Evolutionary Computation

and Applications

at Centre de Mathématiques Appliquées

de l'Ecole Polytechnique

Marc Schoenauer – CNRS

Nov. 1995 – Revised March 97

Contents

1 Introduction **4**

2 Evolutionary optimization: an overview **5**
2.1 The generic Evolutionary Algorithm 5
2.2 Standard Evolutionary Algorithms 6
2.2.1 Genetic Algorithms 6
2.2.2 Evolution Strategies 7
2.2.3 Evolutionary Programming 7
2.2.4 Genetic Programming 7
2.3 Modern Trends: Hybrid Methods 8
2.4 Comparisons 8
2.5 Theoretical results 10
2.6 Evolution of Evolutionary Computation at CMAP 10
2.6.1 The bitstring era 11
2.6.2 The time of multi-representation GAs 12
2.6.3 EC: the adult age 12

3 Function identification **14**
3.1 Overview 14
3.2 Inverse problems 15
3.3 Models and associated evolutionary algorithms 16
3.3.1 Artificial neurons 17
3.3.2 Feedforward neural networks 17
3.3.3 Recurrent neural networks 18
3.3.4 Genetic programming: an analytic model 19
3.4 Case studies 19
3.4.1 Control and motion planning problems 19
3.4.2 Analytic chromatography 23
3.4.3 Constitutive law of materials 25
3.5 Function identification problems: Discussion 27
3.5.1 Comparisons of the different models 27
3.5.2 The generalization issue 29
3.5.3 Complexity 29
3.5.4 Using actual experimental data 30
3.6 Function identification problems: Conclusion 31

4 Shape Identification **31**
4.1 The abstract problem 31
4.2 Representations for shapes 32
4.2.1 Bit arrays 32
4.2.2 Voronoï diagrams 33
4.2.3 H-representation 34
4.3 Two applications in Structural Mechanics 35
4.3.1 Topological Optimum Design 35

4.3.2 Inclusion identification . . . 37
4.4 Discussion and further directions . . . 38
4.4.1 Other representations . . . 38
4.4.2 Comparison of different representations . . . 39
4.4.3 Real-valued shapes . . . 40
4.4.4 Symmetry of representations and problems . . . 41
4.4.5 Identification problems: Shapes vs. functions . . . 42
4.5 Shape identification: conclusion . . . 42

5 When EC joins the tool box of experts 43
5.1 Optical multi-layer systems synthesis . . . 43
5.2 Direction finder optimization . . . 44
5.3 ATC related applications . . . 45

6 Behavioral Memory 47
6.1 Motivations . . . 47
6.2 Guidelines . . . 48
6.3 Constrained trajectory planning . . . 48
6.3.1 The problem . . . 48
6.3.2 An evolutionary approach . . . 49
6.3.3 The Behavioral Memory approach . . . 49
6.3.4 Related work . . . 50
6.4 Grey scale pattern matching . . . 50
6.5 Constraint handling . . . 51
6.6 Discussion and further work . . . 53

7 Control by inductive learning 53
7.1 Motivations . . . 54
7.2 Guidelines . . . 54
7.3 Building an explicit memory . . . 55
7.4 Memory-based strategies of control . . . 56
7.5 Results . . . 56
7.6 Discussion . . . 58
7.7 Further work . . . 58

8 Conclusion 59

1 Introduction

Evolutionary Computation (EC) is a recently emerged concept of Computer Science, which denotes stochastic algorithms that somehow crudely mimic the Darwinian natural evolution of a population. We shall focus in this dissertation on EC from the point of view of optimization.

It seems that EC, as a domain of Computer Sciences, is presently reaching a steady and more mature state, as witnessed by the existence of Evolutionary Computation Journal since 1993 and the 1st IEEE-International Conference on Evolutionary Computation in June 1994 in Orlando [118].

EC is made up of 4 main branches (by alphabetical order):

- Evolution Strategies, born in Germany in the 60's [138, 156], to deal with numerical optimization problems.
- Evolutionary Programming, a branch that appeared in California in the 60's as well, and was first applied on Finite State Machine ([59]).
- Genetic Algorithms, which appeared in Michigan in the late 60's [78], and were primarily designed to optimally solve sequential decision processes more than to perform function optimization [43].
- Genetic Programming [99], at first considered a subset of GAs, but now turning into a research field by itself (the first International GP Conference is to be held in June 96 in Stanford), addressing the challenging problem of employing evolution to *teach computers to do things without being explicitly programmed to do so.*

From the point of view of optimization, EC is a powerful stochastic zeroth order method (i.e. requiring only values of the function to optimize) that can find the global optimum of very rough functions. This allows EC to tackle optimization problems for which standard optimization methods (e.g. gradient-based algorithms) are not applicable. The price to pay is twofold: Firstly, because of its stochastic nature, EC does not offer any guarantee as to its convergence during a given run (the convergence results are weak convergence results); Furthermore, the computational cost of an EC run is very generally very high, and a large number of function evaluations must be performed for a satisfying result to be (hopefully) found. The first common-sense conclusion — unfortunately often forgotten — is that EC should not be used whenever some deterministic optimization method is applicable.

Two main issues will be emphasized here:

- The wide range of application domains where EC can bring a real breakthrough, as no other method is known to apply;
- The need for background knowledge in the design of the instance of evolutionary algorithm to use on a particular problem (choice of representation, fitness function and genetic operators). A careful addition of domain knowledge during these crucial steps can greatly limit the computational cost.

Section 2 presents the main branches of EC, introducing the historical trends and their present challenges. In that framework, the local history (i.e. at CMAP) is discussed through the evolution of the home-brewed EC software - which has been used for all work presented thereafter.

The next three sections are application sections: Section 3 is concerned with function identification by EC: it details both the models for the functions to identify, and the representation used for each model. From the point of view of representation, the Shape Identification domain (section 4) is probably the richer, as no obvious or standard representation exists for shapes (be they in 1, 2 or 3 dimensions). The last application section presents various applications: Interferometer optimization, optical coating synthesis, Air Traffic Control problems (section 5).

Finally, no one can resist the temptation of trying to make the algorithm he or she is using work better. Section 6 and 7 are devoted to the presentation of two successful attempts. The first one (Iterated GAs), clearly comes from the numerical background and is one of our main arguments for using population based methods rather than other stochastic methods like SA or taboo search. On the opposite, the control of evolutionary operators by inductive learning obviously comes from the Machine Learning part of our scientific environment. Nevertheless, both modifications of standard EC algorithms are an example of how hybridization of methods coming from different fields can cross-fertilize scientific research.

The last section is entitled *Summary and Future Work*. But as most future work will have been discussed all along this dissertation, right after the work it will pursue, only general issues will be approached there.

2 Evolutionary optimization: an overview

For the sake of oecumenism, we shall try to introduce a general framework accounting as much as possible for existing Evolutionary Algorithms.

2.1 The generic Evolutionary Algorithm

Let the search space be a metric space E, and let F be a function from E on $\mathbb{R}$ called the *fitness* function. The problem of evolutionary optimization is to find the maximum of F on E (the case of minimization is easily handled by considering $-F$).

A *population* of size $P \in \mathbb{N}$ is a set of P *individuals* (points of E) not necessarily distinct. This population is generally initialized randomly, uniformly on E (see section 6 for a discussion of alternative methods). The fitnesses of all individuals are computed. The population then undergoes a succession of *generations*:

- **Statistics and stopping criterion**: The stopping criterion generally depends on the evolution of the best fitness in the population along generations.
- **Selection**: Choice of P individuals that will generate offspring. Numerous selection processes can be used, either deterministic or stochastic. All are based on the fitness of the individuals. Depending on the selection scheme used, some individuals can be selected more than once. At that point, selected individuals give birth to copies of themselves (clones).

- **Application of evolution operators**: Each one of these copies is applied some evolution operator, giving birth to one or more offspring. The choice among possible operators is stochastic, according to user-supplied probabilities. These operators are always stochastic operators, and one usually distinguish between *crossover* (or *recombination*) and *mutation* operators.
  - Crossover operators are operators from E^k (generally, $k = 2$) into E, i.e. some parents exchange genetic material to build up one offspring[1].
  - mutation operators are operators from E into E.
- **Evaluation**: Computation of the fitnesses of all newborn offspring.
- **Replacement**: Choice of which individuals will be part of next generation. The choice can be made upon offspring only, or among offspring and parents.
- Go to next generation.

Ideally, the evolution operators should be defined on the same space than the fitness function, called *phenotype space*, or behavioral space. Unfortunately, in most cases, one must introduce an intermediate space, called *genotype space*, or representation space. The mappings from the phenotype space in the genotype space is termed *coding*. The inverse mapping from the genotype space in the phenotype space is termed *decoding*. Genotypes undergo evolution operators, and their fitness is evaluated on the corresponding phenotype. The properties of the coding mappings can greatly modify the global behavior of the evolutionary algorithm.

2.2 Standard Evolutionary Algorithms

Most of existing EAs fit in the above framework, and will be presented in that context.

2.2.1 Genetic Algorithms

In the canonical algorithm [78, 66], the genotype space is $[0,1]^n$ [2], the selection scheme is the roulette wheel selection[3], the crossover operators replace bits in the first parent string by the corresponding bits from the second parent, the mutation operator randomly flips the bits of the parent according to a fixed user-supplied probability, and in the replacement phase, all offspring replace all parents.

In more recent works [115, 135], the genotype space can be almost any space, as long as it allows to define crossover and mutation. Moreover, the bias in the roulette wheel selection due to small population sizes was found harmful, and has been replaced by ranking selection[4], or tournament selection[5]. See [67, 44] for a discussion on these selection schemes.

[1] Many authors define crossover operators from $E \times E$ into $E \times E$ (two parents generate two offspring), but no significant difference was ever reported between both variations.

[2] The phenotype space can be any space, as long as it can be coded into bitstring genotypes.

[3] P random choices are made in the whole population, each individual having a probability proportional to its fitness of being selected. The fitness function must take positive values.

[4] The roulette selection is performed on the rank of the individuals (best individual is assigned rank P) rather than on their actual fitness

[5] One selects the best individual among a uniform choice of T individuals, T ranging from 2 to 10.

2.2.2 Evolution Strategies

The original ES algorithm [138, 156] handles a population made of a single individual given as a real valued vector. This individual undergoes a Gaussian mutation: addition of zero-mean gaussian variable of standard deviation σ. The best from the parent and the offspring becomes the parent of next generation. The so-called 1/5 thumb rule[6] was used to adjust parameter σ.

More recent ES algorithms [156, 8] are population-based algorithms, termed $(\mu, \lambda)-$ES or $(\mu + \lambda)-$ES: μ parents generate λ offspring[7], with $\lambda > \mu$. The replacement step is deterministic, i.e. the best μ individuals become the parents of the next generation, chosen among the λ offspring in $(\mu, \lambda)-$ES, or the $\mu + \lambda$ parents plus offspring in $(\mu + \lambda)-$ES. Typical values for (μ, λ) are $(1, 7)$, $(10, 100)$ or $(30, 200)$. When working on real-valued vectors (still their favorite universe) ESs generally use *adaptive mutation*: the standard deviations are part of the individuals, and undergo mutation as well. Last, ESs now frequently use a global recombination operator involving all individuals in the population.

2.2.3 Evolutionary Programming

Originally designed to evolve Finite State Machines [59], Evolutionary Programming emphasizes the phenotype space. As in ESs, there is no initial selection: Every individual in the population generates one offspring. Moreover, the only evolution operator is mutation. Finally, the best P individuals among parents and offspring become the parents of the next generation.

Recent advances [56] handle any space, use adaptive Gaussian deviations for real-valued variables [57], and a stochastic tournament replacement scheme: each individual (among the $2P$ parents plus offspring) encounters T random opponents, increasing its score by one point if it has better fitness. The P individuals having the highest scores get along to the next generation. Note that EP selection-replacement schemes is *elitist*, i.e. the global optimum of the fitness cannot decrease along generations. This is true also for $(\mu + \lambda)-$ES, but not for $(\mu, \lambda)-$ES nor for GAs, where the current optimum can be lost from one generation to another.

2.2.4 Genetic Programming

Genetic Programming as a method for evolving computer programs first appeared as an application of GAs [99]. Original GP evolves tree structures representing LISP-like S-expressions. This allows to define very easily a closed crossover operator[8]. The evolution scheme is called Steady State Genetic Algorithm (SSGA): a parent is selected by tournament (of size 2 to 7), generates an offspring by crossover only (the other parent is selected by a tournament of usually smaller size). The offspring is then put back in the population using a death-tournament: T individuals are uniformly chosen, and the one with the worse fitness gets replaced by the newborn offspring.

[6] When more than 1/5 mutation are successful (respectively unsuccessful), increase (respectively decrease) σ.

[7] There is no selection at that level, i.e. every parent produces λ/μ offspring on average.

[8] By swapping sub-trees between two valid S-expressions, one always gets a valid S-expression.

More recently, mutation operators, e.g. random replacement of a subtree or random change of a node or a leaf, have been used (see also section 3.3.4, or the state-of-the-art book [152]). But, though establishing itself as the fourth wheel of EC, the efficiency of GP is still passionately debated [127].

2.3 Modern Trends: Hybrid Methods

Many researchers modified further evolutionary algorithms by 'adding' some problem specific knowledge to the algorithm. Several papers have discussed initialization techniques, different representations, decoding techniques (mapping from genetic representations to phenotypic representations), and the use of heuristics for genetic operators. Davis [38] wrote (in the context of classical, binary GAs):

> "It has seemed true to me for some time that we cannot handle most real-world problems with binary representations and an operator set consisting only of binary crossover and binary mutation. One reason for this is that nearly every real-world domain has associated domain knowledge that is of use when one is considering a transformation of a solution in the domain [...] I believe that genetic algorithms are the appropriate algorithms to use in a great many real-world applications. I also believe that one should incorporate real-world knowledge in one's algorithm by adding it to one's decoder or by expanding one's operator set."

Such hybrid/nonstandard systems enjoy a significant popularity in evolutionary computation community. Very often these systems, extended by the problem-specific knowledge, outperform other classical evolutionary methods as well as other standard techniques. For example, a system Genetic-2N [113] constructed for the nonlinear transportation problem used a matrix representation for its chromosomes, a problem-specific mutation (main operator, used with probability 0.4) and arithmetical crossover (background operator, used with probability 0.05). It is hard to classify this system: it is not really a genetic algorithm, since it can run with mutation operator only without any significant decrease of quality of results. Moreover, all matrix entries are floating point numbers. It is not an evolution strategy, since it did not use Gaussian mutation, nor did it encode any control parameters in its chromosomal structures. Clearly, it has nothing to do with genetic programming and very little (matrix representation) with evolutionary programming approaches. It is just an evolutionary computation technique aimed at particular problem.

2.4 Comparisons

Many papers have been written on the similarities and differences between these approaches [12, 56, 8]. Different points of view can be adopted.

- The representation issue:
  Original EP, ESs and GAs respectively address only Finite State Machines, real numbers and bitstrings. However, recent tendencies (including most works presented in this dissertation) indicate this is not a major difference.

- Bottom-up versus top-down:
  According to the Schema Theorem [78, 66], GA main strength comes from the

crossover operator: better and better solutions are built by exchanging blocks from partially good solutions previously built, called *building blocks*, in a bottom-up approach. In opposition, EP [56], and to a less extent ESs, claim that such building blocks do not exist, at least for most real world problems. The discussion on crossover has been going on for a long time [54, 58]. And even when crossover was experimentally demonstrated beneficial to evolution, it could be because it acts like a large-step mutation. Recent experiments [88] suggest that the answer is highly problem dependent.

  Another example of the duality between crossover and mutation comes from GP history: the original GP algorithm [99] used only crossover, with no mutation at all, the very large population size being supposed to provide all the necessary building blocks to represent at least one sufficiently good solution. But more recent works on GP (including the ones presented here) accommodate mutation also, on a much smaller population.

- Mutation operators:
  The way mutation operators are applied differ from one algorithm to another. Originally, ES used a heuristic adaptation mechanism (the 1/5 rule [138]), which was later turned into the modern adaptive mutation [156], where any individual carries the standard deviation of the mutation it will be applied, which is therefore evolved and optimized for free.

  The strength of mutation in seminal EP works is basically defined as a function of the relative fitness of the individual at hand.

  GA uses static mutation rate, or user-prescribed evolution scheme to globally adjust either the mutation rate (i.e. the number of individuals that undergo mutation) or the strength of mutation (i.e. the average number of bits that are flipped in an individual undergoing mutation).

  Note that local mutation rates (i.e. dependent on the individual at hand) have a significant impact only when almost all individuals undergo mutation, which is not true for GAs where the mutation rate is generally low. However, the importance of local mutation is confirmed by theoretical results in ESs(see section 2.5). A prerequisite for convergence is the *strong causality principle* emphasized in [138] of ESs: small mutations have small effects on the fitness. It then appears of utter importance to be able to adjust the strength of mutation depending on the individual at hand.

- Selection-replacement mechanisms: From the totally stochastic fitness proportional selection scheme of GAs with generational replacement (all offspring replace all parents), to the deterministic (μ, λ) replacement of ES, through the stochastic, but elitist tournament replacement of EP and the Steady-State scheme (tournament selection and tournament replacement) of the original GP.

Nevertheless, researchers are now more and more mixing up all these features to best fit the application at hand, on a pragmatic basis: some ESs applications deal with discrete or mixed real-integer spaces [10], the "binary is the best" credo of GAs has been successfully attacked [7], and the Schema theorem extended to any representation [135]. Some ESs variation incorporate crossover, mutation has been added to GP, etc. And the different selection operators are more and more being used now by all four groups of researchers.

