated in the solution space and after that the genetic operators are applied to it to give birth to the next generation. The process continues iteratively for a given number of generations or until a "good-enough" solution is obtained.

The main genetic operators are crossover and mutation, although others are also used in different implementations with more or less success. Crossover (reproduction) is responsible for the exploitation of good solutions, being able to improve on these while mutation ensures the exploration of the entire design space, avowing premature convergence towards local minima. A balance between these two operators is crucial. Too much exploration is similar to random search and does not permit proper optimization near the good solutions, while too much exploitation with little exploration can fall into the trap of local optima. The conceptual algorithm of the GA is presented in Fig. 1.

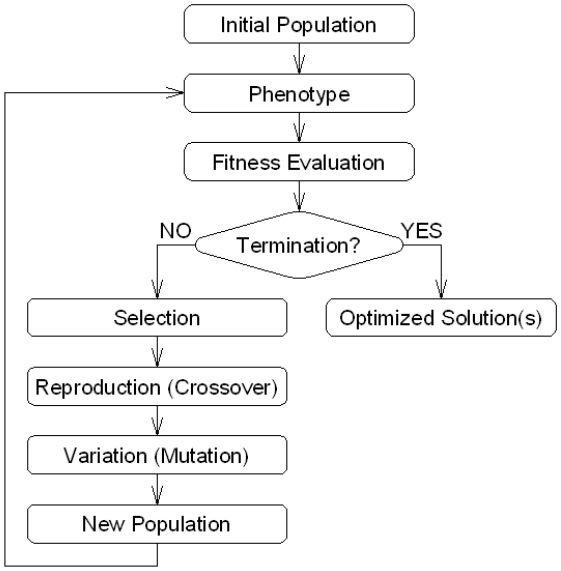

Fig.1. Conceptual flow chart for basic GA

One of the main advantages of GA is that the algorithm needs no knowledge of the problem to be solved although such knowledge can be used to ensure better results. The only connection between the problem and the algorithm is the encoding-decoding and the fitness evaluation.

A. Encoding

One of the biggest challenges in applying a GA is the encoding of the actual problem (the phenotype) in a genetic form (the genotype) as a chromosome. The algorithm must also be able to decode the chromosome and construct the corresponding model.

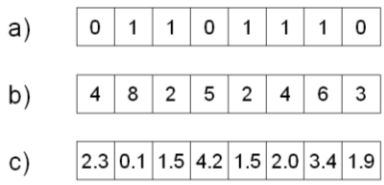

Fig.2. Chromosome representation.
a) binary; b) integer; c) real.

The representation is usually done in the form of a linear string of genes, each gene representing a parameter of the problem. In the original algorithm, the genes are binary, but other types of data can also be used. Fig. 2 presents three types of chromosomes. It is also possible to mix different types of genes in the same chromosome [12], [13]. In such cases, grouping similar genes together gives the best results [12]. The typical chromosome is of fixed length (fixed number of genes), but for certain problems variable-length chromosomes can be used [14]. However, the crossover operator is more complicated to apply in these situations.

While for most applications the linear chromosome is used, for the case of 2D or 3D topological optimization multidimensional chromosomes are more suited [15].

B. Fitness evaluation

The second big issue when implementing GAs is the evaluation of the fitness function. In comparing the solutions, optimization methods evaluate their fitness using the objective function. When the objective is mass, strength and/or stiffness, finite element analysis (FEA) is employed. This is a computationally intensive procedure, requiring huge amounts of resources and long processing times. One of the ways to reduce the computational cost is to implement fitness approximation algorithms, to reduce the number of evaluations needed for one optimization cycle [16], [17].

One important aspect to note is that the FEA is performed on the actual structural model. The algorithm needs to decode the chromosome for each individual (construct the phenotype based on the values of the genes) and then call the FEA solver.

C. Selection

When a new generation is bred, individuals from the current population are selected to pass on their genes. This is usually done by biasing parent selection towards fitter individuals (i.e. better solutions).

Examples of selection types are *roulette-wheel*, where individuals are randomly selected from the whole population (weighting the selection based on fitness) or *tournament* where the fittest individual is selected from a randomly generated subset of the population.

GAs selection often involves *elitism*, meaning the n best individuals will always survive for the next stage, thus keeping the best solutions in the solution pool.

