Evolutionary Identification of Macro-Mechanical Models

Marc Schoenauer[1]**, Michèle Sebag**[2]**,**
François Jouve[1]**, Bertrand Lamy**[1]
and Habibou Maitournam[2]

Chapter 23 of *Advances in Genetic Programming II*, pp 467–488
P. J. Angeline and K. E. Kinnear Jr editors,
MIT Press, 1996.

[1] Centre de Mathématiques Appliquées – URA CNRS 756
[2] Laboratoire de Mécanique des Solides – URA CNRS 317
Ecole Polytechnique – 91128 Palaiseau Cedex – FRANCE
E-mail: `Prenom.Nom@polytechnique.fr`

Marc Schoenauer, Michèle Sebag, François Jouve, Bertrand Lamy and Habibou Maitournam

This chapter illustrates the potential of genetic programming (GP) in the field of macro-mechanical modeling, addressing the problem of identification of a mechanical model for a material. Two kinds of models are considered. One-dimensional dynamic models are represented via symbolic formulations termed *rheological models*, which are directly evolved by GP. Three-dimensional static models of hyperelastic materials are expressed in terms of strain energy functions. A model is rated based on the distance between the behavior predicted by the model, and the actual behavior of the material given by a set of mechanical experiments. The choice of GP is motivated by strong arguments, relying on the tree-structure of rheological models in the first case, and on the need for first and second order derivatives in the second case. Key issues are the exploration of viable individuals only, and the use of Gaussian mutations to optimize numerical constants.

23.1 Introduction

This chapter is concerned with applying GP [Koza 1992] in the field of solid mechanics. More precisely, the aim is to discover a model, or *constitutive law*, characterizing the mechanical behavior of a material from mechanical experiments.

The design of modern structures requires ever more detailed knowledge of the constitutive properties of materials. Such knowledge is needed to predict through numerical simulations the behavior of the structure under external loads. Reliable predictions allow for meeting the engineering requirements at a lower cost. A number of new materials (composite materials, polymers), have recently entered into wide usage and no accurate model of their behavior is so far available. The identification of a constitutive law for new materials therefore becomes a major challenge for mechanical science and industry.

23.1.1 State of the art

The design of an accurate law requires considerable insight in mechanics. When a new material is expected by experts to resemble a well known material, the model of the latter is adjusted. Otherwise, an ever stronger sense of mechanics is needed: Brand new models are elaborated through a time-consuming trial and error process, where the successive models guessed by mechanical engineers are checked against test experiments; these experiments may in turn suggest new models [Lemaitre and Chaboche 1985]. One can also start with a thorough analysis of the mechanical behavior of the material at the microscopic scale; a macroscopic law is then derived from the microscopic model, for instance by *homogenization* [Sanchez-Palencia and Zaoui 1987]. However, such models often result in tremendously time-consuming numerical simulations because of their complexity.

Once the current law is found plausible, its numerical parameters are tuned by minimizing a distance between the observed behavior of the material and the behavior predicted from the law for a set of experimental conditions. This stage of identification is equivalent to parametric optimization [Gittus and Zarka 1985].

Much attention has been paid in designing experiments in order to adequately check a given law [Zarka & al. 1988]. But searching for an accurate law remains a critical issue, as all above approaches fail in many cases: when the target material does not resemble any previous material; when the microscopic analysis does not allow a tractable model to be built ; when the (sometimes highly nonlinear) behavior of the material prevents the expert from fitting any appropriate model.

23.1.2 Goal

The goal of this chapter is to build a system able to provide a mechanical engineer with laws fitting data from available experiments. This amounts to non-parametric optimization, since it involves finding both the structure of the law, i.e., a set of partial differential equations, and the numerical coefficients in these equations. Two kinds of models are considered:

• the one-dimensional dynamic aspects of the mechanical behavior are handled by means of rheological models [Persoz 1960]. These models describe a material as an assembly of elementary elastic, plastic or viscous components. Only mixed rheological assemblies composed of serial and/or parallel branches are considered here. GP operates directly on such rheological models, represented by trees within an *ad hoc* set of functions and terminals. An interpreter of rheological trees then produces the equivalent system of partial differential equations.

• the three-dimensional static aspects of the mechanical behavior of hyperelastic materials are captured by a strain energy function [Green and Zerna 1968]. Energy functions are described via standard function and terminal sets; but not all trees represent valid energy functions and much attention is paid to exploring only viable individuals, i.e., leading to a successful numerical resolution of the underlying system of partial differential equations.

In both cases, solving the system of partial differential equations gives the simulated response of the material when external loads are applied. The fitness of a law is computed as the difference between the simulated and the actual responses, for a set of mechanical experiments. It is worth noting that an automatic building of such laws is beyond reach for all standard identification approaches. As far as we know, this chapter reports the first attempt to use GP to this end.

23.2 One-dimensional rheological models

This section addresses the building of general (visco-elasto-plastic) one-dimensional models using mechanical tests. We restrict ourselves to finding the equation which links the stress function $\sigma(t)$ to the strain applied on this material $\epsilon(t)$ and its time derivative $\dot{\epsilon}(t)$, of the form $\sigma(t) = \mathcal{F}(\epsilon(t), \dot{\epsilon}(t))$.

The presentation will mainly focus on the genetic programming aspects of this work: First, the search space is described in terms of trees, based on a set of *ad hoc* nodes and terminals. An interpreter is devised to compute the constitutive law corresponding to a given tree. Much care is then spent in designing specific strategies to address the optimization of the numerical constants and the exploration of the restricted search space.

23.2.1 The search space

Rheological models allow one to describe most mechanical laws in the one-dimensional frame; they can be thought of as an assembly of elementary components, namely springs, sliders and dashpots, which respectively symbolize elastic, plastic and viscous behaviors. In particular, a rheological model, far from being a black box, provides a deep understanding of the constitutive properties of the current material and can be rearranged by the expert. We restrict ourselves to rheological assemblies composed of serial and/or parallel branches, which allow for describing most known materials. Figure 23.1a shows an example of such rheological model, which was proposed for polyethylene [Kichenin 1992].