2.5 Theoretical results

Theoretical studies of Evolutionary Algorithms are of two types: an Evolutionary Algorithm can be viewed as a Markov chain in the space of populations, as population at time $t+1$ only depends on population at time t (at least in the standard algorithms). The full theory of Markov chains can then be applied. On the other hand, the specific nature of Evolution Strategies allowed precise theoretical studies on the rate of convergence of these algorithms using probability calculus (at least for locally convex functions).

Results based on Markov chains analysis are available for the standard GA scheme (proportional selection with fixed mutation rate) [53, 124]. The need for an elitist strategy is emphasized by Rudolph [143]. When the mutation rate is allowed to decrease along generations, techniques borrowed from the field of Simulated Annealing give more precise convergence results in probability [40, 41]. Yet a different approach is used by Cerf [29], which considers the GA as a stochastic perturbation of a dynamical system (a caricature GA). The powerful Friedlin-Wentzell theory can then be applied, resulting in a lower bound on the population size for a convergence in finite time of a modified GA (in which the selection strength and mutation rate are carefully modified along generations). However, even this last result is non-constructive, i.e., of limited use when actually designing an instance of Evolutionary Algorithm for a particular problem.

On the other hand, ESs have considered theoretical studies from the very beginning: studies on the sphere and corridor models gave birth to the 1/5 rule [138], with determination of the optimal update coefficients for the mutation rate [156]. The theory of ESs later developed to consider global convergence results in probability for the elitist models [9], as well as for the non-elitist $(1,\lambda)-$ES [144]. The whole body of work by Beyer [16, 17, 18, 19] concentrate on the optimal progress rate for different variants of Evolution Strategies (and for instance justify some parameter settings for self-adaptive mutation given by Schwefel [156]). The main weakness of these results remains that they were derived on simple models of function; their main results (e.g., optimal parameter settings) are nevertheless applied without further justification to any function — and usually prove to be efficient hints.

However, one should keep in mind that all the above theoretical analyses address some simple models of Evolutionary Algorithms. As stated in section 2.3, the modern trends of EC gave birth to hybrid algorithms, for which generally no theory is applicable.

2.6 Evolution of Evolutionary Computation at CMAP

The history of Evolutionary Computation at CMAP reflects the main trends of the French EC community: It is true that GAs have been the first visible part of the EC iceberg in France, and is still the most widely known EC algorithm, as can be seen in the Proceedings of the first and second French EC Conferences, EA94 in Toulouse in September 1994 [4] and EA95 in Brest in September 1995 [5]. Is it because of their fashionable name and concepts, or due to better marketing policy of the corresponding community, the question is beyond the scope of this dissertation.

2.6.1 The bitstring era

Many researchers, including us, started using GAs based on Goldberg's book [66]. The resulting algorithm was a a pure *à la Goldberg* bitstrings GA, which looks like:

```
randomly initialize the population
do forever
   decode and evaluate each chromosome (application-specific)
   statistics and display
   stopping criterion ?
       if yes then
          return best chromosome(s)
   select the parents
   crossover and mutate
end
```

The specific part is limited to the decoding and evaluation of chromosomes. The `statistics and display` part allows a posteriori plots of the best and mean values of the fitness along generations, as well as close examination of the whole population. The `stopping criterion` is either a maximum total number of generation, or a number of generations without improvement of the best-so-far fitness, whichever comes first. User-supplied parameters other than these `stopping parameters` are the `population size`, the `selective pressure`[9] for the roulette wheel selection and the `crossover and mutation rates`. Their adjustment is a trial-and-error process (see [39] for a survey of some possible alternative techniques).

This Genetic Algorithm was used for some small applications, basically MSc-like theses [10] [52, 51] in pattern recognition and constrained trajectory planing. Nevertheless, these brought to our attention the first clues of the potential power of GAs.

The pattern recognition problem, detailed in section 6.4, demonstrates GAs ability to solve problems intractable by standard methods, be they deterministic or point wise stochastic methods like Simulated Annealing. And it also was our first and successful attempt to use the Behavioral Memory paradigm (see section 6 or [154, 116]), incorporating into EC the continuation scheme, originated from Applied Mathematics.

The constrained trajectory planning called the TGV problem (see section 6.3, and [51]), made clear the need for a varying length representation for the trajectories, and that was the end of the bitstring representation at CMAP: why enforce all problems to fit into a bitstring representation while handling varying length representations is both feasible and more powerful?

Furthermore, the other side of the laboratory background is Numerical Analysis: most Applied Maths problems deal with real numbers, and the use of bitstrings to represent

[9] The expected number of offspring of the best individual in the population.

[10] Some last year projects in the course of Ecole Polytechnique studies, that are roughly the same level.

real numbers seems quite arbitrary. More than that, the binary tyranny was beginning to weaken [7, 86, 115], and the schema theorem, the backbone of GA theory, was shown to extend to more complex representations [135].

2.6.2 The time of multi-representation GAs

It then became obvious that some steps in GAs were related to phenotypes whereas others were related to genotypes. More precisely, selection and replacement only need to know the individuals by their fitnesses (phenotype space) while recombination and mutation only deal with the actual representation of individuals, i.e.e their genotypes. This allows to encapsulate the domain dependent knowledge within the representation of genotypes, with their specific random initialization and operators. The second version of the CMAP GA simulator witnesses that change of perspective: written in C++, it does separate the evolutionary part of GAs from the application-specific design of the individuals: a specific representation is a derived class of the abstract class `individual`, and must provide the basic methods: `Random initialization, cloning, cross over and mutation (plus i/o functions)`. The evolution part (concerned with the population) only knows about the abstract class.

```
randomly initialize the population          <specific>
do forever
   evaluate each individual                 <specific>
   statistics
   display                                  <specific>
   stopping criterion ?
      if yes then end
   select the parents
   apply evolution operators                <specific>
end
```

At that moment, the selection and replacement mechanisms concerned with genotypes were still the standard GAs': Proportional selection, crossover and global mutation (same operator for the whole population), all offspring replacing their parents. Apart from problem-specific parameters, the same evolution parameters than in the bitstring algorithm have to be supplied. The sharing scheme [68] has been added in the `statistics` procedure: the actual fitness of each individual is modulated according to the density of the population around it: the fitness of isolated individuals is increased, while that of individuals in well-represented regions of the search space is decreased.

2.6.3 EC: the adult age

The encounter with ESs at PPSN (1991, 1992) was the first step toward a more general view of EC. The final step came with a representation for which crossover operators fail to make sense: The problem is that of function identification using recurrent Neural Nets, which is detailed in section 3.4.2. Starting from Angeline's work [6], the whole EP side of EC suddenly appeared as another alternative in terms of both operators (mutation only) and evolution scheme (EP scheme [59]). In the mean time, the power of ESs adaptive mutation for real-valued variables became obvious [12].

In terms of software design, the third version of the lab-made EC shell reflects this change of perspective. A population of individuals (still viewed as an abstract class by the population) undergoes evolution using specific operators. Both the selection step and the replacement step are now performed using derived class of an abstract class `selector`. Moreover, two new selection-related features have been added. The choice of a mate now depends on the first selected parent, allowing to use advanced techniques like restricted mating [66] or the SelSed scheme [139]. An intermediate selection among one (or two) parent(s) and their possibly multiple offspring allows hybrid schemes like the simulated annealing-like tournament between parents and offsprings described in [107], and successfully used in CENA on ATC problems (see section 5.3).

```
choose scheme and first template individual
randomly initialize the population                    <problem-specific>
evaluate each individual                              <problem-specific>
do forever
   statistics
   display                                            <problem-specific>
   stopping criterion ?
      if yes then end
   until enough offspring do
      initial selection of the first parent           <scheme-specific>
      choose evolution operators
      if crossover,
         select mate(first parent)                    <scheme-specific>
      apply evolution operator                        <problem-specific>
      evaluate each newborn offspring                 <problem-specific>
      select between parent(s) and offspring(s)       <scheme-specific>
   enddo
   replace the parents                                <scheme-specific>
end
```

Of course, the number of user-supplied parameters has increased. The `initial selection`, `intermediate selection` and `replacement` mechanisms have to be designated. Furthermore, even for the same problem, the dependency of the operators on the individual has to be chosen, e.g. the standard deviation for the Gaussian mutation of real-valued variables can be fixed[11], depending on the relative performance of the individual at hand or coded in the individual itself, i.e. determined in a *GA-like*, *EP-like* or *ES-like* manner

Depending upon the choice of those parameters, the standard evolution schemes of GAs (both generational and steady state), ESs and EP (with or without adaptive mutations) can be reproduced. Note that standard Simulated Annealing takes place in that framework, using the Metropolis tournament on a population of size 1. Moreover, any new combinations can be experimented.

[11] or obeying a fixed evolution scheme, e.g. geometric decrease along generations,

Next sections will present the applications developped at CMAP using Evolutionary Computation. The first half (chronologically) of the results are obtained using the multi-representation GA package presented in section 2.6.2, while all other applications, as well as on-going work, use the above latest-to-date EC software.

3 Function identification

This section presents some work in the field of function identification, restricted to functions from $\mathbb{R}^m$ into $\mathbb{R}^n$ that are solutions of inverse problems (i.e. not relevant to data analysis). After a general overview of function identification (section 3.1), the more specific context of inverse problems is introduced in section 3.2. The different models used on inverse problems are detailed in section 3.3, and specific case studies are addressed using these models in section 3.4: The difficulties encountered while applying evolutionary methods are described, together with an outlook of the results. A discussion of these results in section 3.5, with an emphasis on criteria of choice between the different models, opens several directions for further work.

3.1 Overview

The target of the general identification problem is a function from $\mathbb{R}^m$ into $\mathbb{R}^n$, for some integers m and n. The choice of a *model* for the target function restricts the search pace to a subspace of functions from $\mathbb{R}^m$ into $\mathbb{R}^n$.

Before choosing both a model and a method for function identification, the following questions must be answered:

- What does one know about the target function?

  If some values taken on a set of points of the definition space are available, the identification problem is amenable to *function approximation* or *data analysis*, or *supervised learning*, depending on the field you are working in. In that case, the data (point of the definition domain, corresponding value of the target function) are often termed *samples*.

  In the opposite case, there must exist some quality measure to be optimized, defined on the functional search space. This quality measure can be either analytically defined or only reachable via a computational process.

- Is there a mandatory model for the target function? Such a model can derive from the problem itself, or can be given by experts in the application domain. The choice of a model determines the search space, and very often constraints the choice of the computational method.

- Last, but not least, what are the experts of the application domain expecting from the identification? If the emphasis is put on understandability, only analytical models are of some use. But if the emphasis is on predictive accuracy, any model can be used.

3.2 Inverse problems

If a statistically significant number of values taken by the target function on a set of sample points are given, the identification then reduces to data fitting, and is amenable to numerical regression using statistical methods, or to supervised learning (or scientific discovery) using Machine Learning techniques, depending on the emphasis put on understandability. For instance statistical data analysis provides numerous methods, both parametric (e.g. linear regression) and non-parametric (e.g. nonlinear multi-regression). Or neural models using gradient-like learning methods can be used if the generalization robustness is an issue.

In most situations where a deterministic algorithm shows efficient, EC was observed to be several orders of magnitude slower. Therefore, we shall focus from now on on situations where usual identification methods (statistical methods, supervised learning) do not apply. A typical example of such situation is when the target function is known from its effects, as involved into a physical or chemical process: the identification is an inverse problem.

The resolution of a direct problem consists in building a simulated process that accurately approximate the actual experimental results of a physical phenomenon from the same experimental input data. Such simulation relies on a function (or algorithm) modeling the underlying physical or chemical behavioral law. Figure 1 gives a symbolic representation of a physical process together with its simulated model.

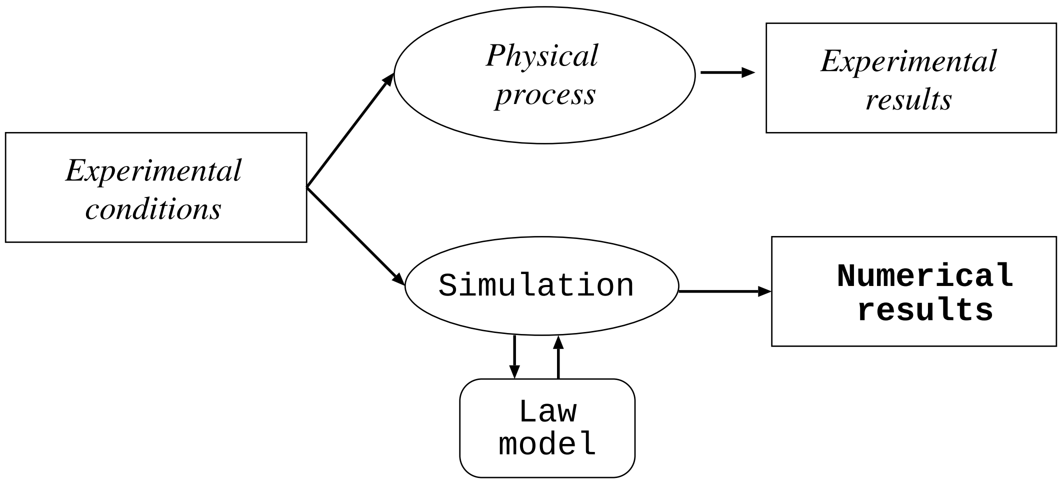

Figure 1: *Direct problem. Physical and simulated process. Simulation is successful (based on an accurate model) iff the error between experimental results and numerical results is small.*

Inverse problems are interested in finding model(s) such that the simulation based on these models successfully approximate experimentation. The data of inverse problems are experimental conditions together with the corresponding actual experimental results.

Whenever a good simulation of a direct problem exists, Evolutionary Computation can be used to address the inverse problem. The fitness of a candidate solution can be computed as shown in Figure 2: the results of the numerical simulation performed using the individual at hand as the function to identify is compared to original experimental results, and the goal is to reach the smallest possible error.

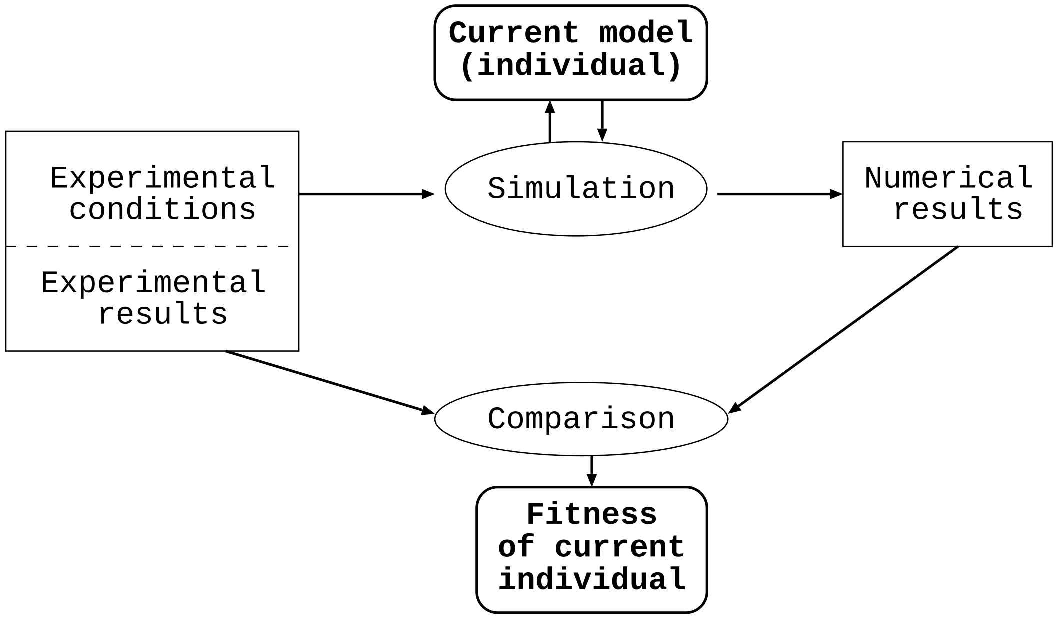

Figure 2: *Evolutionary approach for the inverse problem. The fitness of the individual at hand is achieved by comparing the actual experimental results with the numerical results obtained using the individual in place of the law model.*

A important issue of inverse problem is the *generalization capability* of the solution: How good is the resulting model when used with experimental conditions that are different from those used during the identification process? The usual answer is to use in the identification process many different experimental conditions, also termed *fitness cases*. The fitness is then the average of the error over all fitness cases, and the computational cost is proportional to the number of fitness cases.

Before presenting (section 3.4) some results on three such inverse problems, we first describe the different models used for the function to identify, namely neural networks (both feed-forward and recurrent) and genetic programming.

3.3 Models and associated evolutionary algorithms

The choice of a model for the function to identify fully determines the search space, and is therefore critical for the success of the identification process. A too simple model (e.g. linear models) restricts the search to a small space, and hence might ease the identification task; but it will probably lead to results of poor interest for further use. On the other hand, complex models induce large search spaces, and hence make identification intractable.

The model is sometimes prescribed by the application at hand. Domain expertise can so significantly reduce the search space. But if neither the domain knowledge nor specific technical requirements (e.g. continuity, differentiability) exist, any model can be chosen.