D. Crossover

Crossover or reproduction is the process of breeding new individuals from existing ones, by combining their genes. The three main types of crossover are shown in Fig. 3. Single and multi-point crossover takes blocks of genes (separated at random points) from each parent and combines them. Uniform crossover needs a randomly generated binary mask and combines genes from the two parents based on it. For 2D chromosomes [15] shows an adapted crossover operator gives better results. Fig. 4 presents the block crossover, a generalization of the two-

point operator in two dimensions.

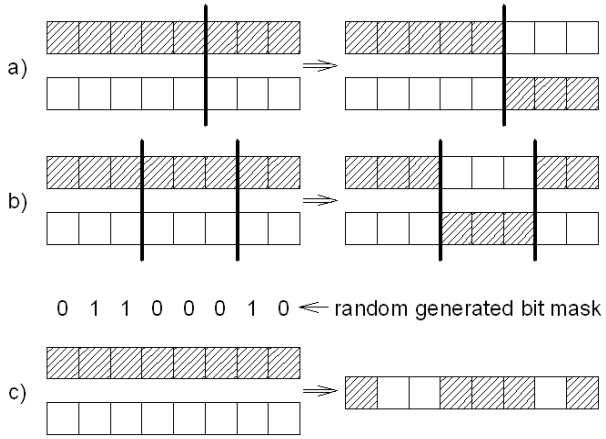

Fig.3. Crossover. a) single-point; b) multi-point; c) uniform.

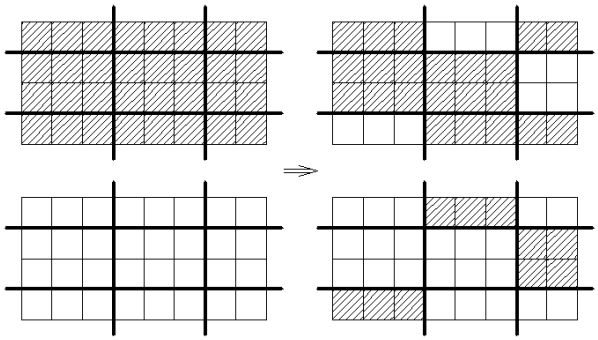

Fig.4. Block crossover for 2D chromosome.

E. Mutation

Mutation is set to occur randomly with a prescribed probability and is usually employed on newly born individuals. It simply randomly changes one gene to another acceptable value. Binary mutation means changing the value from 1 to 0 or vice versa. Fig. 5 shows examples of mutation for different types of genes.

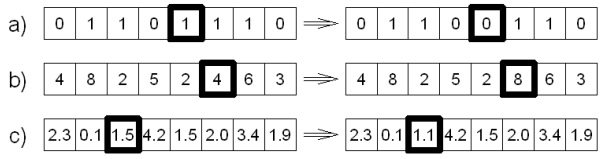

Fig.5. Mutation. a) binary; b) integer; c) real.

Instead of the classical random mutation, epistatic mutation can be used to mutate the genes that are almost the same across all or almost all the individuals in the population, leading to greater diversity and escaping local minima [15]. Another way of controlling mutation is adaptive directed mutation [18].

Because mutation rate is such an important and difficult to set parameter, adaptive mutation rate can be employed, where the mutation rate is encoded in the chromosome. Along mutation, the scientific literature also proposes other techniques such as diversity guided evolutionary programming [19] and immunization [20] for avoiding local minima and ensure a proper exploration of the design space.

III. Particle Swarm Optimization

Particle swarm optimization (PSO) is the most representative optimization method from the family of swarm intelligence and it was first introduced in 1995 [21]. Like GA, PSO works iteratively and uses a population of solutions, this time called particles, each solution being comprised of a set of parameters, but unlike GA the population stays the same and the parameters of the particle give its "position" in the design space. Each individual moves (changes its position) according to its best known position (pBest) so far and the swarm best historical position (gBest). The conceptual algorithm for basic PSO is described in Fig. 6.