Such restricted rheological trees can be represented as tree structures built from two functions (respectively the serial and parallel assembly modes) and three

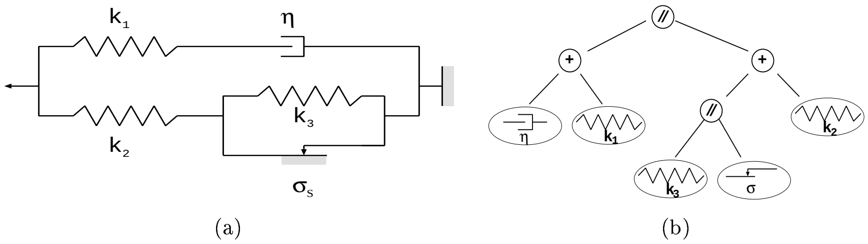


Figure 23.1

A tentative model of polyethylene. (a) Rheological Model. (b) Rheological Tree.

terminals, (respectively springs, sliders and dashpots) (Figure 23.1b). In terms of GP, the function set includes the serial and parallel connectors, both of variable arity $n \geq 2$, involved in rheological models. The terminal set includes three elements, all involving a floating-point coefficient: a spring is characterized by its stiffness, a slider, by its stress threshold, and a dashpot, by its viscosity. A terminal thus consists of a pair (*type,coefficient*), where *type* is 3-valued and *coefficient* is a positive floating-point value.

23.2.2 Fitness computation

A rheological tree ultimately encodes a program modeling the mechanical behavior of the material. This program takes as input a loading history, and outputs the corresponding response of the model.

The execution of this program is a two-step process: the rheological tree is first *interpreted* as a set of partial differential equations. Then, this set of equations plus the equations describing some loading history, is *solved numerically*, and gives the response predicted by the model. Last, the fitness is computed as the distance between predicted and experimental responses, summed over several experiments. These experiments typically involve traction-compression for a given rate of deformation, creep (for a prescribed stress) and relaxation (for a prescribed strain).

23.2.2.1 The interpreter

The interpreter associates with any rheological tree R a set of partial differential equations. Let $\epsilon_i(t)$ and $\sigma_i(t)$ denote the local strain and stress (real-valued functions) attached to node $i$of R. The equation attached to a terminal i is:

- The elasticity equation for a spring with stiffness k_i : $\quad \sigma_i(t) = k_i \epsilon_i(t).$
- The viscosity equation for a dashpot with viscosity η_i : $\quad \sigma_i(t) = \eta_i \dot{\epsilon}_i(t).$
- The plasticity equation for a slider with stress threshold σ_i^s, which involves two alternative modes: either the current stress is less than the threshold σ_i^s, in which case the strain derivative remains at zero, or the stress remains at σ_i^s in which case the strain is undefined:

$$|\sigma_i(t)| < \sigma_i^s \text{ AND } \dot{\epsilon}_i(t) = 0 \quad OR \quad |\sigma_i(t)| = \sigma_i^s \text{ AND } SGN(\sigma_i(t)).\dot{\epsilon}_i(t) \geq 0.$$

The equation associated to a function node p links the stress and strain in the parent node p to the stresses and strains in the *active* children nodes $c_1, c_2, \ldots, c_n$:

- For a serial node, $\left\{ \begin{array}{lllllll} \epsilon_p(t) & = & \epsilon_{c_1}(t) + \epsilon_{c_2}(t) + \ldots + \epsilon_{c_n}(t). \\ \sigma_p(t) & = & \sigma_{c_1}(t) & = & \sigma_{c_2}(t) & = & \ldots & = & \sigma_{c_n}(t). \end{array} \right.$
- For a parallel node, $\left\{ \begin{array}{lllllll} \epsilon_p(t) & = & \epsilon_{c_1}(t) & = & \epsilon_{c_2}(t) & = & \ldots & = & \epsilon_{c_n}(t). \\ \sigma_p(t) & = & \sigma_{c_1}(t) + \sigma_{c_2}(t) + \ldots + \sigma_{c_n}(t). \end{array} \right.$

The state of the model, described through $(\sigma_i(t), \epsilon_i(t), \dot{\epsilon}_i(t))_{i=0}^N$, where N is the total number of nodes in the tree, is governed by the set of partial differential equations attached to the *active* nodes in R, noted $\mathcal{E}_R$.

Restriction on the search space: The main difficulty in this step is that the activity of nodes depends on the loading history: as long as a slider i is not saturated

($|\sigma_i(t)| < \sigma_i^s$), it inhibits (makes inactive) all branches parallel to it and all branches in serial below it. The interpreter normally proceeds by recursively checking the rheological tree (given the underlying semantics of parallel and serial assemblies). This parsing is simplified via a syntactic restriction on the search space, or *language bias*; the children of a *parallel* (respectively *serial*) node are stipulated to be either terminals, or *serial* (resp. *parallel*) nodes.

Note that this language bias significantly reduces the amount of degeneracy of the representation, i.e. the number of genotypes (rheological tree) standing for a phenotype (mechanical behavior). Different aspects of degeneracy are discussed in the literature: degeneracy is blamed for misleading the search [Radcliffe and Surry 1994a]; but special cases of degeneracy, such as induced by introns, are shown beneficial in limiting the disruptive effects of crossover [Levenick 1991], or allowing recessive information (not considered for fitness computation) to be transmitted [Koza 1992].

Nevertheless, there is still room for degeneracy in the so-biased language: reductions similar to arithmetic simplification would be possible (e.g. spring k_1 // spring $k_2 \equiv$ spring $k_1 + k_2$), and sliders may cause parts of the tree to become inactive, i.e. recessive.

23.2.2.2 The solver

The next step consists of computing the response of the model when external dynamical load is applied. This load is described as the sequence of strains applied at instants $t_0, \ldots, t_T$, noted $\epsilon_{exp}(t_j), j = 0, \ldots, T$. The global stress $\sigma_0(t)$ attached to the root of the tree is incrementally computed for any t_j:

1. Derivatives are expressed via finite differences: $\dot{\epsilon}_i(t_j) = \frac{\epsilon_i(t_j) - \epsilon_i(t_{j-1})}{t_j - t_{j-1}}$.

2. The loading history is taken into account: $\epsilon_0(t_j) = \epsilon_{exp}(t_j)$ and $\dot{\epsilon}_0(t_j) = \dot{\epsilon}_{exp}(t_j)$.

3. These equations, together with $\mathcal{E}_R$, are handled as a set of linear equations in the unknown $\sigma_i(t_j)$, $\epsilon_i(t_j)$, $\dot{\epsilon}_i(t_j)$ and $\epsilon_i(t_{j-1})$.

4. Solving this set of linear equations gives the state of the model at time t_j, given its previous state at time t_{j-1}, the initial state of the model being $(0, 0, \ldots, 0)$.

5. For each slider i, the internal stress $\sigma_i(t_j)$ is compared to the threshold σ_i^s; whenever a slider becomes saturated ($|\sigma_i(t_j)| \geq \sigma_i^s$) or leaves saturation ($|\sigma_i(t_j)| < \sigma_i^s$), the system $\mathcal{E}_R$ is rebuilt by running the interpreter.

6. The response $\sigma_0(t)$ for loading $\epsilon_{exp}(t_j), j = 0, \ldots, T$, is noted $\sigma_R(t, \epsilon_{exp}, t_0, \ldots, t_T)$.

