A MODERN METHODOLOGY OF DESIGN OF THREE-DIMENSIONAL STRUCTURES BY A GENETIC ALGORITHMS APPROACH

Sofiane BENANANE[1], Djamel KERDAL[1], Abdelkader BENANANE[2], Abderrahmane OUAZIR[2], Messaoud TITOUM[3]

The computer aided design is realized today by the significant development of computational tools. These computer codes are often intended for advanced design phase of projects. However, there is to our knowledge very few design support tools in preliminary design phase. Indeed, in the life cycle of a construction project, the design phase is often the place of conflicting situations that prevent the overall optimization of the said projects production costs. During this phase, various technical treatments should be held to verify the feasibility of the works in relation to the structural constraints, neighborhood, implementation, etc. In this work, we propose a formulation of the optimization problem of the overall design of a simple metallic structure and a methodology of resolution based on the approach of Genetic Algorithms. The aim is to minimize the overall execution cost.

Keywords: Multidisciplinary optimization, numerical programming, computer aided design, genetic algorithms, optimal design.

1. Introduction

The traditional approach to optimization of metallic structures is based on the minimization of the structure weight. However, the assemblies rarely exceed 5% of total weight of a frame. This low percentage actually hides in reality a high cost which can reach 30% of the total manufacturing cost of a structure [1]. Indeed, the cost of a frame is mainly constituted by the labor cost which essentially depends on the complexity of the assemblies. An optimized structure definition, made only on the unique weight criterion may therefore lead to structural arrangements far from optimal in terms of realization cost.

On the other hand, the modeling of assemblies can affect, significantly, the distribution of internal forces in the structure and also the forces to resume in the

[1] Laboratory of Mechanics of Structures and Constructions Stability, University of Oran, Algeria: benanane.sofiane@yahoo.com, djkerdal@yahoo.fr

[2] Laboratory of Materials and Processes of Construction, University of Mostaganem, Algeria: abdelkaderbenanane@yahoo.fr, abderouazir@yahoo.fr

[3] Laboratory of Materials and Mechanics of Structures, University of M'sila, Algeria: titoum65@yahoo.fr

foundations. That is why Eurocode 3 [2] now allows the use and justification of semi-rigid connections.

Naturally, the aim is to approach as much as possible, the actual behavior of connections. Taking into account the behavior of nodes in the overall analysis is an innovative and promising aspect.

The economic benefits of this approach were the subject of various benchmarking [3]. Its implementation is greatly facilitated by appropriate analysis software already available on the market [4] and various aids to calculation and to characterizing the nodes [5].

For this we have developed an optimization methodology based on minimizing the total cost of realization of the structure. This cost includes material costs, manufacturing and assembly of the metallic superstructure thus as the material costs and realization of foundation systems. This global optimization approach based on the application of Genetic Algorithms, takes into account, in addition, the dimensional characteristics of the elements, the nature of supports and the design of connections.

2. The optimization methods

There are many methods for solving an optimization problem. These methods can be divided into two groups:

-the so-called deterministic methods,

-the stochastic methods or called non-deterministic.

The deterministic methods, such as the gradient, are favourable to a local optimum search but do not allow to leave the wells to find the global optimum.

The non-deterministic methods, such as Monte Carlo, can avoid convergence to a local optimum. Therefore, the genetic algorithms (GA) offer many advantages over conventional optimization methods

3. Overview of an optimization problem with genetic algorithm

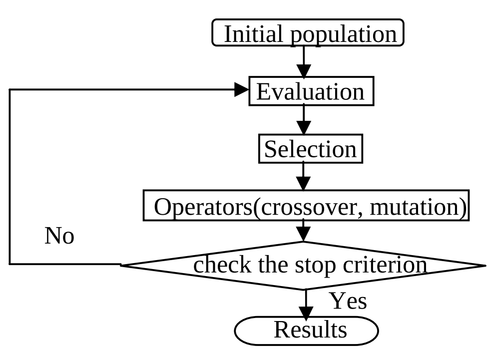

Fig. 1. Presentation of a flowchart-based on genetic Algorithms

A such algorithm requires no knowledge of the problem: we can represent it as a black box with inputs (variables) and outputs (the objective functions). The algorithm only manipulates entries, reads out, again manipulates the inputs so as to improve the outputs, etc. [6].

Three types of evolutionary algorithms have been developed separately and almost at the same time by different scientists: Evolutionary programming [7], the evolutionary strategies [8] and Genetic Algorithms [9].