The choice of a fixed model (e.g. polynomials of fixed degree, neural network with fixed topology) turns the identification problem into a parametric optimization problem. Depending on the application domain, it can then be possible to apply standard numerical methods, like Uzawa gradient-based method for the chromatographic problem (section 3.4.2), or backpropagation through time for the neuro-control problem (section 3.4.1).

Nevertheless, even in such situations can evolutionary methods be of some interest, as will be discussed in section 3.5.

The more complex the model, the larger the search space, and the better the solution to the identification problem ..., if found in reasonable time. We shall now introduce the different models that have be used to address inverse problems, namely multi layered perceptrons (fully connected feed-forward neural networks), recurrent neural networks and the tree-structured functions of genetic programming. Section 3.4 will then present feasibility results of evolutionary computation applied in these search spaces.

3.3.1 Artificial neurons

This section briefly recalls the basic concepts of neural networks used throughout our work, both for the sake of completeness and to precisely define the restricted neural networks we have been dealing with. Thorough introduction and latest developments can be found in numerous textbooks, including [145, 74, 75].

An artificial neuron is a small computing unit with local memory (its *weights*). It has real valued inputs, and computes its (real-valued) output by applying a *transfer function* on the weighted sum of its inputs. We only considered in our studies sigmoidal transfer functions (e.g. $f(\lambda) = \frac{1}{1+e^{-\lambda}}$). A neural network is an oriented graph whose nodes are neurons, and whose edges are called the *connections* of the net. Some neurons receive external inputs, and are termed *input neurons*, while in the context of function approximation or identification, some particular neurons are termed the *output neurons*: their output is considered one of the outputs of the network. Neurons that are neither input nor output are termed *hidden neurons*. If the graph of the network includes loops, the network is termed a *recurrent neural network*, otherwise it is a *feed-forward neural network*.

In the frame of neural networks, function identification consists in building a network (topology, i.e. graph of connections, and weights) such that the outputs, considered as functions of the inputs, best fit the available data. A simpler model, often referred to as neural network *learning*, consists in finding the optimal weights for a given topology.

3.3.2 Feedforward neural networks

The model:
The simplest model of neural network is the multilayered perceptron [119]: neurons are organized in *layers*, the first layer receiving as inputs the inputs of the system, and the output of the network being those of the neurons in the last layer. Each neuron of layer n receives as inputs the outputs of every neuron of layer $n-1$. For given inputs, the output of the network can be computed sequentially, by propagating the information from first (input) layer to last (output) layer.

Multi-layer perceptrons are known to be *universal approximator*, in the sense that they are dense in the space of square-integrable functions [79, 171]. This model is thus sufficient, at least theoretically, to tackle any function identification problem. But, except in the one-dimensional case [133], only thumb rules are known regarding the actual complexity of the neural network needed to approximate a given function with a prescribed precision.

Learning methods:
The parameters of the model usually are are the weights of the connections. When multi-layered perceptrons are used for function approximation, gradient-based methods can be used to learn those weights. The most famous is the stochastic gradient method, which is called in that case the *backpropagation* method [37]. This method received much attention from the neural network community, and has been both criticized and improved [77, 32]. Anyway, no gradient-based method does apply directly to the identification problems in the framework of inverse problems, as no samples are here available.

However, the backpropagation method has been modified to handle a particular instance of inverse problems in the domain of control and motion planning, giving birth to the method of *backpropagation through time* [123]. This method will be discussed in section 3.4.1, as an alternative to the evolutionary computation method we propose for that type of problem.

3.3.3 Recurrent neural networks

Recurrent neural networks (RNN) are networks for which the graph of connections contains loops.

Two different situations are distinguished:

- The goal is to simulate sequences of outputs $(\hat{x_1}, \ldots, \hat{x_n}, \ldots)$ (e.g. time series). The network is used to predict $x_{\hat{k+1}}$ from $(\hat{x_1}, \ldots, \hat{x_k})$: All neurons are simultaneously updated from the activations at step k. In that context, the hidden neurons are sometimes called the short term memory of the network.

- The network is used to predict single function values, as in the general problem of function identification. In that case, only the equilibrium points (the attractors) of the dynamical process of synchronous updates, are used[12]. The outputs of the network are defined to be the activations of the neurons labeled as output neurons at such equilibrium point.

When samples of the behavior of the network are available, gradient-based learning methods have been designed to learn the weights for a fixed topology (e.g. recurrent backpropagation, backpropagation through time, see [129] for a survey of the state of the art) for both fixed point learning and temporal sequences prediction.

In our case, no samples are available. Moreover, we shall only consider RNN via their fixed points for function identification. The disadvantages of considering only fixed points are clear. First, the behavior of a recurrent neural net with arbitrary connection matrix is a priori unpredictable: It can have no fixed point, or multiple fixed points. Second, the computational complexity of a fixed point can indeed be very large, and can hardly be estimated. Partial answers we brought to these issues will be discussed in section 3.4.2.

But why on earth use RNN when multi layered perceptrons are simpler, cheaper and theoretically sufficient? An answer to that question is fairly general: Being a superset of layered networks, recurrent networks certainly can exhibit a greater diversity of behaviors. We shall see in section 3.4.1 that some particular situations in the control problems could

[12]If the norm of the connection matrix is small enough, there is always a unique limit value for the activations of the neurons, the fixed point of the synchronous update process.

not be solved (at least not easily) by multilayered perceptrons. Moreover, the chromatographic problem presented in section 3.4.2 also resisted the layered network approach [73], while the use of recurrent network did solve the problem, at least the simple cases tried so far (see section 3.4.2). Here again, on-going work tries to elucidate the reasons for these positive and negative results, and to systematically compare both kinds of neural networks.

3.3.4 Genetic programming: an analytic model

All the work done on Genetic Programming is joint work with Bertrand Lamy.

This section briefly presents the basic concepts of Genetic Programming (GP). Details and numerous applications can be found in [99, 98]. Only significant differences with the original work of J. Koza are reviewed here. GP was born as a subset of Genetic Algorithms. Though it was not fashionable at that time, the general idea was again that the binary representation is not mandatory for evolutionary computation, and that any representation can be used, provided that adapted genetic operators can be used.

GP handles trees representing programs. The original language of GP is a LISP-like language, involving logical operators that are the nodes of the tree, termed *functions* in GP terminology, and boolean variables that are the leaves of the tree, or *terminals*. The space of trees is then in bijection with the space of LISP S-expressions. The crossover operator exchanges subtrees between the parents, thus always giving a semantically correct offspring. Any language involving n-ary operators and constants or variables can be represented in a similar way. For instance, the real polynomials of n variables can be represented using as function set $\mathcal{F} = \{+, *\}$, the standard binary addition and multiplication operators, and as terminal set $\mathcal{T} = \{X_1, \dots, X_n, \mathcal{R}\}$, where $\mathcal{R}$ stands for all real-valued constants.

Two main differences between the original GP and our work will be emphasized:

- the introduction of specific mutation operators to handle floating-point terminals.
- the use of derivatives. The computation of the derivatives of a function given as a GP tree is straightforward, but does not appear in the GP literature as a major issue. Yet, it is the main reason of our choice of GP for the identification of macro-mechanical behavioral laws (see section 3.4.3).

3.4 Case studies

This section presents our most important results in the domain of function identification. Three different inverse problems are considered, in the framework introduced in section 3.2. Each problem is handled via one of the models and methods proposed in section 3.3. All details can be found in published papers, the cornerstone ones being joined to the present dissertation.

3.4.1 Control and motion planning problems

All work reported in this section is a joint work with Edmund Ronald.

Background
We consider a moving vehicle in some subspace of $\mathbb{R}^n$ (in all case studies, n will be equal to 1 or 2). The situation of this vehicle is given by some *state variables*, and the control of its trajectory is performed through some *control variables* (only one control variable will be considered here). The dynamic of the system is known, in the sense that the next state of the vehicle is a known (computable) function of the current state and the control variable.

In all cases, the time variable is discretized. The numerical simulation can be viewed as the closed-loop shown on Figure 3: The command allows to compute the control variable from the state variables at time t. Both the state and the control variables are then used by the simulation of the dynamics of the mobile vehicle to compute the new state variables at time $t + \Delta t$.

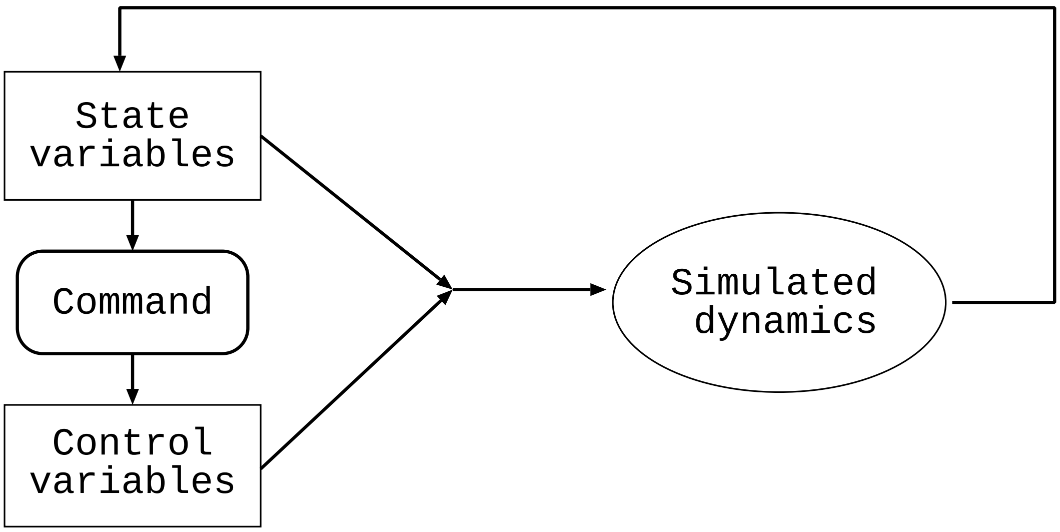

Figure 3: *The numerical simulation of a control problem: closed-loop derivation of the trajectory.*

The following control problems are considered.
CAR PARKING: The position of a car on a two-dimensional parking lot is given by three variables (the coordinates of a point of the car, and the angle between the axis of the car and the x-axis). The control variable is the steering angle of the wheels. The goal is to bring the car to a given position on the parking lot (coordinates $(0, 0)$ and angle 0), from one or more starting points. The number of time steps depends on the starting point, and is *a priori* unknown.

TRUCK BACKER-UPPER:
It is a slightly more complex version of the above car problem, with one additional state variable, namely the angle between the cabin and the truck. This problem was chosen as it is considered a benchmark problem in the neural control literature [123]. Figure 4 illustrates the physical system, while Figure 5 shows what would be the trajectory of the "perfect driver" reaching the target position from one starting point with the shortest trajectory.

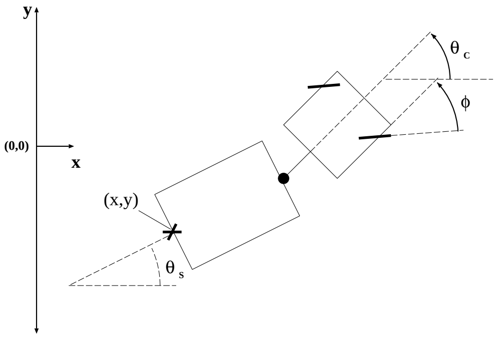

Figure 4: *The truck backer-upper system. The state variables are $(x, y, \theta_C, \theta_S)$. The control variable is ϕ, the steering angle.*

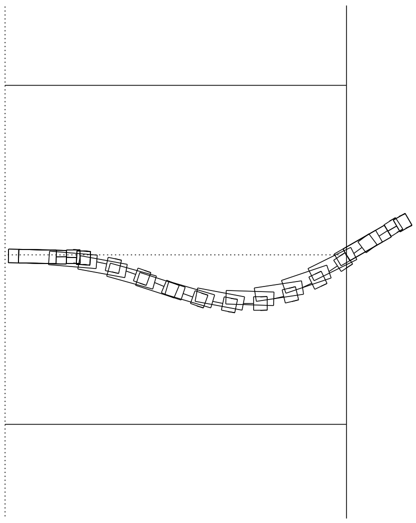

Figure 5: *Successive positions of the truck during a successful parking maneuver.*

LUNAR LANDING:
This famous ancestor of arcade games was chosen because of the difficulty we had to actually interpret the outputs of the controlling neural network of both previous problems. The vehicle is a lunar module trying to land on the moon. The state variables are the altitude and speed of the module, and the control variable is the thrust the pilot can apply. The total amount of fuel is limited, and the goal is to land at the smallest possible speed. Plots of the thrust, the speed and the altitude vs. time allow thorough investigations and understandings that are very difficult to get on any of the two problems above.

State of the art:
Trajectory planning and control problems have received much attention from researchers. Geometric algorithms have been developed, solving almost completely the problem of motion planning [155, 128]. The addition of the kinematics and dynamics of the system has been achieved and implemented on real robots [170]. Moreover, theoretical and practical results exist relating trajectory planning and control: the theory of flat differential systems allow to derive the command of a n-tailer truck to follow exactly a prescribed regular trajectory [141].

We shall only compare here to other neural-based approaches, using these problems as benchmark problems for evolutionary learning of neural networks. In that framework, backpropagation through time is a modification of standard backpropagation method to tackle these problems [123]. The weaknesses of the backpropagation through time are twofold: First, the size of the network makes it difficult for the gradient method to converge from any starting point, and the network must be trained from starting points close to the target point in the first place; moreover, the total number of time steps has to be a priori prescribed, in order to unfold the network appropriately. Second, no constraint can be taken into account, like obstacles in the state space. The use of evolutionary learning of the weights overcomes both difficulties. Furthermore, backpropagation through time does not extend to other identification problems like the chromatographic problem presented in section 3.4.2.

Results using feedforward neural networks
The model used to tackle the above control problems is the multi-layer perceptron model presented in section 3.3.2. The topology of the network is here fixed, and the identification process amounts to the optimization of the parameters of the network: the weights of the connections, but also the slope of the sigmoid functions and some initial renormalization coefficients. The evolutionary algorithm is GAs handling real-valued variables with Gaussian user-supplied mutations.

The basic feasibility of the approach was demonstrated on the car problem, during the "Stage d'option" of S. Damour (this can be approximately compared to a Master thesis) and published in [151]. The generalization properties of feedforward neural networks were studied more thoroughly on the benchmark truck backer-upper problem, and published in [150]. In both cases, however, some limits of the proposed approach were reached.

- The generalization properties of the feedforward neural net seems too weak to overcome the "maximum wheel angle" problem: due to the limit in the steering angle, the best trajectory of the car is discontinuous with respect to the starting position along the half circle delimited by that constraint. And though feedforward networks are universal attractors, we failed in finding a single network that would park the car from both sides of that circle.

- The fine tuning of the last steps of the trajectory was rather difficult to achieve, too: All trajectories from a large range of starting points (far from the "maximum wheel angle" circles) did approach the target parking position, but none did reach it with a good precision.

The first point might possibly be answered by recurrent neural networks: the output of recurrent neural networks are given by the fixed point of the corresponding dynamical system, and it is intuitively possible to design a neural net such that the "maximum wheel angle" circles are the limit of different basins of attractions for the same network.

The problem of fine tuning has been studied on the lunar lander problem. And the solution we propose is the *renormalization of the inputs*: each input to the net is replicated, and a renormalization factor is applied to one of the two identical inputs. Hence, not the same neurons are active (not saturated) for large and small values of the inputs, i.e. far and close from the goal. And experimental results [140] demonstrate that indeed fine tuning is possible in neuro-control.

But the renormalization parameters have to be user-supplied. The ultimate evolutionary step is then to try to evolve these. From the point of view of blind industrial application of a neuro-genetic black box, we try to evolve all critical parameters of our model, i.e. the above renormalization parameters, but also the coefficients of the sigmoid transfer function. The results are satisfactory in the case where the range of the input values is a priori known [149]. In the same context, we also introduce the *expected time to precision* measure of performance: how many independent runs of how many generations are necessary to ensure, with a given confidence, that the best resulting network will land below a given speed ? We feel this to be what engineers would like to know, rather than the average landing speed in terms of generations. Note that this measure of performance could be valuable for most real-world applications.

3.4.2 Analytic chromatography

All work on this problem was achieved by Alessandro Fadda, as part of his on-going PhD.

Background:
Chromatography is a physical process for separating the different components of a chemical mixture using their different absorption rates in some porous medium. Analytic chromatography is widely used in chemical sciences, while preparative chromatography is one of the basic separation processes of petroleum and perfume industries.

The target of the identification problem is the *isotherm* function, giving the absorption rates of the components of the mixture in terms of the concentration of all present components. This function from $\mathbb{R}^n$ into $\mathbb{R}^n$ (n is the number of components) is highly nonlinear. When this isotherm function is known, a mathematical model of the process can be built (namely a nonlinear hyperbolic system of partial differential equations [142]). The "simulation" part in Figure 2 is here a standard numerical scheme obtained by discretizing the hyperbolic system [82]. The complexity of the simulation is in $n \times Z \times T$, if Z and T are respectively the number of space and time steps.

State of the art results in numerical simulation of the direct problem of chromatography involve from one to ten components, and require more than 100 space and time steps to give acceptable accurate results. Note that in preparative chromatography, n ranges from a few thousands to some hundred thousands, and in analytical chromatography, n takes values from ten to one hundred.

State of the art:
A long known model for the isotherm function has been given by Langmuir [100]. It involves rational fractions of the concentrations of the components of degree one. This model has been extended, using statistical mechanics considerations [84], to rational fractions of any degree.