Complexity: The complexity of this resolution amounts to $T \times 2(3N)^3/3$, where T is the number of time steps of the loading history and N is the size of the tree (the resolution of a linear system of size n being $2n^3/3$).

Error: This resolution process involves two kinds of error. First, the mechanical measures are known with a given precision at given instants only; and the discretization of the loading history is beyond our control. Second our handling of derivatives induces numerical errors depending on the current model. We heuristically propose to estimate both kinds of errors *for a given model*, through the difference between the response computed from the whole loading history $\epsilon_{exp}(t), t = t_0, \ldots, t_T$, and the response computed from an excerpt of the same loading history, including only one in two consecutive instants (a loading history typically includes several dozens or a few hundred points):

$$Error = \sum_{t=t_0}^{t_T} |\sigma_R(t, \epsilon_{exp}, t_0, t_1, t_2, \ldots, t_T) - \sigma_R(t, \epsilon_{exp}, t_0, t_2, t_4, \ldots, t_T)|.$$

This estimate intends to capture both the imprecision inherent in the available data, and the error introduced by the solver. A run is considered successful if the fitness of the best model falls below this estimate of the unavoidable error.

23.2.2.3 Fitness computation

A fitness case is an experimental curve involving the loading history $\epsilon_{exp}(t)$ and the observed stress $\sigma_{exp}(t)$, for $t = t_0, \ldots, t_T$. A model R is evaluated through the difference between the observed stress $\sigma_{exp}(t)$ and the stress $\sigma_R(t, \epsilon_{exp})$ computed from the model R according to the experimental loading history:

$$F(R, (\epsilon_{exp}(t), \sigma_{exp}(t))) = \sum_{t=t_0}^{t_T} |\sigma_R(t, \epsilon_{exp}) - \sigma_{exp}(t)|.$$

The total fitness associated to a model is the sum of the errors on all fitness cases, i.e. on the available mechanical experiments (typically three).

As usual, most of the GP computational effort is spent in calculating the fitness. And, since tournament selection [Goldberg and Deb 1991] is used, *the evolutionary process is only dependent on the ordering of fitnesses* [Andrews and Prager 1994]; this suggests several heuristics that could lessen the global computational cost, at least in the earlier stages of evolution:

- Gradually increase the number of experiments to fit: evolution typically starts by considering a unique fitness case, and additional fitness cases are gradually taken into account when the current population meets some criterion, to be defined.

- Gradually increase the number of time-steps taken into account: evolution starts by considering only the first K time-steps of any loading history, and K is similarly incremented when the current population meets some criterion.

- Gradually increase the precision of loading histories: only K evenly spaced time-steps are first considered, with K increasing with the number of generations.

In all cases, the strategy consists of successively optimizing more and more precise approximations of the actual fitness. A similar iterative strategy proved useful for constrained optimization [Schoenauer and Xanthakis 1993]: first stages are concerned with sampling the feasible region. Only then, does evolution deal with the actual optimization problem. However, the successive optimization scheme addresses a basic need for constrained optimization, while here it aims at computational savings. Moreover, our timing of the fitness sequence is more critical, because there is no guarantee as to the smooth convergence of the intermediate fitnesses toward the actual fitness. Failing to change the biased approximation of fitness frequently enough may result in overfitting its bias; evolution then may get stuck in a local optimum.

The change of the fitness function empirically takes place when the current best fitness falls below 75% of the initial best value for the current fitness function.

23.2.3 Evolution operators

The two standard genetic operators, crossover by random swapping of subtrees and mutation by random replacement of a subtree [Koza 1992], are modified for offsprings to meet the syntactic restrictions described in section 23.2.2.2.

Several factors in our problem indicated that a third operator might be useful. Like numerical functions, rheological assemblies allow for values to combine (e.g. the series of two springs with respective stiffness k_1 and k_2 behaves as a spring with stiffness $\frac{k_1 * k_2}{k_1 + k_2}$). The adjustment of numerical values could then be left to random mutation and crossover, as in [Koza 1992]. However, this leads to a dramatic increase of the size of the trees along evolution; and the fitness computation increases as the cube of the size (section 23.2.2.2). Such adjustment of constants is therefore much more expensive for rheological identification than for classical regression problems. A specific operator was thus devised to address the optimization of floating-points values. This operator basically is a random hill-climber, involving an elementary evolution operator termed *surface mutation*.

A surface mutation modifies all floating-point terminals in a given model, via the addition of a Gaussian noise; the variance of the mutation is attached to the coefficient and recombined as in evolution strategies [Schwefel 1981]. A surface mutation is successful if it results in an improved fitness. The random hill-climber repeatedly performs surface mutations, until a number O of consecutive surface mutations is found unsuccessful. The influence of parameter O, termed *stubbornness*, on the dynamics of evolution is discussed in next section.

Any crossover or (classical) mutation is followed by a hill-climbing stage, in order to get the locally best version of the new incoming tree-structure. This strategy introduces into the field of GP the formal memetic approach developed in the frame of GAs in [Radcliffe and Surry 1994b]: the aim still is to confine the genetic population in the region of local optima with respect to the floating-point values.

23.2.4 First results

The aim of GP is to find rheological models fitting the set of available experiments. The GP environment itself was written from scratch in C^{++}: Available GP packages (including GPQuick used in section 23.3.5) required too many modifications, and there is no need for efficient tree evaluation, as each tree needs to be interpreted only once per fitness computation (see 23.2.2). The typical time for a run is about 3 hours on a HP 710 workstation. Table 23.1 below summarizes the different parameters of this implementation. All results are averaged over 10 independent runs.

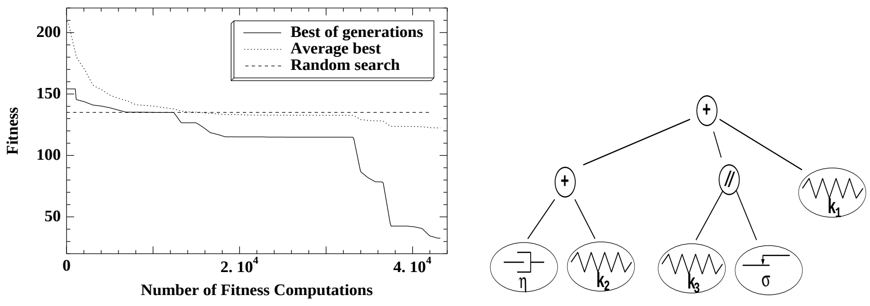
In order to ease further comparisons, the results presented here are concerned with identifying a known rheological model, which represents a hand-crafted model of polyethylene [Kichenin 1992] (Figure 23.1). Parametric optimization was used to adjust the numerical coefficients (k_1 = 790.45, k_2 = 150.20, k_3 = 41.60, η = 6248.60, σ_S = 7.25) from mechanical tests. The fitness cases consider the actual behavior of this model in the same experimental conditions, simulated from a regular PDE solver.