4. Operating principle of a genetic algorithm

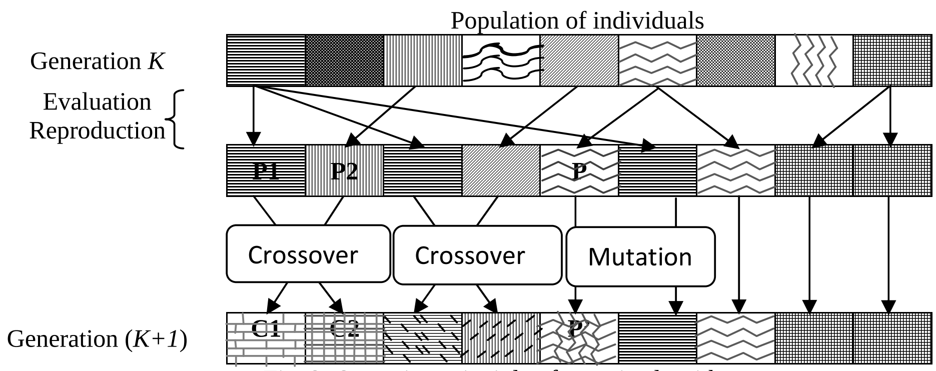

Fig. 2. Operating principle of genetic algorithms

We start by generating a random population of individuals. To pass from one generation *k* to generation *k+1*, the following three operations are repeated for all elements of the population *k*.

✓ Couples of parents *P1* and *P2* are selected according to their adaptations. The crossover operator is applied to them with a probability p_c (usually around 0.6) and generates couples of children *C1* and *C2*.

Other elements *P* are selected according to their adaptation. The mutation operator is applied to them with the probability p_m (p_m is generally between 1% -1 %0) and generates mutated individuals *P'*. Children (*C1*, *C2*) and mutated individuals P' are then evaluated before insertion into this new population, the case where children and mutated individuals replace the parents. Different stopping criteria of the algorithm can be chosen.

✓ The number of generations that it is desired to execute can be fixed a priori. This is what we are attempted to do when we have to find a solution in a limited time.

✓ The algorithm can be stopped when the population does not evolve or does not evolve fast enough.

Now, the different operators introduced above will be studied in detail in the following section:

4.1. The coding

There are three main types of usable coding, and you can switch from one to another easily.

- ✓ *The binary encoding*: this is the most used. The simplest decoding function that allows the transformation of a bit string to an integer x is given by:

$$x = \sum_{i=1}^{n} b_i \, 2^{n-1} \quad (1)$$

Thus the chromosome $A = \{1; 0; 1; 1\}$ is the binary modeling of integer number $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11$

- ✓ *The actual encoding*: is particularly useful where one seeks the maximum of a real function. The actual encodings are now widely used, especially in the fields of application for the continuous variable problems optimization.
- ✓ *The Gray encoding*: where the binary encoding begins to show its limits, this disadvantage can be avoided by using a "Gray coding".

4.2. The selection operator

The selection plays a very important role in genetic algorithms: firstly, to direct research towards the best individuals and secondly, to maintain the diversity of individuals in the population.

Several forms of selection are possible, the best known are [10]:

-The linear selection compared to the rank; the uniform selection with respect to the rank; the selection by the elitist method (or proportional selection); the selection by the elitist method (or proportional selection); the selection by tournament and the selection by the method of biased lottery.

4.3. The crossover operator

This operator is applied after applying the selection operator upon the population P; it finds itself with a population P' of $n/2$ individuals and one has to double that number so that our next generation is full.

So we will randomly created $n/4$ couples and we will authorized them "to recur". The chromosomes (sets of parameters) of parents are then copied and recombined to form two offsprings with characteristics issued from both parents.

Table 1

Crossover with a crossover point

2 Parents					→	2 Children				
A	B	C	D	E		A	B	H	I	J
F	G	H	I	J		F	G	C	D	E

4.4. The mutation operator

This operator has many advantages:

- ✓ It guarantees the diversity of the population, which is essential for the genetic algorithms.

✓ It allows to limit the risks of a premature convergence caused for example by a elitist selection method requiring too much selective pressure at the population.

✓ It allows to reach the ergodicity property which is a property ensuring that each point of the search space can be reached. Due to this property, it is possible to reach the global optimum. This operator consists to change allelic value of a gene with a very low probability p_m, usually between 0.01 and 0.001.