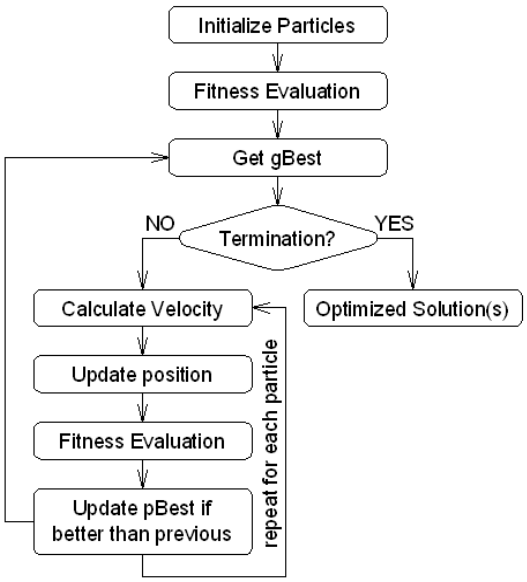

Fig.6. Conceptual flow chart for basic PSO

There are very few PSO implementations for topology optimization in the literature. However, shape and size optimization is employed using PSO in some papers [22].

A. Individual movement and parameter selection

Each particle i moves independently, adjusting its speed $\boldsymbol{v}_i$ according to the current value, its best position so far $\boldsymbol{pB}_i$ and the swarm best position $\boldsymbol{gB}$. The speed and updated position $\boldsymbol{p}_i$ are calculated separately for each dimension x, using (1):

$$\begin{aligned} \mathbf{v}_i^x &= \omega \mathbf{v}_i^x + {}_p\mathbf{r}_p\left(\mathbf{pB}_i^x - \mathbf{p}_i^x\right) + {}_g\mathbf{r}_g\left(\mathbf{gB}_i^x - \mathbf{p}_i^x\right) \\ \mathbf{p}_i^x &= \mathbf{p}_i^x + \mathbf{v}_i^x \end{aligned} \quad (1)$$

The coefficients $\boldsymbol{r}_p$ and $\boldsymbol{r}_g$ are randomly chosen in the open interval (0,1). The parameters specifying the influence of current speed (ω), individual best position (φ_p) and global best position (φ_g) on the speed of the particle are set at the beginning, choosing their right

values being crucial for the performance of the algorithm.

B. Special considerations

PSO is traditionally employed for real coded parameters (continuous), but can be modified to be used for binary or discrete problems by using mapping (perform the calculations normally and then simply round the actual values to match the closest one in the discrete space) or by adapting the mathematical operators in the formulae to work on discrete sets of values [23].

PSO has also been successfully used for multimodal function optimization, where the aim is to find more local optima for the solution instead of just the global one. A complete survey on this matter is done in [24].

IV. Implementation in Computer Programs

There are several software libraries targeted for different programming environments and languages, both commercial and scientific [25]. The most are built as add-ons for MATLAB, which also offers a built-in Genetic Algorithm toolbox. GPLAB [26] is one of the toolboxes developed for MATLAB and can be used both for actual optimization problems as well as for research. Its flexible structure allows the testing of new genetic algorithm operators and procedures.

Although there are some efforts in this direction, more software libraries targeted for the.NET framework would be very useful in integrating metaheuristic methods in the modern paradigm of object oriented programming.

V. Conclusions And Further Research

As this paper and the cited references show, metaheuristic methods have a great potential for optimization in structural problems, being able to handle especially complex and discontinuous problems, impossible to be tackled with analytical methods. However, the full potential is still to be exploited by future research. Of special interest is the way topology, shape and size optimization can be carried out for cast parts using genetic algorithms, particle swarm optimization or a combination of the two.

References

[1] L. Harzheim and G. Graf, “A review of optimization of cast parts using topology optimization. I - Topology optimization without manufacturing constraints”, Structural and Multidisciplinary Optimization, Volume 30, Issue 6, 2005, pp 491-497.

[2] D. Weiss, “Feature-based spline optimization in CAD”, Structural and Multidisciplinary Optimization, Volume 42, Issue 4, 2010, pp 619-631.

[3] K. Tai and J. Prasad, “Multiobjective Topology Optimization Using a Genetic Algorithm and a Morphological Representation of Geometry”, 6th World Congress of Structural and Multidisciplinary Optimization, 2005.

[4] N. Noilublao and S. Bureerat, “Simultaneous topology, shape and sizing optimisation of a three-dimensional slender truss tower using multiobjective evolutionary algorithms”, Computers and Structures, Volume 89, Issues 23-24, 2011, pp 2531-2538.

[5] M. Zhou et al, “An integrated approach to topology, sizing, and shape optimization”, Structural and Multidisciplinary Optimization, Volume 26, Issue 5, 2004, pp 308-317.