The dynamics over the course of evolution are plotted as the best fitness averaged

Table 23.1

Tableau for rheological model identification.

Objective:	Models fitting the available experiments
Terminal set:	Spring, Slider, Dashpot
Function set:	Parallel, Series
Fitness cases:	Loading histories
Raw Fitness:	Sum of errors over fitness cases
Evolution scheme	Generational (offsprings replace parents)
Parameters	Population size: 200
	Crossover rate = .4 and mutation rate = .2
Selection method	Tournament (3)
Wrapper	Specific Interpreter and Solver
Termination criterion	Final generation reached
Success measure	Fitness less than unavoidable computational error (see 23.2.2.2)



(a) (b)

Figure 23.2

Typical results on the identification of rheological model. (a) Dynamics of Evolution. (b) Overall best result, to be compared to Figure 23.1b.

over 10 independent runs obtained for a number of fitness computations; the best of all generations is also indicated. The horizontal line gives as reference the average best result obtained by blind random search for the same total number of trials.

Typical results are shown in Figure 23.2, obtained for 200 individuals with a random hill climber of stubbornness 2, without approximations of the fitness. The rate of success is 20%; the best individual so far is shown in Figure 23.2b, with coefficients $k_1 = 998.892$, $k_2 = 133.085$, $k_3 = 39.6647$, $\eta = 8698.78$, $\sigma_S = 19.0409$. According to experts, the difference with the model of Figure 23.1b comes from the absence of creep in all considered experiments.

We are particularly interested in the impact of the random hill-climber (23.2.3) and of the iterated fitness scheme (23.2.2.3), on the dynamics of evolution.

A first key issue deals with the RHC stubbornness O. Only small values of O were found profitable (see Figure 23.3a): a thorough optimization of coefficients in the first stages of evolution is observed to mislead the search (curve $O = 3$); but a limited optimization ($O = 2$) leads to better results than standard evolution ($O = 1$). However, the level of improvement decreases when the size of the population increases (e.g. $O = 1$ seems the best choice with 500 individuals; but better overall results were obtained with 200 than with 500 individuals).

Another important issue is the effect of the gradual refinement of the fitnesses. Gradually increasing the number of experiments taken into account was found to be prejudicial: after several generations have considered one fitness case only, the population gets trapped in local optima. Gradually increasing the size of truncated fitness cases (involving the first K time steps in a loading history) was found similarly prejudicial: considering only the beginning of the experimental curves results in a population of mostly linear trees and leads to premature convergence.

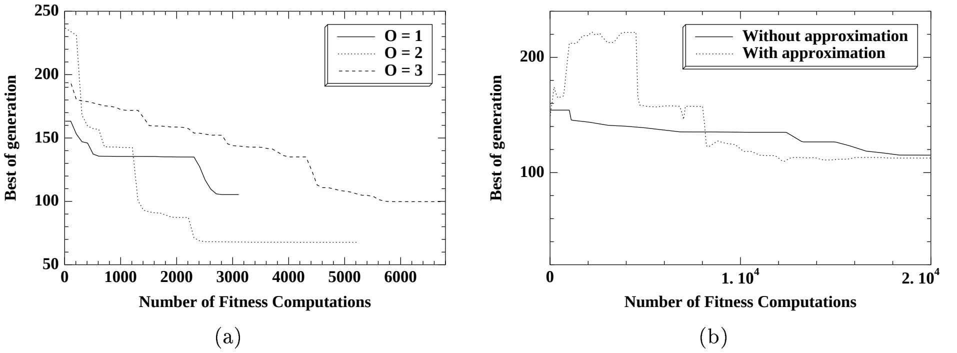


Figure 23.3

Comparative results on the rheological model identification problem. (a) Influence of the Random Hill-Climber. (b) Influence of the approximation scheme.

Last, gradually increasing the precision of the discretization of the fitness cases (i.e. increasing the number K of time steps) is found to be helpful, especially for small populations. Considering a given precision for too long is prejudicial; but starting with a low precision seems to guide the population toward promising regions.

Figure 23.3b shows the dynamics obtained for the three-phase scheme:

• The fitness is first computed with precision 1/3 (only one in three consecutive time steps is considered), until the current best fitness falls below the 75% of the initial best fitness,

• The fitness is then computed with precision 1/2 (only one in two consecutive time-steps is considered), until again the current best fitness falls below 75% of the initial best fitness using this precision,

• Last, the fitness is computed with full precision.

The dynamics of such evolution needs some corrections in order to perform comparisons. First, the difference in computational time is accounted for in the number of fitness computations. Second, the best performance of a population is plotted as the actual fitness (with full precision) of the best individual with respect to the current fitness. This performance is not that of the best individual of the population with respect to the actual fitness, and it occasionally decreases during first and second phases (Figure 23.3b).

23.2.5 Further work

A next step, from a mechanical point of view, is concerned with improving the readability of the result, through syntactic simplifications of the produced trees. Such simplifications could intervene as a new evolution operator: this could, as in the case of classical regression, limit the growth of trees as evolution goes on.

The most exciting developments for the GP community are connected to the structure of the fitness function. Rather than a sum, it could be viewed as a multi-objective function by separately considering the partial fitnesses derived from the fitness cases. This leads to characterize the set of optimal solutions in the sense of Pareto (solutions not dominated on at least one objective). In the same perspective, a new crossover operator could be designed for multi-objective optimization. For instance, the selection-seduction scheme [Ronald 1995] could be turned into a "reciprocal-seduction" scheme, where the reciprocal seduction of any two individuals depends on their complementarity.

23.3 Three-dimensional hyperelastic materials

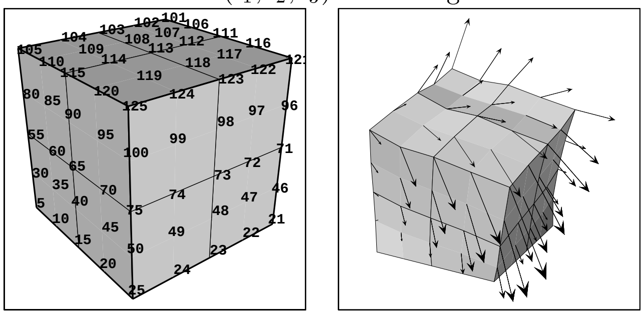
23.3.1 The mechanical model

We restrict the discussion in this section to hyperelastic isotropic materials, e.g. rubber. A material is said to be *elastic* if its deformation under external loads is reversible, and if its equilibrium state does not depend on its history.