As far as we know, the only results of identification of the isotherm function of a chromatographic process use these models, and a standard gradient based optimization method to identify the parameters of the rational fractions. But this implies writing and solving the adjoint problem: only the very particular formulation of the problem of chromatography made the adjoint problem solvable. More generally, the technique using the adjoint problem requires a specific study for every instance of a problem, and this study can be both difficult and time consuming.

Evolutionary recurrent neural networks:
The first results of evolutionary identification for the problem of chromatography have been published in [55].

These results have been obtained using the recurrent neural net model described in section 3.3.3. The evolutionary method is inspired by Angeline's GNARL algorithm [6], which belongs to the Evolutionary Programming trend:

- Both the topology and the weights of the network are evolved simultaneously.
- No semantically meaningful crossover operator could be designed, and the Evolutionary Programming scheme with only mutations was used: every parent gets one offspring, and the best half of the set parents+offspring becomes the next generation.
- GNARL involves two types of mutation operators. Some address the evolution of the weights in a given topology, and are based on a Gaussian mutation inspired from Evolution Strategies [156]. Anther one addresses the evolution of the topology itself, and proceeds by adding or removing some connections or some neurons.

Our main improvements on the original GNARL approach concern

- the mutation of the coefficient of the transfer function (according to the same Gaussian law), which amounts to a global modification of all input weights of a neuron. Experimental comparisons demonstrate the efficiency of this heuristic.
- a strategy for the applications of the different mutation operators. We make a difference between structural mutations (mutations that modify the topology of the network) and parametric mutations (i.e. the mutations of the weights and the transfer coefficients). After a structural mutation has been applied, a certain number of parametric mutations are performed before another structural mutation can be used again. And parametric mutations are applied as long as they improve the fitness of the network. In some sense, this is analogous to some local search in terms of parameters, for every new structure, and is amenable to the memetic algorithm [136].

The computation of the equilibrium point of each network (section 3.3.3) is achieved through standard iterations from the same initial starting point (all activations set to zero). Evolutionary learning of recurrent neural networks provides an answer to the unpredictability of this method: starting from the same state of all neurons (e.g. all activations being zero), a single equilibrium point can be reached. And if no equilibrium is found after a prescribed computation time, zero fitness is assigned to the network at hand, and it will be discarded by the next selection step.

The results obtained using that algorithm on the one-component problem are excellent, and raise the problem of the choice of test cases (that is, the experimental conditions of the chromatographic process, i.e. initial and injection data). The overall performance of the resulting network is measured as the error on experimental conditions that were not used during the identification process. Both too few and too many test cases in the fitness function lead to networks performing poorly with that respect.

The first results on a two-component problem were also encouraging, though the computational cost becomes an important issue: the networks are on average larger, and the computation of the fixed point for every input becomes hazardous – and it is done a few thousand times per fitness computation.

So actual efforts regard the numerical method used to compute those fixed points. An important remark is that those different fixed points are computed for very close input values (this is inside a loop over the time-space discretization). The idea is to use all activation values of the neurons computed with one set of inputs as a starting point for the computation of the fixed point for another set of input values. This simple remark allowed to greatly reduce the overall computational cost of the standard method – iteration of synchronous updates. Moreover, it allows to use the powerful Newton's method[13]. Newton's method rate of convergence is quadratic and the overall computational cost for large complex networks is significantly reduced compared to that of the usual iteration method.

3.4.3 Constitutive law of materials

Background:
The modeling of the macroscopic behavior of materials is a key issue of mechanical sciences: accurate models can make precise predictions of the behavior of structures, which in turn cause manufacturing cost to decrease and failure prevention to improve. Models for usual materials have been known and adjusted for long. But new materials (polymers, composites) appear now, for which no known model applies accurately. Two general trends exist to derive macroscopic models for such new materials: the study of microscopic properties of materials, and the use of micro-macro operators [146], and the direct derivation of macroscopic *constitutive laws* that are adjusted by fitting the modeled behavior on actual mechanical experiments on real structures [65]. We consider the second approach, which nicely fits in the framework inverse problem section (section 3.2). Loadings involving traction, compression, torsion, ... are applied to a structure made of the unknown material, the displacement and stress of the structure are recorded, and are the tool for the identification process.

State of the art:
As far as we know, there exist no general method to identify constitutive laws other than the refinement of expert guesses. And even such parametric identification is hazardous and highly unstable [97].

Results using genetic programming
Two kinds of mechanical models have been used to this end; all results will be published as chapter 24 of the forthcoming book [152].

One-dimensional rheological models
(joint work with M. Sebag and H. Maitournam, LSM):
Rheological models are general models for elastic, plastic and viscous behaviors of materials [130]. A rheological model involves symbolic components assembled in a planar graph. Strings, dashpots and sliders respectively represent elasticity, plasticity and viscosity. The GP representation for a rheological model uses specific functions and terminals: the basic components (strings, dashpots and sliders) together with their real-valued parameters (stiffness, stress threshold and viscosity) are the terminals. They are linked by two operators, the serial and the parallel operator. The numerical simulation of the macroscopic behavior of a structure involves the interpretation of the GP tree at hand to build the system of PDEs, and the numerical solution of that system. This is where the main technical difficulties arise, but this part of the work was performed at LMS and will not be

[13]Newton's method is highly dependent on its starting point.

emphasized here.

Nevertheless, two general issues have to be highlighted:

- The specific representation of rheological models as planar graphs of symbolic mechanical elements makes it difficult to use any standard optimization method for the identification problem. Moreover, experimental comparison with iterated random hill-climbing is an *a posteriori* argument for GP.

- The GP representation for rheological models involves floating-point values (the coefficients of the symbolic terminal elements), and the design of specific floating-point oriented mutation operators proved useful in that context (see details below).

THREE DIMENSIONAL HYPER-ELASTIC MATERIALS
(joint work with F. Jouve and B. Lamy, CMAP):
The target of the identification is here the *strain energy function* which allows to completely describe the mechanical behavior of *three dimensional isotropic hyper-elastic materials* (e.g. rubber). The simulation tool is here the well-known Finite Element Method (FEM) [173, 33]. The strain energy function is a real-valued function of three variables, as the GP representation is straightforward: the function set $\mathcal{F}$ is $\{+, -, *\}$ and the terminal set $\mathcal{T}$ is $\{X_0, X_1, X_2, \mathcal{R}\}$, where $\mathcal{R}$ represents the floating-point constants.

The following original issues are discussed and experimentally validated in our work.

- Specific mutation operators were designed for floating-point terminals handling. Koza's original work did not make intensive use of floating-point terminals: his applications were concerned with symbolic computation, or used discrete values for the floating-point constants, letting their recombination through the node functions adjust more precisely their values. Having to deal with purely numerical problems, we felt the need for floating-point oriented operators to precisely adjust the terminal values. We so decided to incorporate in GP the ideas that proved successful for the optimization of floating point variables in other optimization problems, namely the Gaussian mutation inspired by Evolution Strategies, described in section 2.2.2. Two new mutation operators were added to the GP algorithm, the standard Gaussian mutation of the floating point constants, and some local optimization of **all** floating point constants based on a numerical local hill-climbing method. These proved useful on both problems of constitutive law identification.

- The numerical solution of the direct problem in the case of hyper-elastic laws requires the computation of second order derivatives of the strain energy function.

  On the one hand, the inaccuracy and instability of (first order) numerical derivation is acknowledged in all Numerical Analysis classes - so its use for second order derivatives was *a priori* rejected.

  On the other hand, the computation of the derivatives of functions modeled by recurrent neural networks is generally impossible: function values are fixed points of a dynamical system, and derivatives can only be reached through some implicit functions theorem. Multi-layer perceptrons could probably have been used: Their ability to approximate derivatives has been demonstrated both theoretically [80] and practically [20, 165]. But limitations of feedforward neural net have already been discussed (section 3.3.3). Moreover, the first experiments using derivatives in GP for standard function approximation (not reported in [152] for limited length reasons) demonstrated a tremendous increase in accuracy - at the expense of computational cost, as derivatives trees are in average three times more complex than their primitive original tree.

- The main difficulty is that not all functions of our search space are viable, i.e. valid strain energy functions. Far from that, only 2% of randomly chosen initial trees give an actual behavior through the simulation of the direct problem, the other 98% leading to "unrecoverable error" from the Finite Element solver. So the search space had to be somehow limited: the problem was rewritten in the light of the theoretical mechanical work of Ogden [126].

Actual results on simple Mooney-Rivlin materials (a simple case of Ogden materials) validate the GP approach. Moreover, the simple iterated hill- climbing method is clearly outperformed. The critical issue is now the computational cost: a Finite Element Analysis using a **compiled** strain energy function takes a few seconds on a HP 755 high range workstation, even on one of the simplest possible three-dimensional mesh (a $2 \times 2 \times 2$ mesh). But when it uses an **interpreted GP tree** as the energy function, this cost is increased by a factor 4 to 20, depending on the complexity of the tree and of its derivatives.

3.5 Function identification problems: Discussion

This section tries to highlight the important issues of identification processes, in the light of our results. Directions for future research naturally follow.

3.5.1 Comparisons of the different models

The choice of a model for the target function is critical. Theoretical results (e.g. the universal approximation property of feedforward sigmoidal neural networks) provide guidelines, but are insufficient: These results are not based on a constructive proof (i.e. they do not tell *how* to build a solution), and the complexity for a given accuracy is unknown[14]. Moreover, such a best approximation model might not be the best solution to the identification problem at hand, as other critical issues are not addressed, namely robustness and understandability.

The arguments for the different models experimented in this work are the following:

- The feedforward neural networks are very simple to implement, and have a low computational cost. Moreover, they are known (experimentally) to have good generalization properties. When the topology is fixed, learning the parameters of a network amounts to optimizing a vector of real parameters. For inverse problems, evolutionary optimization is then a good candidate for the optimization algorithm, as standard gradient-based algorithms do not apply. Backpropagation through time is an exception for control problems which does not generalize [15]. Moreover, evo-

[14]Even if some asymptotic expansion does exist, the multiplicative factors are generally missing.

[15]Though it could be theoretically possible to derive an equivalent method for the chromatographic problem, unrolling the – very particular – space and time Godunov numerical scheme.

lutionary optimization allows to adjust the transfer coefficients and the fine tuning renormalization parameters as well, as demonstrated in [140].

- But in some situations (e.g. in the parking problem when the desired behavior is not a smooth function of the initial conditions), despite their universal approximator properties, feedforward neural networks encounter their practical limits. This is where recurrent neural networks can hopefully take over. Regarding function approximation as fixed points of recurrent neural networks, it should be possible to design networks that exhibit basins of attraction fitting the expected behaviors. Furthermore, chromatographic results witness the feasibility of recurrent neural networks approach for function approximation [55].

  It is true that function approximation based on recurrent neural networks is one order of magnitude more expensive than that based on feedforward neural networks. However, using powerful numerical analysis methods (e.g. Newton method) for the computation of the fixed points of recurrent neural networks can significantly reduce this extra cost. Moreover, it is generally acknowledged that recurrent NN require less complex topologies that feedforward NN for the same task.

- The main advantage of genetic programming for function identification remains its providing of analytic solutions. Whenever understandability is an issue, GP emerges as a first-rate method. The choice of the set of operators (the nodes of GP trees) is still a degree of freedom for the user to somehow guide the search. These operators determine the search space, with the same good old dilemma: too large the search space, and the search algorithm will never find its way out, too restricted and there might not exist any good solution within it.

  A by-product of the analytic form of GP-trees is the straightforward computation of the derivatives of GP solutions. This was a decisive argument for the choice of GP for the identification of strain energy functions [152]. The first results witness that GP at least outperforms a blind iterated stochastic hill climber. Moreover, a specific choice of the node operators allowed us to search spaces for which no other search method exists so far, as for the identification of rheological models [152].

Our results so far establish the basic feasibility of the models of neural networks (both feedforward and recurrent) and of genetic programming for three different types of identification problem. It is now necessary to perform more systematic comparisons of every model on every problem it can be applied to. On going work presently addresses the problem of the truck backer-upper with recurrent neural networks. Our point is to demonstrate that recurrent neural networks can actually solve more difficult problems than multilayered perceptrons, as argued in section 3.3.3, and partially demonstrated on the chromatography problem[16].

But different criterion of choice exist, and the aim of systematic comparison is not to establish an absolute hierarchy in terms of accuracy and/or computational cost. As a matter of fact, first experiments using GP on the car control problem [34] seem to indicate that GP is consistently outperformed by neural networks in terms of computational cost. Nevertheless, understandability can be a strong argument favoring the use of GP (e.g.

[16]Though more thorough experiments are needed on that problem, too.

chemical engineers are very interested to actually see a meaningful isotherm function emerge from the identification process).

3.5.2 The generalization issue

The identification problems studied so far use one or more test experimental conditions as data to the identification algorithm. A measure of the *generalization* ability of result is how well it does perform on experimental conditions that were not seen during the identification process. Indeed, the generalization issue is of utter importance for all engineering identification problems: it is a necessary condition for this solution to be useful.

The generalization capability of the identified function depends on both the model and the algorithm. Some possible directions to handle the generalization have been sketched above, and are presently investigated:

- A gradual increase of the number of fitness cases taken into account in the fitness function is a common way of applying the Behavioral Memory Paradigm *à la Monsieur Jourdain* to address the generalization problem (see section 6). However, no significant improvement has yet been observed in term of generalization abilities. Even more, this method requires a careful ordering of the fitness cases not to degrade the results. Nevertheless, as the computational time is proportional to the number of fitness cases, this method tends to reduce the overall computational cost.

- The coevolution of solutions and test cases is a more promising direction, as quoted in [76] for sorting networks. A parallel evolution of a population of functions and a population of test cases will hopefully select difficult and discriminant test cases which in turn will allow the emergence of robust solutions. The first steps in that directions on the problem of chromatography use at the moment a deterministic evolution criterion for the population of test cases, with promising preliminary results.

- Other heuristics, proposed to tackle a similar difficulty in the framework of classifier systems [64], will be investigated: instead of accounting the different fitness cases solely by the mean square error on all fitness cases, the selection procedure can be modified in such a way that the best individuals on at least one fitness case remain in the population.

- It has been experimentally demonstrated in many situations and theoretically proved in the framework of backpropagation for feedforward neural networks [111, 69] that the addition of noise during the learning phase can be beneficial with respect to the robustness issue. This direction must be investigated in evolutionary inverse problem solving.

3.5.3 Complexity

Computational cost is a general problem of EC, and will be discussed as such in section 8. Only specific solutions for identification problems will be discussed here.

As a matter of fact, the complexity problem is even more crucial in inverse problems, where the computation of a single fitness involves the resolution of as many direct problems as data test cases. There are two more possible sources: First, the numerical solution of the direct problem is often in itself time consuming, even when using a **compiled** function in place of the target of the identification problem. Second, during the numerical simulation of the direct problem occurring within the identification algorithm, the individual at hand can only be **interpreted**, whatever the model. While feedforward neural networks do not generate too much overhead, this is a major issue for both recurrent neural networks and genetic programming.

A possible solution for recurrent neural networks is to restrict to strictly contractive connectivity matrices, ensuring a geometric convergence toward a unique fixed point. However, from an evolutionary point of view, such a restriction a major problem: it turns the basic optimization problem into a constrained optimization problem. As the space of contractive RNNs is not closed with respect to the evolution operators, only a posteriori tests can be performed. Moreover, even contractive matrices can lead to a large number of iterations if the contracting factor approaches 1. We have proposed a solution for recurrent neural networks based on Newton's algorithm, that experimentally outperforms the usual iterative method. The theoretical aspects of this method will be investigated, in the framework of similar work on polynomial equations [164]. The expected issue is an average upper bound for the complexity of the numerical computation of the equilibrium point, in terms of the required precision and the complexity of the network (number of neurons, number of connections), even for networks with non-contractive connectivity matrix.

The problem with genetic programming is slightly different. Many methods have been proposed to minimize the evaluation cost, reviewed in [96]. The Prefix Jump Table structure we have chosen so far is the best we could find according to that survey[17]. But another issue is the following: When a successful solution has been found by GP, two rather different expressions of this solution are exhibited in the literature. The some-pages long expression built by GP together with the one-line long equivalent expression resulting from syntactical simplifications.

As a matter of fact, GP trees are known to increase along generations, more by their size than by their actual complexity: In program trees, expression following an always false expression are never evaluated; In function trees, expressions multiplied by a null term have no impact on the output value of the tree.Such non-coding parts, metaphorically similar to *introns*, seem beneficial to evolution, as they allow the transmission of recessive material, in both fields of GAs [103] and GP [99]. So a systematic syntactic simplification of GP trees may be prejudicial to optimization. However, handling such simplified trees during the fitness computation (and in particular when derivatives trees have to be evaluated as in section 3.4.3) certainly will reduce the overall computational cost of GP-based function identification.

3.5.4 Using actual experimental data

In order to convince engineers that an identification method can actually be of some use to them, this method must address real data. This raises two difficulties:

- Experimental data are generally expensive to get, which makes them rare and sparse. This forbids using coevolution scheme to improve the robustness of the generalization.

[17]A further step in that direction consists in handling trees made of assembly code, as in [125]. But this means the end of the understandability argument.

- Experimental data are noisy. As quoted above, noise can be beneficial to improve the robustness of the identification. But too much noise certainly makes the identification task more difficult.