[6] L. Harzheim and G. Graf, “A review of optimization of cast parts using topology optimization. II - Topology optimization with manufacturing constraints”, Structural and Multidisciplinary Optimization, Volume 31, Issue 5, 2006, pp 388-399.

[7] A.R. Yıldız, N. Ozturk, N. Kaya and F. Ozturk, “Hybrid multi-objective shape design optimization using Taguchi’s method and genetic algorithm”, Structural and Multidisciplinary Optimization, Volume 34, Issue 4, 2007, pp 317-332.

[8] R.C. Eberhart and Y. Shi, “Comparison between genetic algorithms and particle swarm optimization”, Proceedings of the 7th International Conference (EP98 San Diego), Volume 1447 of Lecture Notes in Computer Science, 1998, pp 611-616.

[9] S. Sanchez-Caballero et al, “Recent Advances in Structural Optimization”, Annals of the Oradea University, Fascicle MTE, Volume XI (XXI), 2012, pp 2.118-2.127.

[10] D.H. Wolpert and W.G. Macready, “No Free Lunch Theorems for Optimization”, IEEE Transactions on Evolutionary Computation, Volume 1, Issue 1, 2007, pp 67-82.

[11] J.H. Holland, “Adaptation in Natural and Artificial Systems”, University of Michigan Press, 1975, ISBN 0-262-08213-6.

[12] J.E. Rodriguez, A.L. Medaglia and J.P. Casas, “Approximation to the optimum design of a motorcycle frame using finite element analysis and evolutionary algorithms”, Proc. of the 2005 Systems and Information Engineering Design Sympos. 2005, pp 277-285.

[13] O. Buiga and C.O. Popa, “Optimal mass design of a single-stage helical gear unit with genetic algorithms”, Proceedings of the Romanian Academy, Volume 13, Issue 3, 2012, pp 243-250.

[14] I.Y.Kim and O. de Weck, “Variable Chromosome Length Genetic Algorithm for Structural Topology Design Optimization”, 45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference, 2004, pp 1-12.

[15] C. Kane, F. Jouve and M. Schoenauer, “Structural topology optimization in linear and nonlinear elasticity using genetic algorithms”, 1995.

[16] Y. Jin, “A comprehensive survey of fitness approximation in evolutionary computation”, Soft Computing, Volume 9, Issue 1, 2005, pp 3-12.

[17] J. Ziegler and W. Banzhaf, “Decreasing the number of evaluations in evolutionary algorithms by using a meta-model of the fitness function”, Proceedings of the 6th European Conference in Genetic Programming (EuroGP'03), Volume 2610 of Lecture Notes in Computer Science, 2003, pp 264—275.

[18] P.H. Tang and M.H. Tseng, “Adaptive directed mutation for real-coded genetic algorithms”, Applied Soft Computing, Volume 13, 2013, pp 600-614.

[19] M.S. Alam. et al, “Diversity Guided Evolutionary Programming: A novel approach for continuous optimization”, Applied Soft Computing, Volume 12, Issue 6, 2012, pp 1693-1707.

[20] Y. Lu et al, “A new immune genetic algorithm”, Computer and Automation Engineering (ICCAE), Volume 1, 2010, pp 714-718.

[21] J. Kennedy and R. Eberhart, “Particle Swarm Optimization”, Proceedings of IEEE International Conference on Neural Networks, Volume 4, 1995, pp 1942-1948.

[22] P.C. Fourie and A.A. Groenwold, “The particle swarm optimization algorithm in size and shape optimization”, Structural and Multidisciplinary Optimization, Volume 23, Issue 4, 2002, pp 259-267.

[23] W. Chen and J. Zhang, “A novel set-based particle swarm optimization method for discrete optimization problem”, IEEE Transactions on Evolutionary Computation, Volume 14,Issue 2, 2010, pp 278-300.

[24] Y. Liu, “A Survey on Particle Swarm Optimization Algorithms for Multimodal Function Optimization”, Journal of Software, Volume 6, No. 12, 2011, pp 2449-2455.

[25] T. Sag and M. Cunkas, “A tool for multiobjective evolutionary algorithms”, Advances in Engineering Software, Volume 40, 2009, pp 902-912.

[26] S. Silva, “GPLAB - a Genetic Programming toolbox for MATLAB”, Proceedings of the Nordic MATLAB Conference, 2003, pp 273-278.