The central problem in nonlinear three-dimensional elasticity [Green and Zerna 1968] consists of finding the equilibrium position of an elastic body that occupies a reference configuration $\overline{\Omega}$ in the absence of applied forces. When subjected to external loads, the deformed configuration is characterized by its *deformation*, a mapping $\varphi : \overline{\Omega} \longrightarrow \mathbb{R}^3$. The equation of equilibrium forms a boundary problem [Ciarlet 1988] in terms of φ, $T(x, \nabla\varphi)$ (the first Piola-Kirchhoff stress tensor of the real positive 3×3 matrix $\nabla\varphi$), the density of the applied body forces per unit volume, the density of the applied surface forces per unit area on some part of the boundary Γ of Ω, and a given deformation on the other part of Γ.

A material is said to be *hyperelastic* if there exists a potential law W, named the *strain energy function*, such that $T(x, M) = \frac{\partial W}{\partial M}(x, M)$ for all $x \in \overline{\Omega}$ and all real positive 3×3 matrices M. Furthermore, if the material is *isotropic*, this potential W can in fact be written as a function of $\{\iota_k(\nabla\varphi^T\nabla\varphi)\}_{1\leq k\leq 3}$, the three principal invariants of the (positive definite) matrix $\nabla\varphi^T\nabla\varphi$, namely $\iota_1 = \mathrm{tr}\,(\nabla\varphi^T\nabla\varphi)$, $\iota_2 = \det(\nabla\varphi^T\nabla\varphi)\,\mathrm{tr}\,(\nabla\varphi^T\nabla\varphi)^{-1}$ and $\iota_3 = \det(\nabla\varphi^T\nabla\varphi)$.

This function $W(\iota_1, \iota_2, \iota_3)$ is the target of the identification process.



(a) (b)

Figure 23.4

(a) The initial state of the structure, with the $5 \times 5 \times 5$ nodes of the $2 \times 2 \times 2$ mesh. (b) The experimental displacements under oblique traction on the upper side.

23.3.2 The numerical model

For a given valid (see section 23.3.6.2) strain energy function $W(\iota_1, \iota_2, \iota_3)$, the behavior of a structure submitted to boundary conditions and a given load can be numerically computed using the finite element method (FEM). The structure is discretized into *elements* and the system of PDEs is discretized, giving a finite system in terms of the values of the unknowns at some discrete points, i.e. the *nodes* of the mesh. The result is a discrete system of PDEs, that can be solved using iterative numerical methods.

We consider the $[0, 1]^3$ cube (all physical measures are given in standard units), a regular cubic mesh is used. Each element has 27 nodes (9 nodes per face, and a node at center of the element), to allow an interpolation of degree 2. Figure 23.4a illustrates such a $2 \times 2 \times 2$ mesh, and the resulting system has 3×125 unknown variables (3 displacements per node). The discrete system is solved by Newton's method, which implies the computation of the derivative matrix of the system and its inversion, for each nonlinear iteration. The complexity of one finite element analysis thus depends on the size of the discrete system (number of elements $\times$ number of degrees of freedom per element) and on the number of iterations to solve each nonlinear system (which can only be upper bounded). It does happen that Newton's method fails to converge, and such a situation can only be detected after the maximal number of iterations have been run. The complexity is maximal when no meaningful solution can be derived.

The direct problem is: *For each loading case, the corresponding response of the structure, in term of displacement (or strain or stress if needed) is given by a finite element analysis,* **provided the energy function is given**.

23.3.3 The inverse problem

We are now interested in the inverse problem: *From experimental responses (displacement, strain or stress) of a structure under known loads, find the strain energy function of the structure's material.*

A mechanical experiment is performed for each loading case L_i, $i \in \{1, \ldots, N_L\}$, and the corresponding displacement fields $\vec{D}_i^{exp}$, $i \in \{1, \ldots, N_L\}$, are recorded.

For any function $W(\iota_1, \iota_2, \iota_3)$, the solution of the direct problem gives the displacement fields $\vec{D}_i^W$, $i \in \{1, \ldots, N_L\}$ of the structure made of a material that would have W as its strain energy function. The solution to the identification problem is thus the function W that minimizes the mean square error $\sum_1^{N_L} ||\vec{D}_i^W - \vec{D}_i^{exp}||^2$.

23.3.4 Why GP?

A fundamental issue in function identification lies in the model chosen for the function to identify. If a parametric model is known, the identification then reduces to parameter optimization. However, when no specific model has been identified, a variety of general models can be used. The quasi-parametric models of polynomials, rational fractions or any other nonlinear regression models are *a priori* discarded. This is because the precision of the result depends on the number of parameters, which has to be fixed: too low, no accurate solution exists in the search space, too high, the complexity of the parametric optimization gets too large.

Two other widely used non-parametric identification tools are neural networks [Rumelhart and McClelland 1986] and genetic programming [Koza 1992]. In both models, the complexity of the solution is identified together with the parameters. Note that both feed-forward and recurrent neural networks have proven to be robust identification tools, in situations where no other approach succeeded (e.g. [Yao 1993; Fadda and Schoenauer 1995]).

But a characteristic of the numerical modeling problem for mechanical structures described in section 23.3.2 is that the finite element analysis implies the computation of numerical values for first- and second-order derivatives of the strain energy function.

• If the chosen model only allows the computation of values of the function itself, numerical derivation is the only way to obtain derivative values. But the instability and inaccuracy of numerical derivation forbids its use to compute second order derivatives.

• Direct computation of derivative values is impossible for recurrent neural networks in which output values are given as a fixed point of an iterated process, and barely tractable for feed-forward neural networks.

Before addressing the identification problem, we describe in next section our GP environment, and discuss two important issues: the adjustment of floating-point terminals and the influence of using derivatives in the fitness computation.

23.3.5 The GP environment

The basis of our GP package is A. Singleton's GPQuick (in C++), tailored to handle floating-point terminals and to compute derivatives. Throughout the end of the chapter, the function set is made of the standard binary operations $\{+,-,*\}$ and the terminal set consists of the variables for the problem, plus the floating-point random terminal $\mathcal{R}$ (to be discussed in next subsection).

The evolution scheme is a combination of steady state and generational: copies of the 30% of the population, chosen using a tournament of size four, are altered using the genetic operators. The replacement of the resulting trees is then performed by repeated tournaments of size two among the initial population.

The genetic operators are crossover, standard mutations (change of function, insertion of a random subtree or promotion of a subtree), plus Gaussian mutation to adjust the floating-point terminals. Rather high values for the mutation rates were found to be necessary, probably due to the small sizes of population used.