On-going work on the chromatography problem deals with real data from chemical experiments. The same data have been used to perform the identification of the coefficients of the Moreau-Valentin isotherm using the method proposed in [83]. And this method proved its poor generalization capabilities when the available data involve two different experiments on the same components with different experimental conditions: only one experiment can be fitted by the method, and the other one is then poorly approximated. We have little doubt the evolutionary recurrent neural networks can perform better regarding the generalization issue.

Large amounts of data are readily available in the Laboratoire de Mécanique des Solides, and the GP-based identification methods will be tested using these data.

3.6 Function identification problems: Conclusion

Whatever the model, the optimization method (be it evolutionary or not), the test cases available, we feel that domain expertise is a necessary condition for successful identification. This is true in the first stages of the process, to give a good initial direction (e.g. choice of the best numerical scheme for the chromatographic problem), as well as all along the identification process (e.g. to provide significant test cases), and to understand the results and eventually redefine the search space to correct previous errors (e.g. reformulate the strain energy function after the first disastrous results on strain energy function identification). This can only be achieved by a tight cooperation between computer scientist and open-minded experts of the domain of application.

4 Shape Identification

This section is devoted to the second main field of applications of this work, namely shape optimization and identification. The general abstract problem is presented in section 4.1. Some representations for shapes are introduced in section 4.2, and an overview of the results on two applications in the domain of Structural Mechanics is given in section 4.3. Further research, discussed in section 4.4, is concerned with a systematic comparison of the different possible representations. But the representation and the problem at hand cannot be studied separately. This point of view is discussed in section 4.4: The long-term objective is to build heuristics (or theoretical criterion ?) determining "the" representations adapted to the current problem. Another further issue is to fill the gap between both fields of shape identification and function identification.

4.1 The abstract problem

Identifying a shape amounts to find a partition of a subset of $\mathbb{R}^n$ ($n = 2$ or $n = 3$), called the *design domain*, into two subsets. What each subset actually represents, as well as the optimality criterion on shapes is problem-dependent. In the following, a *shape* will denote such a bi-partition of the design domain.

When the goal is simply to identify a known target shape, the fitness of an individual is given by its distance to the target shape (e.g. Hausdorff distance). But in all of our applications, the shapes are embedded into physical systems and the target shape(s) is (are) defined as the shape with an optimal behavior with respect to this physical system (e.g. according to engineering requirements). Any shape is then evaluated from a two-step process: the physical behavior of the shape is simulated; then this behavior is estimated either with respect to its physical properties, or to engineering requirements.

Two important problems pertaining to that frame in the domain of Structural Mechanics will be introduced in forthcoming section 4.3, namely the Optimum Design problem, and the identification of elastic inclusion. But we shall first focus on the shape representation issue.

4.2 Representations for shapes

4.2.1 Bit arrays

All work on bit-arrays is part of C. Kane's PhD thesis [93].

In all considered applications, determining the behavior of a shape is basically done by means of a Finite Element Method (FEM) [173, 33], i.e. by a numerical simulation of the underlying physical process on a discretization of the domain of the shape. The bit-array representation is directly based on a regular mesh made of rectangle elements. Each element of the mesh is given a boolean value (0 or 1) indicating which subset it belongs to. This representation is the most straightforward from the point of view of both FEM (this is the simplest possible mesh of the domain) and GAs (the representation is a bitstring). All previous stochastic works addressing the topological optimum design problem have adopted that representation, be they based on simulated annealing [63] or GAs [87, 31, 30].

Nevertheless, a bit-array is not a bitstring. Even though standard genetic operators can be applied on the bit-array representation, we demonstrate on a simple case of optimum design (section 4.3.1) that specific two-dimensional crossover operators clearly outperform the standard one- and two-points bitstring crossover operators [95]. This is due to the geometrical bias induced by one-dimensional operators: they can exchange only horizontal parts of the square domain; if the good schemata involve vertical bands of the structure, they tend to be disrupted by any one-dimensional crossover, and the *building block* hypothesis therefore does not apply. The two-dimensional operators are based on exchanging two-dimensional regions of the shapes (e.g. rectangular blocks or regions separated by random lines).

Using specific two-dimensional evolution operators on bitarray representation, GAs have successfully tackled Optimum Design problems that could not be addressed by other techniques [94] (see section 4.3.1). So what are the needs for other representations?

The bit-array representation is likely the simplest way of representing shapes. However, its precision is that of the underlying mesh: the results cited above were obtained on rather coarse meshes (e.g. 10×20) whereas real-world problems and accurate analyses of the mechanical behavior of the shapes would require (e.g. 100×200). Note that the cost of fitness computation through FEM is roughly quadratic in the size of the mesh. But using a finer mesh would increase not only the cost of the fitness computation, whatever the representation, but also the complexity of the representation in the case of the bitarray representation. That increase of the size of the individuals would in turn require to increase the size of the population and the number of generations to reach the same level of convergence, though this increase cannot be quantified precisely[18]

The price for the simplicity is complexity: the bit-array representation poorly scales up, neither when refining the mesh, nor when handling three-dimensional problems. We investigated other representations in order to overcome these limitations, and in particular to dissociate the complexity of the representation and that of the underlying FEM involved in the fitness computation.

4.2.2 Voronoï diagrams

A possible way of representing shapes comes from computational geometry, more precisely from the Voronoï diagram theory. The ideas of Voronoï diagrams are already well-known in the FEM community, as a powerful tool to generate good meshes [62]. However, as far as we know, the representation of shapes by Voronoï diagrams and their evolutionary optimization is original.

Consider a finite number of points $V_0, \ldots, V_N$ (the *Voronoï sites*) of a given subset of $\mathbb{R}^n$ (the design domain). To each site V_i is associated the set of all points of the design domain for which the closest Voronoï site is V_i, termed *Voronoï cell.* The *Voronoï diagram* is the partition of the design domain defined by the Voronoï cells. Each cell is a polyhedral subset of the design domain, and any partition of a domain of $\mathbb{R}^n$ into polyhedral subsets is the Voronoï diagram of at least one set of Voronoï sites (see [131, 21] for a detailed introduction to Voronoï diagrams, and a general presentation of algorithmic geometry).

Consider now a (variable length) list of Voronoï sites, each site being labeled 0 or 1. The corresponding Voronoï diagram represents a shape (a partition of the design domain into two subsets), if each Voronoï cell is labeled as the associated site (we suppose here that the Voronoï diagram is regular, i.e. to each cell corresponds exactly one site). Note that Voronoï representation of shapes does not depend in any way on the mesh that will be used to compute the behavior of the shapes. Furthermore, Voronoï diagrams being defined in any dimension, the extension of this representation to $\mathbb{R}^3$ and $\mathbb{R}^n$ is straightforward.

The evolution operators on the Voronoï representation are inspired from both the two-dimensional crossover operators designed for the bit-array representation and the usual operators for variable length representations:

- The crossover operators exchange cells on the basis of geometrically-based choice (as in [95]); this mechanism easily extends to any dimension [92];
- a first mutation operator performs a Gaussian mutation on the coordinates of the sites, or randomly flips the boolean attribute of some sites;
- a "standard" mutation for variable-length representations adds or deletes some sites from the list.

[18]This assumption is based on experimental considerations. The only theoretical hint can be found in [28], on a modified GA, where the minimal population size required for convergence linearly depends on the size of the bitstring representation.

An important remark is that this representation presents a high degree of *epistasis* (the influence of one site on the physical shape is modulated by all neighbor sites). This will be discussed further (section 4.4) in a more general context.

Practically, the fitness of all shapes is evaluated using the same fixed mesh; this intends to limit the bias due to the unavoidable numerical noise of FEM. A shape described by Voronoï sites is thus mapped on this fixed mesh: the subset a given element belongs to is determined from the label of the Voronoï site nearest to the center of gravity of that element. A systematic algorithm is used to solve this classical nearest neighbors problem, with complexity $m \times n \times l$, where $m \times n$ is the discretization of the mesh, and l the number of Voronoï sites. Since the number of Voronoï sites is generally quite small (some dozens), using powerful computational geometry algorithms to generate the Voronoï diagram (e.g. the powerful randomized algorithm recently developed [21]), plus some *planar point location* algorithm to find out to which cell a point belongs to, would not be beneficial in that context.

Yet significant gains in the fitness calculation could probably be achieved during the mapping step: for instance, only a few Voronoï sites are modified via mutation, and only mesh elements close to these sites need to be updated and recalculated. But as this would require memorizing the local mapping of a shape, the computational savings can only be achieved at the expense of the memory requirement. On the other hand, in both applications presented below, the dominant part in terms of computational complexity is the FEM, which behaves quadratically in terms of the mesh size. So only further experiments will tell whether optimizing the linear mapping step makes any sense.

4.2.3 H-representation

Another representation for shapes was investigated during the "stage d'option" of J. De-Jonghe at Ecole Polytechnique [46]. The idea consists in having a default value for the design domain (e.g. material in the Optimum Design problem), and in coding the other subset as a (variable length) list of elementary shapes (e.g. the "holes", representing void in the Optimum Design problem). This first try did not come out, due to technical difficulties, among which the computational cost that arose from the file-based interface with a general purpose Finite Element program. More over, using this FEM solver implied a re-meshing for every fitness computation, and could result in large numerical noise which could in turn completely hide small fitness differences. Recent advances using that representation result form a close collaboration with F. Jouve (CMAP), whose Finite Element program [90] is now totally interfaced with our Evolutionary Computation shell.

The evolution operators are quite similar to those of the Voronoï representation:

- crossover by geometrical (2D or 3D) exchange of holes;
- mutation by Gaussian modification of the characteristics of some holes;
- mutation by addition or deletion of a hole;

At the moment, the only available results deal with rectangular holes with boundaries parallel to the axes, described by their center and their dimensions. The holes are taken in a domain larger than the design domain, in order not to bias the boundary parts of the design domain. On-going work [163] is concerned with any rectangles, as well as triangular holes.

The H-representation, as the Voronoï representation, is independent from any mesh, and hence its complexity does not depend on any required accuracy for the simulation of the underlying physical phenomenon. Its merits and limitations will be discussed in the light of the experimental results presented in next section.

As for the Voronoï representation, the simulated behavior of the shapes is computed on a given fixed mesh, to limit the numerical noise due to re-meshing. The criterion to decide which subset an element does belong to, is based on whether its center of gravity belongs to a hole or not[19]. The extra computational complexity due to the projection on the fixed mesh is in $m \times n \times l$ for a $m \times n$ mesh and an individual made of l holes.

Here again, some computational savings might be achieved through updating the mapping of the modified regions only. Same remark as for the Voronoï representation applies: for both applications presented below, the major computational cost is that of the FEM, and the price for computational savings is an expense of the memory requirement. Further experimentations are necessary to determine up to which mesh size significant improvements of the overall process can be achieved that way.

4.3 Two applications in Structural Mechanics

4.3.1 Topological Optimum Design

Background

Optimum Design in Structural Mechanics consists in finding the best design for a structure in an initial domain of $\mathbb{R}^2$ or $\mathbb{R}^3$. In the general framework defined earlier (section 4.1), one subset of the bi-partition is made of material, and represents the actual structure, while the other subset is "made of" void. The optimality criterion is governed by the good mechanical properties of the structure, depending on the constitutive law (behavior) of the material (hyper-elasticity in our applications).

In most cases of Optimum Design, the goal is to minimize the weight of the structure (i.e. the amount of material), while meeting some engineering requirements for given loading cases (e.g. forces, pressure, prescribed displacements, ...). This problem is of utter importance for part manufacturers, as a small decrease in the weight of some widely used part can result in large cuts in the cost of manufacturing.

Two contexts are distinguished:

- When the solution is sought as the continuous deformation of a given initial shape, iterated small modifications of the shape are a deterministic way toward the solution. The methods of *domain variation* (or *sensitivity analysis*) [27] are based on gradient-like optimization techniques in that context.

- But continuous deformations do not affect the topology of the structure (e.g. its number of holes). When the topology of the solution is unknown, the problem amounts to *topological optimization*; the only deterministic method in that context, as far as we know, is the *homogenization method* [15, 2]. Homogenization proceeds by first

[19] More generally, any elementary shape can be used, provided there exists a low cost algorithm to tell whether a given point lies inside or outside this shape.

relaxing the problem and considering probabilistic shapes in the design domain: the density of material ranges in $[0, 1]$ instead of being either 0 (for void regions) or 1 (for the structure itself). The optimal solution is known to lie in the superset of probabilistic shapes, and can be approximated by deterministic gradient-based methods. This probabilistic shape must then be mapped into a real shape. Spectacular results in two and three dimensions have been obtained by the homogenization method [1]. The greatest limitation of this approach is that dealing with probabilistic shapes is possible so far in the frame of linear elasticity only. Moreover, even in this frame, homogenization method can handle neither more than one loading case at the time, nor loading applied on the unknown boundary (e.g. a uniform pressure on side of the structure).

Our experiments consider the classical benchmark problem of topological optimization in two dimensions, known as the cantilever plate: a plate is fixed on one of its boundary, and one (or more) punctual forces are applied on the other end of the design domain. The goal is to minimize the weight of the structure while staying below some upper bounds on the maximal displacement of the points where the forces are applied. The mechanical model will be either linear or nonlinear elasticity. Our experimentations demonstrate that all limitations pertaining to the homogenization method can be overcome by EC, at the expense of large computational time.

Optimum Design and bit-array representation
On the benchmark problem of topological optimum design, the bit-array representation allowed significant breakthroughs, and hence demonstrated the potential power of evolutionary computation in that domain. These results are published in [94], and all details will appear in [93]. All these results were obtained using standard GA-scheme on bit-array representation (see section 2.6.2). The most significant results are the following:

- Overall, EC can accommodate **any mechanical model** for which there exists a numerical simulation algorithm. This is confirmed as the results we obtained for the large displacement model appear to be the first results in Optimum Design of structures in nonlinear elasticity.
- The optimization can take into account more than one loading case (experimental conditions), as well as loadings applied on the unknown boundary of the structure.
- In some situations, many optimal, or near-optimal solutions exist. Evolutionary algorithms, using for instance the sharing scheme [68], are able to provide the engineer with a range of such solutions, allowing him to take into account inarticulate criterion.

The limitations of these results come from the coarse meshes that have been used. This drawback of the bitarray approach have been discussed in section 4.2.1.

Comparative results
The same benchmark problem of Optimal Design has been used for the first comparative results for the three representations of shapes presented in section 4.2 [148]. As expected, and even for a rather coarse mesh, the bit-array representation is outperformed by all other

representations in terms of computational cost as well as in terms of quality of the solution. Both the Voronoï representation and the H-representation derive a quasi-optimal solution. But these solutions are more rapidly found as H-based than Voronoï -based shapes. A tentative explanation for this is discussed in section 4.4.

4.3.2 Inclusion identification

Background:
The underlying engineering problem is the identification of defaults in manufactured parts, e.g. small coal scoriae in steel parts, which can greatly decrease the lifetime of the part. More generally, an unknown structure is made of two materials with different known mechanical characteristics. The goal of identification is to find out the repartition of each of the materials using non-destructive mechanical experiments (traction, compression, torsion, ...), via fitting the corresponding displacements observed on the target structure.

The repartition of the two materials in the structure is again described by a bi-partition of this structure. Likewise, the mechanical behavior of the structure can be computed from that partition and the constitutive laws of both materials for any loading case, using FEM. The fitness of an individual is then computed from the difference between this simulated behavior and experimental behavior of the target structure under the same loadings. At the moment, the reference experiments are obtained through simulated experiments for two reasons: First, numerical noise is much easier to control than experimental noise. Second, since the target shape is known, this allows better insights into the actual behavior of evolution.

The general problem of inclusion identification for elastic materials is known to be theoretically ill-posed [25]. However, in some particular situations of parametric models, theoretical results attest the identifiability of some coefficients [26], but these are not constructive results (i.e. they give no practical indications on how to find these coefficients). Some numerical methods have been developed [35], which are highly unstable: after a few iterations during which some sort of convergence toward the solution is experienced, numerical instabilities suddenly appear, and rapidly lead to some numerical overflow.

Note that inclusion identification is an inverse problem, which fits in the general framework for inverse problems introduced in section 3.2: the bi-partition of materials can be expressed as a boolean function defined on the domain of the structure. Unfortunately, the function identification techniques developed in section 3.3 are not adapted to the identification of boolean functions. Section 4.4 will examine how to fill this gap between function identification and shape identification.

Preliminary results

The test cases are those of [35]. The structure is a two dimensional square, and the discretization is fixed to 24×24. The mechanical model is that of linear elasticity. A numerical simulation takes about 0.1 s on a HP755 workstation. The evolution scheme that demonstrated the best results involves a $(50, 200)-$ES evolution scheme (see section 2.2.2), with EP-like local mutation strengths, as described in section 2.4.

The first results obtained using the Voronoï representation, though not yet published, demonstrate outstanding performances when compared to the state-of-the-art results of [35], and have received citations by two leading researchers working on inverse problems in Solid Mechanics [26, 22]. Figure 6 shows some steps of a successful run: the plots are the mappings of the Voronoï diagrams[20] on the fixed 14×24 mesh, and the optimal solution (with respect to that mesh) is found at generation 63.