Table 2

Example of a mutation

A	B	H	D	E
	Mutation	↓		
F	G	C	I	J

4.5. The replacement operator

This operator is the simplest, his job consists to reintroduce the offspring obtained by successive application of the selection operators, crossover and mutation (population *P'*) in the population of their parents (*P* population).

4.6. Comprehensive approach: Example illustrating the implementation of all previous operators

This example first allows to understand easily the formalism of genetic algorithms but also to give an idea about the manner of their programming.

This example is due to Goldberg [11]. It is to find the maximum of the function $f(x) = x$ on the interval [0, 31] where x is a natural number. There is 32 possible values for x: Therefore we choose a discrete coding of 5 bits: for example, one thus obtains the sequence 0, 1, 1, 0,1 for 13, the sequence 1, 1, 0, 0,1 for 25, etc.

First, we initialize randomly the population and we fix its size to 4 individuals. We simply define the fitness as being the value of x, since we seek the maximum value throughout the interval [0, 31]. A higher value of x implies higher precision in finding the maximum of the identity function.

If, by definition, the "fitness" is "the value of x" as one can deduce from the previous sentence, then it is evident that increasing x is equivalent with increasing the fitness.

Table 3

The initial population

Individuals	Sequence	Fitness	% of total
1	01011	11	18.6
2	10011	19	32.2
3	00101	5	8.5
4	11000	24	40.7
Total		59	100

Secondly, we choose for example a selection by the method of biased lottery.

We rotate the wheel successively four times, typically we rotate n/2 times where n represents the individuals number, so 2 times in this case, but the number 2 is too small, we decide to turn it 4 times. We then obtain the new population:

Table 4

Individuals selected by the biased lottery method

Individuals	Sequence
1	11000
2	00101
3	11000
4	01011

Thirdly, the crossover operator is applied using a single point crossover. Normally each couple gives 2 children that are added to our new population *P'* passing from *n*/2 individuals to n individuals, but as, in the case of our example we have already reached our n individuals, the 2 children of the couple will then replace their parents. So two couples are formed randomly:

- Couple 1: the individual 2 with the individual 4.
- Couple 2: the individual 1 with the individual 4.

The crossover points are also randomly drawn. We get the following result:

Table 5

Results of application of the crossover operator with a single point

Parents	Children
00 | 101 01 | 011	01011 00101
110 | 00 010 | 11	01011 11000

We apply the mutation operator which randomly chooses whether to make a mutation and on which locus to do.

Table 6

Results of application of the mutation operator on individuals generated by crossing

Chromosome before mutation	Chromosome after mutation
01011	11011
00101	00101
01011	01111
11000	11000

Fourthly, then we apply the replacement operator who decides to replace 100% of the population *P*, the population *P* is entirely replaced by *P'* and its size remains fixed.

Table 7

The new population after application of various operators

Individuals	Sequence	Fitness	% of total
1	11011	27	38
2	00101	5	7
3	01111	15	21.1
4	11000	24	33.8
Total		71	100

In one iteration, the maximum is increased from 24 to 27, and overall fitness of the population has relatively increased from 59 to 71. We stop here for this example but in reality, we will continue to produce successive generations until obtaining the global maximum: 31

5. Illustrated examples: Application of genetic algorithms to steel structures

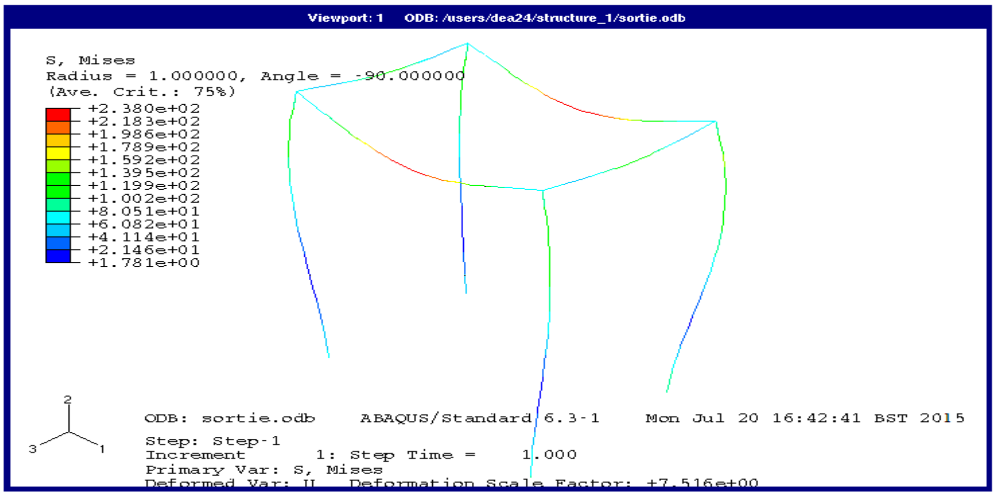

Fig. 3. Structure 3D with circular sections

To show the interest of the suggested methodology and validate our results, the structure has been analysed using two different approaches:

- ✓ The first is the *Monte-Carlo optimization*
- ✓ The second is the *optimization by the genetic algorithms* which is the object of our study and which involves the tools that we have developed ourselves.

Common assumptions to both approaches

-The connections between elements (n = 4) are perfectly embedded (so rigid)

-We adopt two classes of bars for the steel structure (S1 for columns section and S2 for beams section) and two parameters for slabs of foundations (L1 for length and L2 for width).

- The yield strength of steel is σ_e = 235 MPa.

- The allowable differential settlement is equal to 0.1mm

- The foundation depth is fixed to 2m.

-The financial envelope reserved to this project is: 2.0 e^{03} €

Notations regarding the Monte-Carlo table

- Local gap: represents the Monte Carlo accuracy

- Min gap: is the best gap of the Euclidean norm at the current iteration i of MC

- Current gap: is the min gap of the previous iteration. If this is the first iteration, its value is predefined in the source code and is equal to 10^4 (that is the largest gap known at present for the Monte Carlo algorithm).
-nit: represents the number of Monte-Carlo iterations
-Resumpt: represents the number of Monte Carlo resumption
The results obtained by the Monte Carlo method are presented in the following table:

Table 8

The Monte-Carlo optimization

numet	Abaqus Calcula-tion	4	4	4
Par_nomgene.inp (mm^2)and (mm)	S1=350 S2=300 With embedded supports	S1=350 S2=300 Length =1000 Width =500	S1=461.00 S2=199.44 Length =758.2 Width =363.3	S1=459.48 S2=208.05 Length =747.4 Width =270.2
Parameters variation (%)		50	50	50
Response values (MPa)	$\sigma_1 = 268.865$ $\sigma_2 = 176.253$	$\sigma_1 = 230.378$ $\sigma_2 = 138.638$ gap_max=-28617.2 diff _settl =4.22e^{-04} cost= 1173.68	$\sigma_1 = 233.491$ $\sigma_2 = 144.196$ gap _max=-27250.5 diff _settl =3.91e^{-04} cost=1051.74	$\sigma_1 = 233.516$ $\sigma_2 = 136.624$ gap _max=-25147.4 diff _settl =4.39e^{-04} cost=846.21
Optimised parameters (mm^2) and (mm)		S1=461.00 S2=199.44 Length=758.2 Width=363.3	S1=459.48 S2=208.05 Length=747.4 Width =270.2	S1=464.64 S2=181.96 Length=742.5 Width =352.4
Min gap		2510.4	2491.2	2428.8
Current gap		10^4	2510.4	2491.2
Local gap		0.001	0.001	0.001
nit+Resumpt MC		1	1	1

Notations regarding the genetic algorithms method

The problem of the overall design of steel structures may, in our opinion, be globally posed as an optimization problem for minimizing the objective function or the criterion of the overall cost *(CG)* of the structure in accordance with the conditions or constraints of Eurocode .
The *(CG)* is a function of a number of variables such as the three variables: I, X_a and X_n that will be explained explicitly in the following sections.
Thus, we can implicitly formulate the optimization problem as follows:

$$Min\ CG\ (I, X_a, X_n) \tag{2}$$

where:

CG: represents the overall production cost of the structure; *I*: vector of dimensional characteristics of bars; X_a: vector of the supports nature; X_n: vector of the nodes nature.

The overall cost *(CG)* of a metallic construction (superstructure and foundations) can be written:

$$(CG) = (CS) + (CF) \tag{3}$$

with:

(CS): Cost of steel superstructure; *(CF)*: Cost of reinforced concrete foundations.

The cost *(CS)* is itself composed of the following costs:

$$(CS) = (Mat) + (Fab) + (Mon) \tag{4}$$

with:

(Mat): Materials cost of profiles and assemblies; *(Fab)*: Manufacturing cost;

(Mon): Assembly cost of different elements on site.