23.3.5.1 Handling floating-point terminals

In past GP research, no great attention was paid to the floating-point terminals. In most cases, as in [Koza 1992], discretized versions of floating-point terminals are used, and their adjustment to precise values is left to the combination of functions

Table 23.2

Tableau for the symbolic regression experiments

Objective:	Symbolic regression on polynomial $P(x, y) = 2.5 * x^2 + 1.2 * y^2 - 1.8 * x + 0.6y$
Terminal set:	x, y, $\mathcal{R}$
Function set:	+, -, *
Fitness cases:	20 points (x, y, P(x,y))
Raw Fitness:	Mean square error on the fitness cases
Fitness:	$1/(1 + rawfitness)$
Population size:	300 (30% generate offsprings)
Operator rates	Crossover: 0.3 – Mutation: 0.5 – Copy: 0.2
Selection method	Tournament (4)
Replacement method	Death Tournament (2)
Termination criterion	Final generation reached

or to random mutations. In [Iba, de Garis and Sato 1994], the difficulty is tackled using a regression method, which is workable only on data fitting problems. We feel that, at least for numerical applications, floating-point terminals must be given a better chance to precisely adjust. The modifications to the original GPQuick are:

Initialization: The precision of floating-point terminals is the machine precision. They are chosen randomly in $[10^{-3}, 10^3]$.

Evolution: Two new mutation operators are allowed:

• A mutation by addition of a random Gaussian variable, as first used in evolution strategies [Schwefel 1981]. The average of the number of terminals of a given tree that undergo this Gaussian mutation is fixed (five in our experiments).

• A random hill-climber resembling that described in section 23.2.3, with the difference that this hill-climber is used as an alternative mutation operator rather than being systematically applied after any other operator.

Comparative results: Comparative experiments were performed on a symbolic regression problem with two variables (see Table 23.2), involving three different methods to handle floating-point terminals: the standard evolution by GP crossover and mutation, excluding Gaussian mutations of floating-point terminals, was compared to the two mutation operators described above.

The best results (Figure 23.5) are obtained using the Gaussian mutation operator. The gradient-like hill-climbing strategy performs quite poorly indeed, especially in the early generations. It does end up a little higher than the Gaussian mutation, but these plots do not account for the computational overhead.

The other point is that the GP-style method performs quite well, only slightly

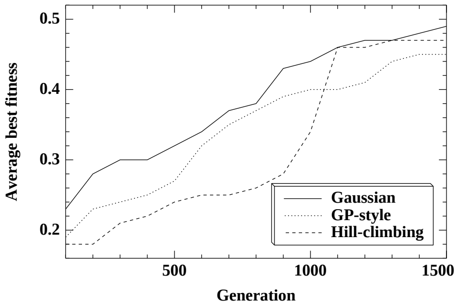


Figure 23.5

Best of generation results of ES-style, GP-style or gradient-like floating-point terminals handling.

worse than the Gaussian mutation. However, its major drawback is a tremendous increase of the average structural complexity of the trees. As in Section 23.2.3, this greatly increases the computational complexity of one fitness computation. So all further experiments reported in this chapter use Gaussian mutation.

The results of Figure 23.5 witness that GP, as well as GAs [Goldberg 1989] can suffer from too much determinism: optimizing the floating-point terminals of raw tree structures does not help. On the opposite, when partial convergence has begun, such technique can speed up the evolution, suggesting a mixed technique that could choose between both strategies adaptively along generations as done in [Spears 1995, Julstrom 1995].

23.3.5.2 Using derivatives

The most specific aspect of this application, from a GP point of view, is the use of derivatives during the computation of the fitness. The computation of the derived trees are straightforward, but the complexity of the derived trees is significantly larger than that of the initial tree.

It is clear from the function set used here ($\{+, -, *\}$) that the raw size of a derivate tree is increased by a factor depending on the complexity of the initial tree and on the number of times function $*$ appears. Some trivial simplifications can easily be implemented, (e.g. $\frac{\partial}{\partial x}(\text{Constant terminal}) \equiv 0$). But our experiments suggest the increase for the size of the derivate tree compared to that of the initial tree ranges from around 1.5 times for simple trees (in the early stages of the runs) to up to more than 3 times for the usually more complex trees appearing in the end of the runs.

23.3.6 Strain energy function identification

We now come back to the problem of identifying the strain energy function of some hyperelastic material from mechanical experiments (section 23.3.3).

23.3.6.1 Fitness computation

The geometry of the structures is shown in Figure 23.4a, together with the very coarse mesh used through all experiments. The structure is fixed below, and the loading cases consist of forces applied on the whole upper surface, with different directions. The experimental displacement fields are, at the moment, results of numerical simulations using a known strain energy function. The influence of the computational error thus vanishes, and the best solution is known.

The computation of one fitness case amounts to between 12 and 50 (number of Newton's nonlinear iterations) $\times$ 8 (number of elements of the mesh) $\times$ 27 (number of integration-points per element) computations of the outputs of the 3 first-order and the 6 second-order derivate trees, plus one numerical solution of the 378 by 378 linear system and one comparison with the experimental displacements.

For an average number of nonlinear iterations of 25, with 12 loading cases (see section 23.3.6.3), the time consuming part of a fitness evaluation lies in the execution of the GP trees (around 35000 per fitness computation). As these trees are derivate trees, their average complexity, compared to that of symbolic regression trees is around 3 for first-order trees and 9 for second-order trees (see section 23.3.5.2). So the 35000 derivate-tree evaluations actually amount to about 250000 standard tree evaluations. But the most important point is that the maximal computational cost is reached *when Newton's method fails to converge*: such failure can only be detected after the maximal number of iterations (50) has been reached. The next section addresses this difficulty by trying to *a priori* detect such situations.

23.3.6.2 Constraints on the strain energy function

The very first attempts proved to be complete failures. The reason for that is fairly simple: only around 2% of random trees actually are valid strain energy functions. Using a random function as a strain energy function leads in 98% of the cases to non-convergent iterations (best case) or to unpredictable abortion (worst case) of the finite element analysis.

A closer look at hyperelastic materials theory shows that there are many necessary conditions energy strain functions must satisfy:

- The limit when any of the three variables goes to infinity must be $+\infty$;
- The limit when the third variable ι_3 goes to 0 must be $+\infty$;
- When the structure is in its initial state ($\varphi(x) = x$, i.e. $\nabla\varphi = \mathbf{1}$ and $(\iota_1, \iota_2, \iota_3) = (3, 3, 1)$), the following *no-loading condition* holds:

$$\frac{\partial W}{\partial \iota_1} + 2\frac{\partial W}{\partial \iota_2} + \frac{\partial W}{\partial \iota_3} = 0.$$

The *a priori* restriction of the search space to match these conditions is impossible. But the first two limit conditions can be checked *a posteriori*: the price to pay is that of a few computations of W with at least one variable taking large values. During the initialization of the population, trees failing the test are discarded, and after crossover or mutation, offspring that fail are assigned zero fitness.