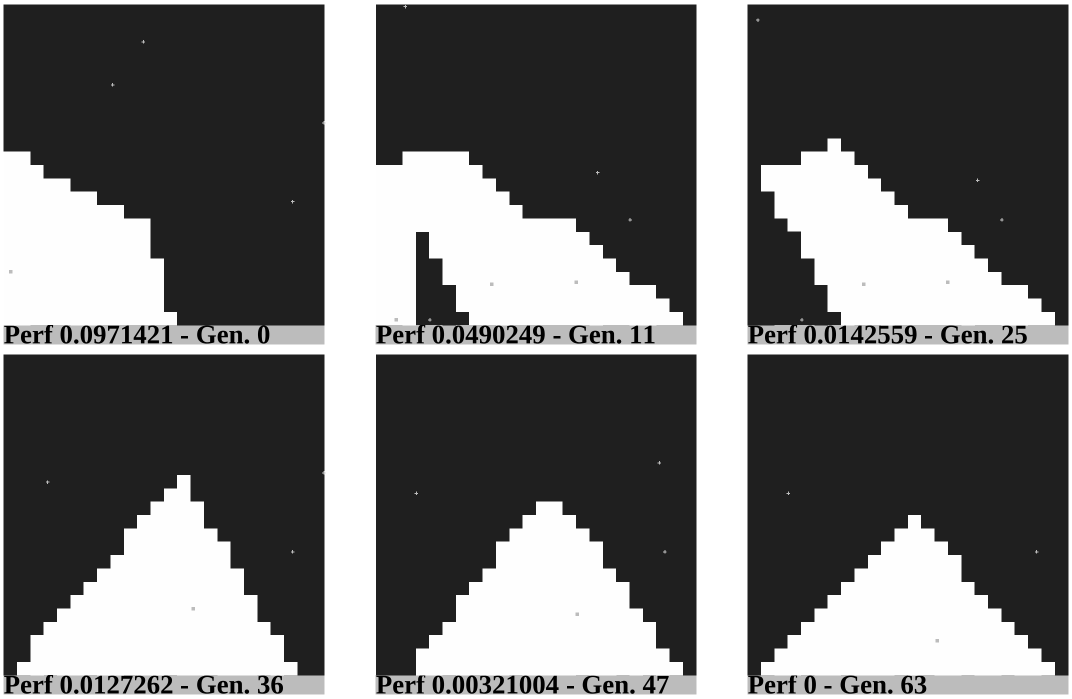

Figure 6: *Snapshots of the best individual in the population at different generations for the inclusion problem. The target shape is found at generation 63 for a population size of 100.*

4.4 Discussion and further directions

4.4.1 Other representations

Many other representations can be used for shapes, and several seem worth investigating.

As mentioned in section 4.1, the CAD community has been studying the problem of shape representations for a long time. Different goals lead to develop different representations. Nevertheless, the widely used B-spline representation deserves more attention from the optimization side. Shapes are represented by their contours (curves in two dimensions, surfaces in three dimensions), and these contours are analytically described by polynomials computed using given *control points*. These control points could in turn be tuned by any Evolutionary Algorithm. We expect no major difficulty in adjusting such control points via EC; the difficulty would rather be to keep track of the connectivity of the different connected components of the boundary.

[20] The Voronoï sites should be visible, white dots on black background and black dots on white background.

A recent attempt [72] to represent shapes using the L-systems paradigm [132] demonstrated ... the need for further study. L-systems are a way to simulate plant growth using grammars (rewriting rules). Spectacular results have been obtained using evolutionary algorithms on similar grammars in the domain of virtual animation [166]. However, the first experiments on the Optimum Design problem of section 4.3.1 only involve grammars working on cells of a fixed discretization of the design domain, thus encountering the same scaling up problem than the bit-array representation. Moreover, no systematic study of the kind of shape that could be best evolved by this method has yet been performed, though the intuition suggest that this representation might be suited for truss structure optimization (e.g. crane arms).

Another possibility to design shapes is to use the fractal IFS (Iterated Function System) theory [14]. An IFS is a system of mappings from the design domain into itself. A dynamical system on that design domain is built by considering iteratively the image of one point by one mapping randomly chosen in the system. This dynamical system has a unique attractor if all functions are strictly contractive. The inverse problem of finding the IFS which has a given area (shape) as its attractor has already been addressed by EC, using parametric representation of affine IFS [61, 105], or Genetic Programming for more general IFS [36]. In both cases, the fitness function was defined as a distance between the evolved IFS attractor and the target attractor. The good experimental results of this approach make it worth a try on shape identification problems.

4.4.2 Comparison of different representations

Experimental comparisons of shape representations encounter the difficulty of designing fair experiments: the mutation operators are not the same for all representations, (e.g. for the fixed-length bit-array representation and for the two other variable-length representations), forbidding any satisfactory way of comparing the mutation strengths. Moreover, experimental comparative results on one problem cannot be generalized to too different problems.

So it is essential to design a methodology, or at least some heuristics, to guide a future practitioner of evolutionary shape identification in his choice: Having so many possible representations for shapes makes it more difficult to choose among them when facing a specific instance of shape identification problem. Some promising directions are given in the literature.

- The fitness variance theory of Radcliffe [137] studies the variance of the fitness as a function of the order of an extension of schemas called *formae* [135], and, simply put, shows that the complexity and difficulties of evolution increases with the average variance of the fitness. But if the formae and their order (or their precision) are well-defined on any binary representation, including the bit-array representation of section 4.2.1, it is not straightforward to extend these definitions to variable length representations discussed above.

  Moreover, Radcliffe's fitness variance does not take into account the possible evolution operators. Further step in that direction would be to study the variance of the change of fitness with respect to a given evolution operator (e.g. the Gaussian mutation of Voronoï sites for different standard deviations).

- The fitness distance correlation of Jones [89] studies the correlation between the distance to the optimal point and the fitness. Simply put again, the idea is that, the stronger this correlation, the narrower the peak the optimum belongs to. Conjectures based on this remark are experimentally confirmed in the GA-frame. Nevertheless, the difficulty in shape representation is to define a distance which is meaningful for both the representation and the problem at hand. The first important issue is whether the considered distance should be purely genotype-based (i.e. defined on the coded individual only) or partially (or totally) phenotype-based (i.e. defined on the same space than the fitness function).

4.4.3 Real-valued shapes

In the problem of inclusion identification presented in section 4.3, the mechanical characteristics of the materials involved are supposed to be known. But in many cases, at least one material has unknown mechanical behavior (e.g. the unwanted scoriae in steel parts can have different origins, and hence obey different laws). A further challenge is to identify both the repartition of the materials and the mechanical parameters of the materials involved.

Another very important (economically speaking) problem in the same vein is petroleum prospection. Before digging a well, acoustic experiments are performed: elastic waves are generated by an explosion at a given point, and *seismographs* (records of waves coming from underneath the ground) are recorded at different points at the surface of the surrounding area (and eventually in some wells nearby). From these seismographs, the engineers must derive a map of the underground material, and estimate from that if digging a well is worth trying. Refining this identification process can save huge amounts of vain efforts.

Up-to-date methods are based on a B-spline representation of the physical characteristics of the materials, and the optimization is performed using some gradient-based techniques. As could be expected, only very simple cases can be successfully solved: Too many local optima prevent any deterministic method from finding the global optimum for complex cases. Moreover, one typical trait of that problem is that the underground is made of different materials near one another: hence, the target function (i.e. the physical characteristic of the materials) is made of areas of constant values separated by discontinuities, which makes it very difficult to approximate by continuous-based models, such as B-spline, polynomials, ...

Both the generalized inclusion problem and the inverse geophysics problem can be addressed through a slight extension of the Voronoï and H-representations:

- The label of both representations can be a floating-point variable instead of a boolean value. All evolution operators apply except the mutation involving the changing of label, which can be turned into the standard Gaussian mutation.
- the computation of the phenotype in the Voronoï representation is straightforward: every Voronoï cell is given the value of the corresponding site.
- Some additional computations are needed for the H-representation, as the holes are

likely to intersect. The value of a point (or element of a mesh) will be set to the average value of all holes containing it.

A common project with IFP (Institut Francais du Petrole) is being launched to study these representations and the efficiency of the corresponding EC algorithms on the inverse geophysics problem.

4.4.4 Symmetry of representations and problems

More original viewpoints seem important regarding the adequation between representation and the problem at hand.

These questions are related to the way genetic material (schema, or formae) is *transmitted* from parents to offspring (regardless of the operators thereafter applied), or, from another point of view, to the degree of *epistasis* of the representation. In biology, the epistasis is a measure of how the expression of one single gene in the phenotype is influenced by other genes in the genotype.

- In the bit-array representation, the contribution of a given bit to the phenotype is clear, and it is not modified by any other bit. The epistasis is here minimal, all genetic material transmitted to the offspring is expressed, i.e. is dominant. We call such a situation a *strong transmission*.

- In the Voronoï representation, the influence of a site on the final phenotype can be greatly modified by the other neighbor sites. If it becomes surrounded by other sites belonging the other subset, it can almost disappear from the discretized shape. Some part of genetic information transmitted to offspring is recessive, and, by contrast to the bit-array situation, we call this case *weak transmission*.

- In the H-representation, the situation is even more complex: Only holes are strongly transmitted. When a hole is transmitted from parent to offspring, the whole area will stay a hole. But no such strong transmission can be achieved for a non-hole. The transmission is *asymmetric*, strong for the hole value, and absolutely weak for the default value (i.e. any hole covering the same area changes its value). Or one can say the holes represent dominant genetic material, while the default value is recessive.

The problems themselves can be similarly studied and classified.

- The Optimum Design is by nature asymmetric: the fitness function aims at reducing the weight, i.e. the amount of material of value 1. On the other hand, meeting the engineering requirements (bounds on the displacement and/or the stress) goes through increasing the weight.

- On the opposite, in the inclusion problem, both values are a priori equivalent with respect to the fitness. But domain knowledge provides some guesses, e.g. there are usually small inclusions of material 1 in a structure almost totally made of material 0.

These considerations might help to understand the reasons why the asymmetric H-representation seems to outperform the symmetric Voronoï representation on the asymmetric Optimum Design problem. Experiments on the inclusion problem are on-going. The long-term goal is to propose criterion in order to quantify the rate of asymmetry of a given problem, and determine accordingly the transmission scheme most promising.

4.4.5 Identification problems: Shapes vs. functions

In the perspective of the generalized inclusion identification or in the inverse geophysics problem, the difference between shape identification and function identification becomes hardly perceptible.

All problems of shape optimization and inclusion identification are relevant to function identification, but the expected outlook of the solution (discontinuous piecewise constant) allows one to guess that the models described in section 3.3 will probably perform poorly on that kind of functions. It is well-known in standard approximation theory than numerical oscillations appear around such discontinuous points.

The only exception worth trying is genetic programming (section 3.3.4): when the function set only contains usual arithmetic operators (e.g. $+, *$) or functions (e.g. $\sin, \log$), the functions represented by GP-trees will encounter the same numerical problems to fit discontinuous functions than classical approximation methods. But specific nodes can be added in GP-function set to overcome this difficulty: ternary nodes of the form `if (Branch1 > 0) then Branch2 else Branch3` might prove successful.

Another bridge between shape identification and function identification can be built the opposite way. As we already mentioned, a floating-point value can be given instead of a label in both the Voronoï and the H-representation. This could be called a zeroth order approximation, giving piecewise constant functions on the design domain. But a first-order approximation can as well be built from the same data, borrowing ideas from the standard interpolation theory of Finite Element, and giving piecewise linear continuous functions on the whole design domain.

The ultimate step in that direction is a complete hybridization of both approaches: instead of a label or a floating-point value, basic components of shape representations (Voronoï sites or holes) can be attached a complete function; Any model can be chosen, which comes with its evolution operators. But due to the expected very high computational cost of such hybrid approach, the issues of shape representations and function identification should be first thoroughly and separately studied before any attempt toward more complex schemes.

4.5 Shape identification: conclusion

We have presented different evolutionary approaches for shape optimization and identification. The emphasis is put on the representation: the simple bit-array representation allowed significant advances in the domain of topological shape optimization (e.g. the first results in nonlinear elasticity). But other representations are needed to tackle the complexity issue. The Voronoï and the H-representations are both independent of the discretization used in the FEM, and demonstrate good preliminary results on both the

Optimum Design problem and the inclusion identification problem. Their main difference appears to be related to epistasis, i.e. the complexity of the decoding mapping between genotypes and phenotypes: the phenotypic traits in the Voronoï representation are controlled by the nearest genes (Voronoï sites) whatever their value, whereas no other gene (hole) can modify the expression of a gene, and any gene can modify the default value in the H-expression. The relationship between this phenomenon and the performance of the different representations will be made specified in future works.

5 When EC joins the tool box of experts

This section describes three fields of application of Evolutionary Computation, namely the optimization of direction-finders, the synthesis of optical coating filters and some problems related to civil Air Traffic Control. In all three applications, actual breakthroughs have been achieved by the Evolutionary approach, which is now operational among standard tools.

Two other application areas, namely constrained trajectory planning and grey scale pattern matching will be presented in the further section 6, illustrating the presentation of the Behavioral Memory paradigm.

5.1 Optical multi-layer systems synthesis

This work was done in collaboration with the Solid Optic Laboratory of Université Paris 6. It is a real-world problem, and the results obtained so far show how EC, by handling mixed discrete and continuous search space, can obtain outstanding results on difficult problems [108, 109, 110]. Experts in the field have been positively enthusiastic. The GA dedicated to this application is now part of the Optic Laboratory standard tool-box among other optimization tools.

An optical multi-layer system are filters made of a series of layers, made of a different materials, where each material has different optical properties. The goal is to design systems with desired global properties, e.g. antireflection coating, beam splitters and rejection filters. The design variables are the number of layers, and, for each layer, its thickness (a real number ranging in a user-supplied interval) and its optical indices, taking discrete values depending on the material.

An individual is a variable-length list of layers, each layer being defined by two parameters, its thickness taking continuous real values, and its optical indices, taking real discrete values. The adaptation of standard GA (see section 2.6.2) to this representation was straightforward:

- geometrical crossing over of blocks of layers, similar to one-point or two points crossovers, but exchanging layers at corresponding positions in the system;
- mutation of individuals by addition or deletion of layers (respecting a maximal overall thickness);
- mutation of thicknesses by addition of gaussian noise, and of optical indices by change of material.

The evolution scheme is standard generational GA, as described in section 2.6.2. An interesting point is that a local optimization tool, part of the standard tool-box of optical engineers, was used to refine the overall best GA solution. A similar work using evolution strategies [10], but without any local deterministic optimization, reported results almost as good. This confirms two tendencies of modern Evolutionary Computation:

- Evolution Strategies seem to outperform GAs to reach high precision results for real-valued variables;
- one should always use a local algorithm to refine the solution given by any evolutionary algorithm.

Further work in the field is concerned with more industrial applications, and a contract is in discussion. A preliminary step lies in interfacing the GA with the existing tools for filter design. In particular, no special understanding of EC is to be required from the user. Another challenge is to extend this approach to deal with the new technology of materials having non-constant optical indices. The complexity of this problem is beyond reach of all available tools for filter manufacturing. But, in the frame of EC, such a problem is amenable to function identification (see section 3).

5.2 Direction finder optimization

The work reported in this section has been done by L.Taieb as part of his PhD thesis at CMAP [168]. This thesis is funded by Thomson RCM company. All details can be found in [169].

Interferometry is a way to determine the direction of some incoming electro-magnetic signal by using the difference of phase observed between this signal received by different antennas.

The problem is to find the optimal placement of the antennas: the first point is to avoid ambiguous localizations for any signal in a given range of frequencies; the second point is to maximize the reliability and robustness of the localization with respect to measurement errors.

There are three reasons to mention this work here:

- The evolutionary part of this application is a strict black box use of the lab GA package for real-valued variables described in section 2.6.2. So it is in complete contradiction with the thesis defended here, that every problem must be carefully designed to best fit into the evolutionary optimization framework – no doubt that any rule of importance must admit exceptions !

  Note however that, whereas the implementation of the evolutionary part of that application is straightforward, the programming of the fitness function requires great numerical skill.

- A tremendous improvement on previous "hand-crafted" results has been achieved using GAs. State of the art results were limited to 3 aligned antennas, the case of 4 antennas being solved by trials and errors. Using GAs, the only limit for aligned antennas is the computational cost (problems with 8 antennas are solved in a few hours of powerful workstation), and the general case of three-dimensional direction finder is part of on-going work. This application thus confirms the wide range of EC potentialities. But, though quite important for the global credibility of EC in the applied Maths community, this still is not the most striking fact about this application.

- This localization problem intrinsically involves some kind of periodicity since it is expressed in terms of difference of phases, number of wavelengths between the antennas, and so forth. However, even the experts of the field had not yet been able to give a formal characterization of the underlying regularities. But, using the sharing scheme, the GA was able to exhibit multiple solutions. And a close look at these numerical results, on the simplest case of 3 aligned antennas, gave L. Taieb the crucial hint that was missing. He derived analytical expressions giving both the number and the description of optima depending on the total length of the direction finder. the proof of this result is given in [169]. These formulae reduce to a few seconds the computational cost of all possible optimum for any total length, which took up to several hours using GAs. It is one of the rare cases where numerical experiments finally lead to theoretical results.

The analytical formula for 3 antennas is now being extended to any number of antennas. And here again the numerical GA method plays an essential role: First, it gives insights into what varies from the 3 antennas case to the general case. Then it is used to validate (and sometimes invalidate) some analytical hypotheses, hopefully leading to some formulas dealing with any number of aligned antennas.

5.3 ATC related applications

Because the CMAP was only partially involved in them, we shall briefly mention four applications of GAs in the domain of Air Traffic Control (ATC), though they are interesting both from the representation point of view adopted in this dissertation, and by illustrating real world applications of GAs. This work is done in collaboration with the *LOG()* team of CENA, headed by J.-M. Alliot, and most of it is part of D. Delahaye's PhD work [47].