The cost *(CF)* consists of two following basic units:

$$(CF) = (Ter) + (P_rF) \tag{5}$$

with:

(Ter): represents the excavations cost; *(P_rF)*: Production cost of foundations.

Exploitation of results:

- <u>*Concerning the Monte Carlo approach*</u>

It should be noted that each Monte Carlo iteration requires 3^4 = 81 draws (so 243 calculations for this structure).

The results exploitation of the Monte Carlo optimization shows that the optimal solution is the one that corresponds to the minimum of min-gap (here equal to 2428.8). Therefore, the solution is the vector (S1 = 464.64, S2 = 181.96, Length = 742.5, width = 352.4). The local gap is not being reached, *it is therefore a local optimum*. However, this solution corresponds to minimum cost (here equal to 846.21€).

<u>*- Concerning the genetic algorithms approach*</u>

It is notoriously known that the Genetic Algorithms often use vectors and matrices to modelise engineering problems. The choice of the most appropriate programming language is obviously MATLAB.

Therefore, our application example was implemented with MATLAB and interfaced to calculation code based on finite elements ABAQUS. By following scrupulously the steps of paragraph 4.6 above and applying the values 0.6 and 0.006 respectively for the probabilities of crossover and mutation, the best solution vector found by genetic algorithms method after 27 iterations is:

V=(373.57, 146.30, 596.97, 283.33) which corresponds to 680.35 €.

5. Conclusions

The analysis of the two approaches provides the following synthesis table:

Table 9

Comparison of two approaches

	STRUCTURE		FOUNDATIONS			TOTAL COST
	Section Columns (mm^2)	Section Beams (mm^2)	Lenght (mm)	Width (mm)	Depth (mm)	(euros)
Monte-Carlo Approach	464.64	181.96	742.5	352.4	2000.00	846.21
Genetic algorithms Approach	373.57	146.30	596.97	283.33	2000.00	680.35

The contribution of genetic algorithms methodology can be represented by the relative ratio of costs produced by each of two approaches.

$\boldsymbol{R = \frac{846{,}21 - 680{,}35}{846{,}21} = 0{,}196}$ or approximately a gain of 20%

It therefore appears that this new approach is very promising and should be further refined by a number of measures before the validation of the results. Indeed, for the sake of objectivity, the order of the gain magnitude of recognized cost for this sample cannot be generalized for the simple reason that our tool is in its first release and it therefore incorporates not all optimization constraints of the Eurocode in terms of resistance calculation, stability and dimensioning of constructions.

In final version, an extensive program of numerical simulations on various conceptions can allow to elaborate "design rules" that lead, in early phase, the designers towards optimized total solutions.

R E F E R E N C E S

[1]. *M. Hamchaoui*, Conception économique des assemblages en Construction Métallique. Thèse de doctorat, Université de Savoie, France, 1997.

[2]. Calcul des structures en acier et Document d'Application Nationale – CTICM-Paris, AFNOR 1992.

[3]. *A. Colson, J.M. Hottier, A. Moricet*, Modèle simplifié d'assemblages semi-rigides. Analyse économique comparative. Revue de la Construction Métallique, Vol.4, 1996, pp.55-67.

[4]. *Y. Galea , A. Bureau*, PEPMicro - Analyse plastique au second ordre de structures planes à barres. Manuel d.utilisation, CTICM, France, 1998.

[5]. *J.P. Jaspart*, Steel moment connections according to Eurocode3, Simple design aids for rigid and semi-rigid joints. In: COST C1. Proceedings of the second state of the art workshop, Prague, 1994, pp.159-167.

[6]. *D. Whitley*, Genetic Algorithms in Engineering and Computer Science, chapter Modeling hybrid genetic algorithms, John Wiley, 1995, pp. 191–201.

[7]. *D.B. Fogel*, Evolutionary Computation, IEEE Press, 1996.

[8]. I. *Rechenberg*, Optimierung technischer systeme und prinzipien der biologischen evolution. PhD thesis, Frommann-Holzboog, Stuttgart, 1971.

[9]. *J.H. Holland*, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Ann Arbor, MI: University of Michigan Press ,1975.

[10]. *I.L. Chambers*, The practical handbook of genetic algorithms, applications. Chapman & Hall/CRC, 2001.

[11]. *D.E. Goldberg*, Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison Wesley,1989.