Taking the third condition into account is achieved by mechanical means: *Ogden materials* [Ogden 1984] are defined by a strain energy function of the form

$$W(\iota_1, \iota_2, \iota_3) = W_g(\iota_1, \iota_2, \iota_3) - \alpha \log(\iota_3),$$

for any function W_g and positive α. The behavior of almost all known hyperelastic materials can be approximated using some Ogden law. The idea is to use GP to identify W_g, W being defined by the above Ogden equation. Parameter α is computed *a posteriori* such that the *no-loading* condition holds.

23.3.6.3 The fitness cases

The "experimental" data are built using the simplest form of Ogden materials, Mooney-Rivlin materials [Ciarlet 1988] where the W_g function to identify is linear:

$$W_g(\iota_1, \iota_2, \iota_3) = a\iota_1 + b\iota_2 + c\iota_3$$

Three basic loading cases are used throughout the experiments. In each loading case, forces are applied to the upper boundary of the cube structure. Loading #1 is made of vertical forces. Figure 23.4b plots the displacement fields of the structure submitted to loading case #2 (oblique forces), inducing traction and compression in the structure. Loading case #3 resembles loading case #2, with different direction of forces inducing torsion.

23.3.6.4 First results

Experimental settings are summarized in Table 23.3. The first results dealt with Mooney-Rivlin materials with $a = b = c = 1$. Such a problem was far too easy: the exact solution was found every time in less than 5 generations by both GP and blind random search. So the Mooney-Rivlin energy function of the "experimental" data for all following experiments uses values $1.1, 1.2, 1.3$ for a, b, c.

The choice of strength for the loading cases raises the next concern: random strain energy functions of Ogden materials, although satisfying the necessary conditions listed in section 23.3.6.2, can model materials with weird behaviors, ending in stiff, highly nonlinear systems. And the Newton method used to solve the nonlinear system is not robust enough to still be able to converge. This implies these energy functions get null fitness, and thus will likely be eliminated by the selection pressure. But unfortunately, they represent a large majority of the initial populations of any GP run. Therefore, it is essential to be able to assign *partial credit* (as emphasized in [Kinnear 1994]) to these early solutions.

Our approach for overcoming this difficulty is that any viable strain energy function behaves smoothly around the initial position, i.e. for very small loads. But on the other hand, very small loadings give rise to quasi-linear behaviors, whatever the energy function. Under very small loads, any function whose linearization around the no-load point $(3, 3, 1)$ resemble the initial Mooney-Rivlin function gets a low error on the displacements. For instance, for a force of intensity 10^{-3}, the solutions found by GP, achieving very low error on the fitness cases, are useless to predict

Table 23.3

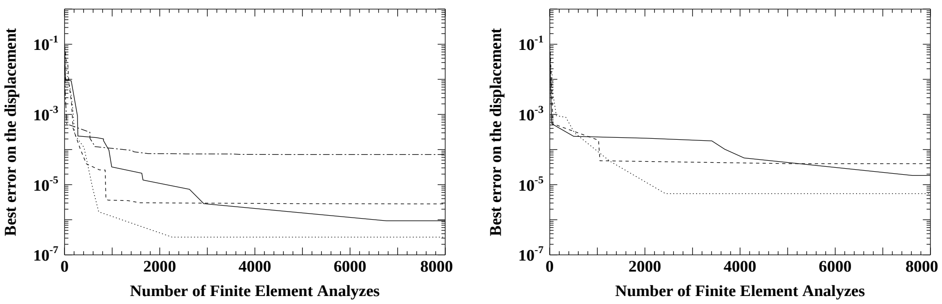
Tableau for the strain energy function identification

Objective:	Energy function fitting the experiments
Terminal set:	x, y, $\mathcal{R}$
Function set:	+, -, *
Fitness cases:	Three basic loading cases (different directions of forces)
Fitness:	Mean square error on the fitness cases
Parameters	Population size: 100
Operator rates	Crossover: 0.3 – Mutation: 0.5 – Copy: 0.2
Selection method	Tournament (2)
Replacement method	Generational
Wrapper	Compute α to meet the *no-load* condition.
Termination criterion	Final generation reached

the behavior of the structure for large loads.

Therefore not significant loads are necessary in order to distinguish between different energy functions having the same linearized behavior. Figure 23.6a presents the first successful results in this respect, obtained with forces intensity of $8\ 10^{-2}N$. The maximal displacement of the structure is around 0.5.

The results are at the moment limited to four runs of GP (Figure 23.6a). They are to be compared to results of Figure 23.6b, obtained by three runs of a stochastic iterated hill-climber (SIHC). During an elementary hill-climbing step, an individual undergoes mutation, being replaced by any fitter offspring. It ends after a prescribed (20) number of unsuccessful mutations. SIHC repeatedly performs such elementary steps, starting from randomly chosen individuals (and one "run" is therefore an artificial sequential grouping of some of these hill-climbing steps). For both GP and SIHC, only the number of finite element analyses actually performed are recorded. Non viable trees are simply discarded.



(a) (b)

Figure 23.6

Comparative results between GP and a stochastic random hill-climber. (a) Four GP runs. Best errors: $3.2\,10^{-7}$, $9.33\,10^{-7}$, $2.83\,10^{-6}$ and $7.22\,10^{-5}$. (b) 24000 SIHC fitness computations. Best overall error: $5.55\,10^{-6}$.

These curves clearly indicate that GP outperforms the blind local optimization approach in terms of error on the displacements. Moreover, the best three GP runs did find out the linear form of the W_g function, the overall best coefficients being $(1.278493549, 1, 1.527635828)$. We are aware that further experiments are needed to actually demonstrate the full power of this approach.

23.3.6.5 Discussion and Further Work

The number of fitness evaluations is quite limited in the results of section 23.3.6.4, due to the large computational complexity of the FEM. As quoted in section 23.3.6.1, this complexity depends not only on the mesh (chosen as coarse as possible) and on the structural complexity of the tree at hand, but also on the number of iterations of the nonlinear numerical method, which is greater for large than for small loads. To address both the problem of complexity related to load intensity and the issue of partial credit to initial trees, we propose to use an iterative scheme, modifying the fitness cases during evolution, in a similar way as for the one-dimensional rheological model identification (section 23.2.2.3). On-going experiments use force strengths gradually increasing from $10^{-3}N$ for the first generation to $10^{-1}N$.

Another on-going attempt to minimize the computational cost is to use symbolic computation to simplify the structural complexity of the derivative trees. Preliminary results in that direction suggest that a benefit can be expected. Moreover, a *post-mortem* optimization of the floating points terminals *of the simplified version* of the overall best tree would certainly improve the results.