The CENA is the organism in charge of studies and research for improving the French ATC systems. When joining two airports, an aircraft must follow prescribed routes and fly over radio-beacons; these beacons are necessary for pilots to know their position during navigation and help controllers to visualize the traffic. Note that the flight altitude is generally prescribed: all ATC problems will here be viewed as two-dimensional. Two aircrafts passing the same beacon can collide if their trajectories become close enough (< 5 miles): this is a case of *conflict*. The major concern of traffic controllers is to successfully solve conflicts, by giving the pilots instructions to modify their respective routes to stay far enough from one another. The first application of GAs to ATC [3] dealt with conflict resolution. Two fairly simple models are used to modify conflicting routes, based on a time discretization. In the first experiments, each plane can be ordered to modify its direction of $\pm$ 30 degrees at the beginning of each time step, and the fitness function amounts to extra flying time of both planes. The second model incorporates discrete speed variations as well, and the fitness function is modified to take into account the fuel expense (which depends on the speed). In both case, bitstring encoding is straightforward, and the simple *à la Goldberg* algorithm is used (see section 2.6.1). Up to 4 planes conflicts can be solved, and results compare nicely to those of some A^* algorithm and of Simulated Annealing.

Anyway, 4 planes conflicts almost never arise in real ATC: As too many simultaneous planes are impossible to control, each controller only controls planes flying through a given *sector*. The second difficult optimization problem in ATC consists in delineating those sectors. This problem is addressed by GAs in [48]. Air space sectoring must respect multiple constraints, such as the convexity of the sectors and load balancing between sectors. Moreover, the beacons cannot be too close to sectors boundaries to avoid conflicts arising between two sectors. The goal of air space sectoring is to minimize the coordination workload (the amount of information the controllers have to exchange when a plane moves from one sector to another), while respecting those constraints. This problem amounts to graph partitioning with topological constraints, which belongs to the class of NP-Hard problems.

The chosen model for the sectoring is that of Voronoï diagrams (as introduced in section 4.2.2). Note that the number of sectors (and hence the number of Voronoï sites) is prescribed by the known loads of the different routes in the total space, and the maximum load one controller can handle. Each Voronoï cell is defined by the two coordinates of the corresponding Voronoï site. A binary representation of real-valued coordinates was first used on that problem, with poor results as the size of the problem (the number of routes and beacons, and hence the number of sectors) increases. On the other hand, floating-point representation found good solutions for large size problems.

A secondary problem of air space sectoring is the grouping of sectors during "after hours": when the traffic gets lower (at night), one controller can take over multiple sectors. The sectors have to be grouped together, and this is another NP-hard problem successfully solved by GAs [50]. The number of groups is known (from the total loads), and a grouping is represented by a list of points, the centers of the groups. Each existing sector is assigned to a group via a classical nearest neighbor approach. One individual is a floating-point vector, and standard floating-point operators are used.

As described above, the sectoring problem much depends on the traffic from beacon to beacon in the total domain. There are many possible routes (lists of beacons) between any two airports. Another optimization problem is to assign each plane a route given its starting point and destination [49]. This is a constrained problem (e.g. all companies must be treated equivalently) in which the link costs are *non separable asymmetric*, which here again makes the problem NP-Hard[21]. The chosen representation is problem specific: each pair origin-destination is assigned a list of beacons describing the prescribed route. Specific initialization procedure has to be designed, in order to generate non disastrous population (such as resulting from blind random initialization) with sufficient diversity. Good results were obtained, both on toy networks (for which the solution is known) and for real large networks (for which no better solution than the GA one could be obtained).

All the work described in that section was achieved using the CENA GA algorithm,

[21] The cost of one link depends of the flows of all links in the same sector. Hence the overall cost heavily depends on the order in which origin-destination pairs are assigned to links.

similar to that of section 2.6.2. Its originality lies in the addition of a simulated-annealing like tournament between parents and offspring after crossover, as described in [107]. This additional scheme is shown to significantly improve the results, and it has now been incorporated in the latest version of our algorithm, described in section 2.6.3.

At the moment, the air sectoring problem and the traffic assignment problem are treated separately. But each problem depends on the other one, and a simultaneous optimization of both is a mandatory further step. Other on-going work at CENA involves real-scale simulations, before any real use can be imagined.

6 Behavioral Memory

6.1 Motivations

The Behavioral Memory paradigm, first introduced by De Garis [42], relies on the following claim: the set of solutions, given by the population evolved in a given landscape, contains more information than just one solution, even optimal. The localization of the last population somehow summarizes the history of the population, how it did *behave* while evolving under the selective pressure due to the fitness function at hand, hence giving indirect insights into the fitness landscape. For instance, the mean distance between individuals in the last population can be an indication of the steepness of the fitness function around the optimal value, and gives hints on the stability of that solution. On the opposite, the existence of different clusters is a sign of the multi-modality of the fitness function.

This suggests to exploit yet another analogy with natural evolution: the environment of natural species was never static, and nowadays species have had to survive different successive environments (e.g. the geological eras). Modifying the fitness landscape along evolution is a way to mimic such a succession of environments.

Another point of view on Behavioral Memory comes from Applied Maths, and is inspired by the homotopic continuation methods. The continuation method to solve a problem $P_{\mathcal{F}}$ involving a function $\mathcal{F}$ (whatever the problem might be) consists in designing a mapping ϕ from $[0,1]$ into the functional space of $\mathcal{F}$, such that $\phi(1) = \mathcal{F}$ and $P_{\phi(0)}$ admits a trivial solution. Depending on continuity properties of mapping ϕ, the solution to $P_{\phi(t+\delta)}$ can be approximated from the solution to $P_{\phi(t)}$, for small enough values of $\delta > 0$. The solution to the original $P_{\mathcal{F}}$ problem can thus hopefully be obtained by following the solution of $P_{\phi(t)}$ when t goes from 0 to 1: a sequence $0 = t_0 < t_1 < \ldots < t_n = 1$ is constructed, and each solution to $P_{\phi(t_i)}$ is built from the solution of $P_{\phi(t_i)}$. $P_{\phi(t_0)}$ is the trivial solution of $P_{\phi(0)}$, and $P_{\phi(t_n)}$ is the target solution of the original problem $P_{\mathcal{F}}$.

This idea has been widely used in Numerical Analysis. An example of application has already been mentioned in section 3.5: we propose to use a similar continuation method to compute fixed points of recurrent neural networks, following [164] (which addresses the problem of polynomial equations). Another example will be given in section 6.3 below, where a continuation method was the proposition of numerical engineers to solve the constrained trajectory planning problem.

6.2 Guidelines

The basic idea of Behavioral Memory-based evolution is to use a population resulting from a first evolution era, driven by a first fitness function, as the starting point for a second evolution era driven by another fitness function. This second evolution era is biased by the information contained in its non-random initial population. Hopefully, a good choice of the first fitness function can help to find a good optimum to the second fitness function. And of course this process can be iterated as many times as necessary; the only risk is that of premature convergence.

More precisely, one key issue for such an iterated scheme is *genetic diversity*: if an intermediate fitness function drives evolution for too long, the population will likely converge with respect to that fitness; the bias induced by using this population as a starting point for the next era is too strong, and this next era reduces to a coupling between a local search (in the small region where the population is located) and hazardous random search: Only a lucky mutation can help escaping the (probably local) minimum. It is therefore of utter importance to preserve the diversity in the population during intermediate evolution eras. Many schemes have been proposed to this end, usually in the context of multi-modal function optimization. A comprehensive survey of these schemes can be found in [106]. The sharing scheme, (see section 2.6.2, or [68]) has been used in all presented work.

In many applications, the BM paradigm is being used implicitly. When the fitness depends on test cases, a common strategy is to gradually incorporate these test cases into the computation of the fitness function. Or when approximate versions of the problem at hand are available, they are sometimes used in the early generations of evolutionary runs. The argument is generally the computational cost: These approximate fitness functions usually are much faster to compute than the target one.

We concentrate on problems where the overall performance of EC strongly depends on the design of intermediate fitness landscapes.

6.3 Constrained trajectory planning

As mentioned in the introduction, this work was our first try of GAs on a problem that resisted fixed length binary encoding. It was posed as a subject for Ecole Polytechnique students in 1991 as a small thesis in their last year Numerical Analysis course [51].

6.3.1 The problem

Given a three-dimensional geographic map and two points A and B on this surface, the goal is to find the shortest path from A to B that constitutes an admissible trajectory for a very high speed train, the French TGV (Train à Grande Vitesse). Due to the high speed of the train, an admissible trajectory must satisfy tight constraints regarding its slope and its curvature. The only available tool in the applied maths tool-box is a simple local greedy algorithm that repeatedly advances one small step, trying to minimize the distance to the goal while respecting all constraints. But this local method fails when some backtrack should be necessary (no step forward is possible without violation of one constraint), which happens with very easy landscapes (e.g. any steep hill straight ahead).

The typical strategy developed for this problem [23] consists in embedding this local method into a homotopic continuation scheme (as introduced in section 6.1). More precisely, a succession of surfaces is considered, smoothly ranging from flat plane to the target surface. The path on the flat landscape is the straight line between A and B, and the path found for a given surface is used to guide the local method on a slightly different surface. However, this strategy gets trapped in a local optima in many real-world contexts, failing to find a path from a valley to another.

6.3.2 An evolutionary approach

The key idea of the evolutionary solution (not involving the BM paradigm yet) is fairly simple: find a sequence of *intermediate points* $A_0 = A, A_1, \ldots, A_n = B$, for some integer n, which are *feasible* in the sense that the dummy local algorithm succeeds in joining any couple $A_{i-1}, A_i, i \in [1, n]$. The goal consists in finding one sequence of feasible intermediate points that minimizes the length of the total trajectory.

The "natural" representation of a trajectory is then a variable length list of intermediate points. The mutation operators on variable length real-valued representations are (today) classical: addition or deletion of an intermediate point, modification of some coordinates by Gaussian mutation. A specific crossover operator driven by geometric considerations is designed: after a crossing point has been randomly selected in the first individual, the crossing point in the second individual is chosen by giving higher probability to points geographically close to the first point.

The first try for the fitness function is defined the following way: The goal is to minimize the total length of the path for feasible trajectories, and, for non-feasible trajectories, to minimize the distance to the goal before the local algorithm stops. A penalization term insures that any feasible trajectory get a higher fitness than all infeasible ones.

This approach immediately outperformed the deterministic continuation method; moreover, it addresses much more difficult landscapes. But an implicit assumption is hidden in this fitness function: that minimizing the distance to the goal would lead to trajectories joining A and B. And this is not true for some atypical landscapes, like the "circus" problem: a circular slope is surrounding a flat valley, point A is inside the circular valley while point B lies outside (the solution is a spiral path allowing to climb the hill without ever encountering too steep slopes).

6.3.3 The Behavioral Memory approach

To overcome this difficulty, the idea is to first evolve a population that would only try to go as far as possible from point A, without any reward for reaching point B, and without any idea of minimizing the length of the trajectory. This evolution era comes to an end when sufficiently many individuals in the population do actually join A and B (whimsical trajectories are many at that time). The next evolution era is driven by the target criterion, i,e, minimizing the total length of an admissible trajectory. During this new evolution era, non feasible trajectories are crudely given zero fitness, and are therefore soon discarded. As quoted above, preserving the diversity of the population in the first era is critical, and is achieved using the sharing scheme. The most spectacular result obtained by this algorithm was to find its way out of the above mentioned "circus".

6.3.4 Related work

This problem does not fit in the classical motion planning scheme 3.4.1, where some points of the domain are forbidden once and for all. In opposition, the TGV constraints are expressed in terms of forbidden directions depending on both its past trajectory (the curvature constraint forbid all but some directions ahead) and the local roughness of the surface (due to the slope constraint). Nevertheless, evolutionary methods have been proposed for motion-planning problems, and could probably be modified to handle the dynamic constraints of the TGV problem (using the BM scheme for difficult surfaces):

- The Ariadne's clew algorithm [112] derives trajectories for mobile robots using the same idea of intermediate points, though only for Manhattan paths (all partial paths are parallel to axes). The goal of the Ariadne's clew algorithm is real time control of robots arms with a large number of degrees of freedom, and moving constraints (another robot moving around). The domain is there the state space of the mobile, and the obstacles do change as time goes on.

- The approach of the Evolutionary Planner of [104] resembles more our algorithm: variable length representation with roughly same operators are used. But the constraints are here fixed obstacles, and promising regions are explored via heuristics more deterministic that the Behavioral Memory based approach.

Though the problems addressed in these works deal with boolean obstacles (fixed in the case of the Evolutionary Planner, moving in the case of Ariadne's clew algorithm), they could probably be modified to handle the dynamic constraints of the TGV problem.

6.4 Grey scale pattern matching

This image recognition problem was very instructive in several respects: In the first place, it was the other early application [52] on which the Behavioral Memory paradigm was tested and validated. Second, it allowed a comprehensive comparison between GAs and standard Simulated Annealing. More details on this problem can be found in [116].

The goal is to find the excerpt of a grey satellite image that best matches a target detail. The image is made of 2000×2000 pixels, each one having an integer grey level in $[0, 255]$. The target detail is a square of 16×16 such pixels. The search space is the set of all 16×16 squares in the original image (represented from their upper left corner).

The evolutionary algorithm used here is an intermediate between the simple GA of section 2.6.1 and the multi-representation GA (section 2.6.2): integer variables (the coordinates of the left corner of the squares, in $[0, 1984]$), with Gaussian mutations and linear recombination, and standard GA selection and generational replacement scheme.

The fitness of an individual is "naturally" given by the difference between the corresponding detail and the target one. The *m.s.e. fitness* is the sum over the 256 pixels of the square of the difference of grey levels. The first results were quite disappointing though, probably due to the highly rough fitness landscape. Better results were obtained when using a local optimization algorithm (a simple hill-climber) after every genetic operator. This amounts to restrict the search space to local optima only, and is sometimes referred to as a *memetic* algorithm [24, 136]. The results of that memetic algorithm were quite satisfying for most target details, and the total computational cost was comparable to that of a basic Simulated Annealing scheme dealing with the same representation.

However, strongly contrasted details were hardly identified, both by GAs and SA. The reason for that is fairly simple. Suppose that the target detail is made of one black pixel centered in a white area, and that the satellite image is totally white except for one black pixel. There exists exactly one optimal individual, mapping the black pixel in the target detail onto the black pixel on the global image. Most individuals are all white, and hence differ from the target detail by one pixel only. But all individuals close enough to the optimal solution contain one black pixel at a different position than the black pixel of the target detail. Hence, they differ from the target detail by *two* pixels, and have a m.s.e. worse than all white individuals. Such fitness resembles the Stromboli volcano in the Mediterranean sea: starting from the sea level, one has to climb the crater (i.e. explore regions with worse-than-average fitness) before being allowed to reach the global optimum. Most optimization algorithms (including SA and classical GA) will fail to locate the global optimum in similar situations (e.g. the white water-wall in the dark sea on the actual satellite image of Dunkerque harbor).

This difficulty is overcome using a two-step BM scheme. The first evolution era is driven by a rough estimate of the m.s.e. fitness. All details are considered through their *histogram* of grey levels, i.e. the number of pixels in the individual carrying a given grey level. The fitness of an individual is computed as the difference between its histogram and that of the target detail. This frame causes the "Stromboli" effect to almost disappear. For instance, in the toy example of the single black pixel on the totally white image above, all individuals having one black pixel will have the same histogram than the target detail, and hence will be considered global optima for the histogram-based fitness function. At the end of the first era, the population will be located in the optimal regions with respect to the histogram-based fitness function. Using the sharing scheme ensures that almost all optimal regions for that fitness function will be sampled in the population. A few generations run on this population using the original m.s.e. fitness are then sufficient to find the global optimum.

Note that we were not able to find such contrasted details using the SA algorithm. But some of the most recent advances in SA (e.g. Adaptive Simulated Annealing [81]) might give analogous results on this problem if used by experienced users.

6.5 Constraint handling

The problem of handling constraints in evolutionary algorithms has received much attention, and no universal method is yet known. An exhaustive survey can be found in [114]. Behavioral Memory offers an alternative approach to constraint evolutionary optimization.

The ideal way of handling constraints is to have the search space limited to the *feasible* space (the set of points satisfying all constraints). In some cases, a suitable change of variables can transform the constraint problem at hand in such a desirable way [117]. But on the other hand, the main difficulty in many (deterministic) methods of constraint problem solving is to find even one feasible starting point.

The basic idea underlying the Behavioral Memory-based Constraint Handling Algorithm (BMCHA) is to evolve the same population using different successive fitnesses to sample the feasible region, i.e. get a population of feasible points, regardless of the objective function at hand. It is then possible to use a standard GA *in the feasible region* to perform the optimization task on the objective function, henceforth throwing away infeasible points by giving them zero fitness.