23.4 Summary

This chapter demonstrates the basic feasibility in structural mechanics of a GP-based approach for the identification of both one-dimensional rheological models and three-dimensional hyperelastic strain energy functions. Moreover, the use of GP for these problems is fully justified: The structure of the solutions is unknown, and no other non parametric technique can apply.

From the point of view of mechanical modeling, this work opens many perspectives regarding the identification of materials from their experimental behavior, as they are insofar restricted to heuristics and specific studies. Another exciting perspective would be to ease the design of new materials with stipulated behaviors.

This study represents one of the few applications of GP to Applied Science [Oakley 1994]. Such cross-fertilization of Evolutionary Computation and Applied Science was made possible through a thorough collaboration between experts of both fields. The background knowledge in mechanics was essential in building the wrapper for the rheological model identification (section 23.2.2.2), and restricting the search space, in order for GP to explore only viable energy functions (section 23.3.6.2).

The most limiting factor of this approach is the computational complexity. We propose to gradually refine the fitness (sections 23.2.4 and 23.3.6.5). Preliminary results show that such an iterative scheme can significantly reduce the time-to-solution for the rheological model identification. However, more sophisticated switching strategies remain to be investigated in order to avoid population to get trapped in the local optima of intermediate fitnesses. Another critical issue deals with the precise adjustment of the floating-point values in numerical GP applications (sections 23.2.3 and 23.3.5.1).

As a conclusion, we feel that the unique ability for GP to explicitly deal with derivatives is one of its most appealing aspects for Engineering and Design problems. The presented work suggests that actual breakthroughs can result from a tight interaction between GPers and open-minded researchers and engineers in Applied Science.

Acknowledgments

Thanks to M. Bonnet (LMS–Ecole Polytechnique) for useful discussions, and to D. Andre, P. Angeline, J. Daida and J. Koza whose comments greatly improved the clarity and the readability of this chapter.

Bibliography

Andrews, M. and R. Prager (1994). Genetic Programming for the Acquisition of Double Auction Market Strategies. In *Advances in Genetic Programming*, K. E. Kinnear Jr (ed.), Cambridge: MIT Press, pp 355–368.

Ciarlet, P. G. (1988). *Mathematical Elasticity, Vol I: Three-Dimensional Elasticity.* Amsterdam: North-Holland.

Fadda, A. and M. Schoenauer (1995). Evolutionary chromatographic law identification by recurrent neural nets. In 5^{th} *Conference on Evolutionary Programming*, J. R. McDonnell, R. G. Reynolds and D. B. Fogel (eds.), Cambridge: MIT Press, pp 219-235.

Gittus, J. and J. Zarka, Eds (1985). *Modelling small deformations of polycrystals.* Elsevier Applied Sciences Publishers.

Goldberg, D. E. (1989). Zen and the art of genetic algorithms. In *Proceedings of the* 3^{rd} *International Conference on Genetic Algorithms*, Morgan Kauffmann, pp 80–85.

Goldberg, D. E. and K. Deb (1991). A comparative study of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, J. G. Rawlins (ed.), Morgan Kauffmann, pp 69–93.

Green, A.E. and W. Zerna (1968). *Theoretical elasticity.* University Press.

Iba, H. , H. de Garis, and T. Sato (1994). Genetic Programming Using a Minimum Description Length Principle. In *Advances in Genetic Programming*, K. E. Kinnear Jr (ed.), Cambridge: MIT Press, pp 265–284.

Julstrom, B. A. (1995). What have you done for me lately ? In *Proceedings of the* 6^{th} *ICGA*, L. Eshelman (ed.), San Francisco, CA: Morgan Kauffmann, pp 81–87.

Kinnear, K. E. Jr (1994). A perspective on Genetic Programming. In *Advances in Genetic Programming*, K. E. Kinnear Jr (ed.), Cambridge: MIT Press, pp 3–19.

Kichenin, J. (1992). *Comportement thermomécanique du polyéthylène. Application aux structures gazières.* PhD thesis, Ecole Polytechnique – Palaiseau – France.

Koza, J. R. (1992). *Genetic Programming: On the Prorgamming of Computers by means of Natural Evolution*, Cambridge: MIT Press.

Koza, J. R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*, Cambridge: MIT Press.

Lemaitre, J. and C. Chaboche (1985) *Mécanique des Matériaux Solides.* Paris: Dunod.

Levenick, J. R. (1991). Inserting introns improves genetic algorithm success rate: Taking a cue from biology. In *Proceedings of the* 4^{th} *ICGA*, pp 123–127.

Oakley, H. (1994). Two Scientific Applications of Genetic Programming. In *Advances in Genetic Programming*, K. E. Kinnear Jr (ed.), Cambridge: MIT Press, pp 369–389.

Ogden, R. W. (1984). *Non-linear elastic deformations*, Ellis Horwood Ltd.

Persoz, B. (1960). *Introduction à la rhéologie*, Dunod: Paris.

Radcliffe, N. J. and P. D. Surry (1994). Fitness variance of formae and performance prediction. In *Foundations of Genetic Algorithms 3*, D. Whitley and M. Vose (eds.), San Francisco, CA: Morgan Kauffmann, pp 51–72.

Radcliffe, N. J. (1994). Formal memetic algorithms. In *Evolutionary Computing: AISB Workshop*, T.C. Fogarthy (ed.), Springer Verlag, LNCS 865, p1–16.

Ronald, E. (1995). When selection meets seduction. In *Proceedings of the* 6^{th} *ICGA*, L. Eshelman (ed.), San Francisco, CA: Morgan Kauffmann, pp 167–173.

Rumelhart, D.E. and J.L. McClelland (1986). *Parallel Distributed Processing*, Cambridge: MIT Press.

Sanchez-Palencia, E. and A. Zaoui (1987). *Homogenization techniques for composite media.* Lecture Notes in Physics, Springer Verlag.

Schoenauer, M. and S. Xanthakis (1993). Constrained GA optimization. In *Proceedings of* 4^{th} *ICGA*, S. Forrest (ed.), San Mateo, CA: Morgan Kauffmann, pp 573–580.

Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models*, John Wiley & Sons.

Spears, W. M. (1995). Adapting crossover in a genetic algorithm. In *Proceedings of the* 5^{th} *Conference on Evolutionary Programming*, J. R. McDonnell, R. G. Reynolds and D. B. Fogel (eds.), Cambridge: MIT Press, pp 367-384.

Yao, X. (1993). A review of evolutionary artificial neural networks, *Int. Jour. of Intelligent Systems*, **8**, pp 539–567.

Zarka, J., J. Frelat, G. Inglebert, and P. Kasmai-Navidi (1988). *A new approach in inelastic analysis of structures*, Martinus Nijhoff Publishers.