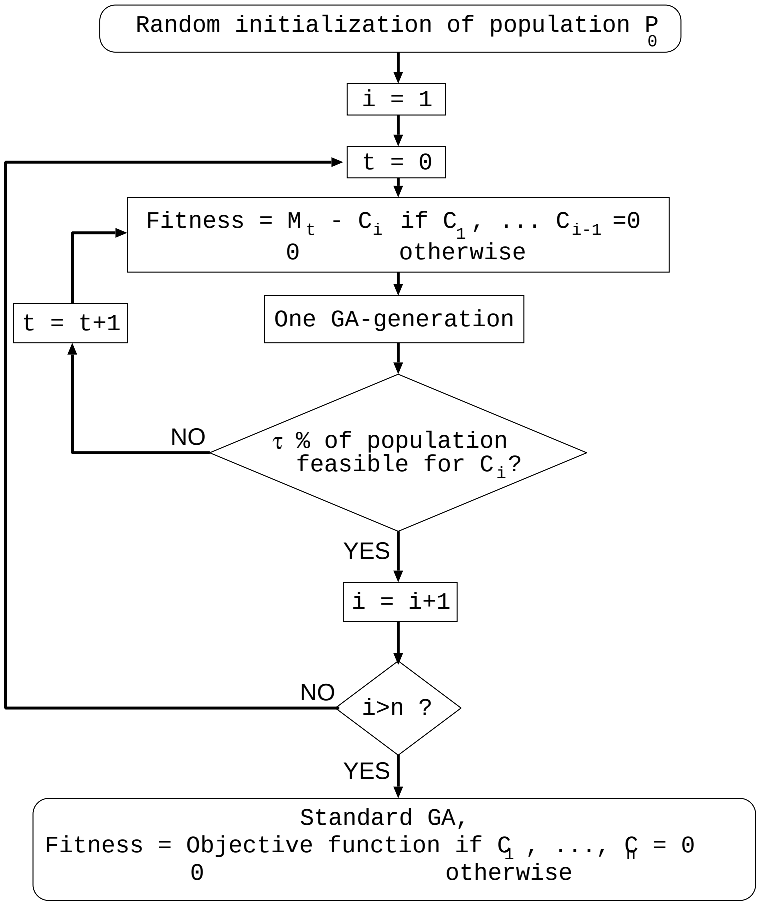

Figure 7: *Flow chart of Behavioral Memory-based Constraint Handling Algorithm*

Suppose the optimization problem is to find the maximum of $\mathcal{F}(x)$ while respecting the constraints $g_i(x) \leq 0$, $i \in [0, n]$.

Define $C_i(x) = Max(g_i(x), 0), x \in E, i \in [0, n]$ and let M_i^t be the maximum value of C_i in the population at generation t of era i. The flow chart of the BMCHA (Figure 7) will now be detailed:

Each era but the final one is devoted to the satisfaction of a single constraint — while ensuring that the population remains feasible for the constraints previously considered. So era i is driven by a decreasing function of the violation C_i of the i^{th} constraint. Moreover, all feasible points must have the same fitness to prevent any convergence toward a particular feasible point. Finally, the maximum of these violations M_i^t over the current population (t is the generation counter) is computed, and the fitness of an individual *which is feasible for the constraints* $0, \ldots, i-1$ is set to $M_i^t - C_i$. Infeasible points for constraints $0, \ldots, i-1$ are assigned null fitness, and will immediately be discarded through selection. Era i ends when a large enough part of the population (τ %, user-supplied parameter) is feasible for constraint i. As stated above, the sharing scheme is used throughout these eras to keep the diversity of the population as large as possible.

Ultimately, this gives a population made of a majority (τ %) of feasible points (i.e. satisfying all constraints). The last era addresses the optimization of the objective function $\mathcal{F}$ in a standard way, except that the infeasible points are discarded by being assigned null fitness. Note the use of the sharing scheme is not necessary in this last era.

More details can be found in [154], as well as a toy example, and results of an application to a truss structure optimization problem (see also [153] for the latter problem). The BMCHA has also been compared to other constraint handling methods in the survey [114]. Though it does not apply to any constrained problem, it demonstrates quite efficient on a bunch of test functions.

6.6 Discussion and further work

The applications of the Behavioral Memory paradigm presented above are arguments for population-based stochastic algorithms against single point algorithms. It was one appealing features of GAs, compared to SA.

Many heuristics described in other sections are more or less implicitly based on the BM approach: gradually refining the time and space discretization in the chromatographic identification problem (section 3.4.2), or in the shape identification problems (section 4.3), or modifying the strengths of the loading cases for the problem of identification of mechanical constitutive law (section 3.4.3)[22].

From a general point of view, the BM paradigm is part of the coevolution scheme. Since its introduction [76], coevolution has been investigated for a great variety of tasks. It concerns all problems where the fitness of an individual depends on the other individuals. In the BM frame, the fitness globally depends on the population: the fitness is refined whenever the current population is "sufficiently good" with respect to the current fitness.

A more flexible approach would be that of Adaptive Behavioral Memory: each individual carries the fitness function it is evaluated with. Segregate populations (as in [102]) made of individuals evaluated from the same fitness function undergo evolution, with a bias favoring fitness functions close to the target fitness function. The coexistence of different species, as being evaluated with different criterion, can prevent the most "advanced" ones to prematurely converge, by bringing them brand new genetic material all along: after all, the Neanderthal and the Homo Sapiens did exist together during quite a long period of time before the former disappeared. Further work in that direction will address the problem of inter-population crossover, and study the bias required to prevent evolution to fall asleep in intermediate fitness landscapes.

7 Control by inductive learning

The work reported in this section has been done in collaboration with Michèle Sebag and lately with Caroline Ravisé (LMS — Ecole Polytechnique and LRI — Orsay). It stems

[22]The choice of the gradually increasing intensities for experimental loadings to be used for the identification is mandatory for two reasons: First, large realistic loadings result in most of random guesses (in the initial population) to be non viable (i.e. the nonlinear iterations of the FEM do not converge, and no fitness value can be given); Second, all energy functions do behave like their linear part around the initial configuration for very small loadings, which makes impossible to distinguish between different laws with same linear part at that point. One solution to that problem is to gradually increase the intensity of the loadings along evolution.

from our previous interests in Machine Learning.

7.1 Motivations

It has been emphasized that the behavior of an EC algorithm widely varies depending upon the bunch of settings devolved to the user (apart from the representation of the problem domain). By decreasing order of generality, the user chooses

- The selection and replacement schemes (size of tournaments, selective pressure,..);
- The size of the population;
- The space of operators and their probabilities.
- The way operators depend on individuals they apply on.

It is long acknowledged that these settings strongly influe on the speed of evolution, which ranges from "slow" to "unbearable".

Many works are concerned with adjusting the global parameters of GAs, such as population size and selective pressure [101], or crossover and mutation probabilities [70, 39, 71, 91]. In ESs, a satisfactory adjustment of these parameters is adaptively guided on the basis of theoretical results [9].

Finer controls are concerned with locally adjusting the effects of evolution operators, i.e. depending on the individual operators apply on. In the GA frame, some authors have presented ways to control respectively type of crossover [167] or the crossover mask [147] most adapted to the individual crossover applies on. In the ES scheme, the adaptive control of mutation is automatic, since the individual carries the strength of mutation to be applied on it [156].

In all of these works, the tuning ultimately relies on statistic estimation (be it off-line [70] or on-line [39]), or on evolution itself [101, 147, 167, 156]. In the last case, the control parameters are encoded in the representation of individuals: evolution can thereby optimize "for free" the control of evolution.

7.2 Guidelines

It was our conviction that at least a part of EC control can be done in a deterministic way. An argument for this is the following: assume that one can define what an error of evolution means; then, an advanced evolution should clearly not repeat its past errors...

Let us make more precise the notion of error of evolution. Give birth to an individual that will not survive the immediate selection step is without doubt an error, in the sense that it brings no usable information and wastes time. So a basic (but not misleading) control would involve storing all past events (births of individuals) that turned out to be errors — in order not to repeat them. In other words, what is needed here is an *explicit memory of evolution*: not only the information hidden in the individuals that have survived evolution, but also that of individuals that have *not* survived.

Before discussing how to extract a tractable explicit memory, let us note the complementarity of this approach with respect to the Behavioral Memory scheme. The BM scheme takes advantage of the implicit memory of evolution encapsulated in the converged population. The analogy for such an implicit memory would be the History of Evolution Successes. In opposition, an explicit memory would rather be the History of Evolution Errors — almost as instructive...

With respect to that distinction, all previous approaches of control are based on the past successes of evolution, inasmuch they store the choices that demonstrated to be good; they do not take explicitly advantage of past errors.

Of course, dealing with the exhaustive list of past errors would be soon intractable, and the explicit memory of evolution must somehow summarize this information. This asks two questions: that of the representation of memory, and that of acquiring such a memory from the story of evolution on a given landscape. In previous works, memory was encoded via the values of control parameters, best suited to the general population [70, 39] or to a given individual [167, 147, 156].

The approach we have chosen [161, 162] is inspired from Machine Learning: the formalism of explicit memory is that of logical rules (in the control parameter space, see below), and the construction of explicit memory is achieved by learning rules from examples. We first describe how to build such an explicit memory and then discuss how to use this memory in order to guide evolution.

7.3 Building an explicit memory

This approach is so far concerned with controlling GAs only, and aims at determining the *bad* operators (crossover and mutation) *not* to apply on individuals.

The control of operators in the literature is done in either very general or very specific terms: e.g. it either governs the number of crossing points [167], or the precise localization of these crossing points [147].

The formalism of rules allows for an intermediate level of control; rule-based memory of evolution allows things such as: *Whenever there was a crossing point between bits 2 and 7, and not between 9 and 10, then it turned out to be an error....* It therefore allows control to locally depend on the topology of the problem domain — in contrast with the uniform control of [39]. But, whereas the local control of [147] can only recommend one point in the operator space for a given individual in the problem space, rule-based control allows to discourage the selection of operators in large *regions* of the operator space for any individual in the population.

By means of deduction, such rule-based memory delivers advises relevant to further evolution steps: *if, in a particular instance of crossover, there exists one crossing point between bits 2 and 7, and none between 9 and 10, then applying this crossover will be an error according to past experiences.*

Note that the hypothesis part of a rule is nothing but a schema in the operator space (boolean space of same dimension than the problem domain, in our restricted frame).

Induction then processes examples, i.e. points in the operator space that are labeled as successes or errors of evolution; from such examples, induction extracts rules, i.e. delineate the regions (schemas) of the operator space that contain examples of success (resp. errors) only. Depending on the regions a further operator belongs to, it will be a priori considered successful or fatal error. Besides these two categories, we introduced the category of *not-events* of evolution [161]. Not-events typically generate offspring that have same fitness as their parents.

For the sake of convenience, we used the machine learning shell developed in the Laboratoire de Mécanique des Solides à l'Ecole Polytechnique [157, 158]. Very likely, any other learning algorithm could have been used, provided that it can deal with noise[23].

7.4 Memory-based strategies of control

The memory of evolution can conveniently be viewed as a set of schemas in the operator space, labeled as *fatal errors, not-events* or *success.* Let $\mathcal{M}_i$ denote the memory of evolution up to generation i. Then $\mathcal{M}_i$ can be used to guide evolution in next generations, through biasing the selection of further operators.

Practically, memory of evolution allows two different strategies of control. The conservative strategy, termed *classical control* only aims at not repeating past errors; to this end, operators conjectured *errors* according to the rules are discarded (not effectuated).

Classical control ensures at least partially the prevention of disruptive operators, i.e. operators that do not transmit the good schemas already emerged in the population [45]. Disruption hinders GA evolution inasmuch the replacement scheme is not conservative: if the best individuals are lost, they must be rediscovered, possibly many times, before they can fully develop; the phenomenon of *fleeting discoveries* is considered one cause for the slowness of GAs [121]. This is far less true in the ESs frame: fleeting discoveries are avoided or limited by elitist selection. However, applying disruptive operators is still, most of the times, a waste of time.
But disruptive operators cannot be systematically discarded: on all non-trivial problems, individuals must sometimes be allowed to quit good regions in order to reach better regions [28]. Practically, disruption is beneficial to evolution in the sense that it limits the loss of diversity of the population [45].

Note that a memory-based classical control does not blindly discard all disruptive operators; rather, it prevents from using again those disruptive operators that have led to negative results in the past.

A second strategy, termed *modern control*, is aimed at preventing the loss of genetic diversity in the population. To this end, modern control rejects operators conjectured *not-events* according to the rules.

7.5 Results

The above schemes of EC control have been validated on two typical toy GA problems, known as the Royal Road [120, 121, 60] and the Ugly (GA-deceptive) problem [172] and on a combinatorial optimization problem, known as the multiple knapsack problem [11]. Preliminary results confirm that significant savings in the number of fitness computations needed to reach the optimal solutions, can be obtained this way [162, 159].

The main limitation of this approach is related to the computational cost of induction. Memory-based control induces a computational overhead: the cost of memory acquisition through induction from examples.

[23] E.g., an operator may be disruptive most of the times for a given population and fitness landscape, and still succeeds sometimes. This means that examples of operators may be inconsistent, i.e. a given operator is associated different labels. Induction must then stand a certain amount of inconsistency, or noise, in the data.

This would be nothing if induction could be done once and for all. But memory must be periodically updated in order to follow evolution. Memory of evolution summarizes the behavior of operators in the explored regions of the fitness landscape; as operators can behave quite differently in new regions, rules guessed from old examples become irrelevant. More than that, they can even be misleading, as they discourage judicious explorations...

To overcome this limitation, every once in a while (every three generations in our experiments), we gather fresh examples of the operator behavior in the regions currently explored by the population; induction is then performed anew from these examples. As the cost of induction is quadratic in the number of considered examples, which is here set to the size of the population, this is not negligible.

In order for our approach of control to make sense, the expected savings in terms of fitness computations must be balanced by the cost of building and maintaining the memory of evolution. Of evidence, control makes sense only when dealing with expensive fitnesses. The operational efficiency of this scheme therefore remains to be confirmed on really expensive problems...

Besides its operational aspects, memory-based control offers valuable tools for investigating the behavior of evolution and the role of operators. This has been demonstrated by comparing what happens when either crossovers only, or mutations only, are controlled (i.e. undergo the rule-based prediction deciding whether they will be effectuated or discarded). Rather unexpectedly, the control of mutations appears to significantly improve the performances (up to a factor 2 in terms of fitness calculations to the optimum), whereas the memory-based control of crossovers does nothing at best, and even happens to slow down the search.

The fact that mutation control is more beneficial to evolution, than crossover control, is counter-intuitive: as many authors [167, 147, 45], we started with the opposite guess [161]. As a matter of fact, the crossover rate is several orders of magnitude higher than the mutation rate in the GA scheme: one then reasonably concludes that the disruptive effects of crossover are much more dangerous than that of mutation.

A posteriori, it becomes evident that disruptive crossovers can be neutralized by the population they apply on: if the population has partially converged on some schemas, these will be transmitted to the next generation through any crossover. In opposition, nothing can ever prevents the disruptive effects of mutations — except control. It has been emphasized that the variance of the fitness with respect to the distance to the optimum is a critical factor of complexity for evolution [89]; this criterion also measures the frequency of disruptive mutations in the regions of optima; controlling disruptive mutations appears a powerful means to speed up the last steps of evolution, which tend to resemble Sysiphus when the global optimum is situated on a narrow peak.

On the other hand, if the disruptiveness of mutation can be limited, there is nothing against a large rate of mutation ... Further experimentations are needed to check to what extent a large-rate controlled mutation can be an alternative to crossover; this would bring some arguments to the never-ending debate about the respective advantages of crossovers and mutations.

7.6 Discussion

The proposed approach offers a new perspective on the control of evolution:

First, it poses the control problem in terms of what should *not* be done, as opposed to what should be done. Proceeding by interdictions is intuitively less misleading than by recommendations, for two reasons: there are far more things to forbid than to recommend, especially after the early generations of evolution. Moreover, we do not know nothing about what should a priori be recommended (for non trivial problems) whereas we know something about what should be forbidden, namely what has previously led to negative results.

Second, it proposes a new formalism for control, that of logical rules. This frame allows a modular control, that depends on the regions of the fitness landscape, (on the topology of the problem); in opposition, the control based on the global operator rates does not allow any modularity.
In the meanwhile, this frame allows a flexible control compared to dictating “the” right operator to apply on a given individual.

Last, it proposes induction as a method to acquire the means for evolution control. Coupling evolution with induction can be viewed as a new paradigm for evolution, namely “civilized evolution” as opposed to “Darwinian evolution”. The analogy for civilization would involve both the "culture", i.e. the memory used for guide the evolution steps, and the generations obeying this culture. By pursuing the metaphor, culture may become inaccurate if it is originated from too old experiments, and this results in misleading evolution (it is the “decadence” effect). The transition between two civilizations is done as follows: the new civilization starts with the best solutions elaborated in the old one, together with a refreshed memory (new culture).

7.7 Further work

This opens to several avenues of research. First, memory-based control has hitherto been employed in the GA-frame only. Its extension to fixed-length representations makes no difficulty since many induction algorithms, among which ours, can deal with real-valued vectors. Its extension to variable-length representations, and tree-structures as encountered in GP, requires much stronger induction algorithms, i.e. able to deal with relational rather than attribute-values representations [134, 122, 160].

Second, a more tight coupling of induction and evolution remains to be imagined, in the line of [13]. Baluja et al. demonstrate for simple problems, that it is possible to extract some explicit “knowledge” of the current population, (here the center of gravity of this population (!)). This explicit knowledge allows to directly build next population (via sampling the neighborhood of this center of gravity); this scheme thereby completely sidesteps the usual evolution operators. Of course, the center of gravity of the population provides very rough information (likely irrelevant for multi-modal fitness functions). We shall investigate to what extent induction from the current population can directly characterize the regions worth a thorough exploitation, as well as the regions not yet explored — which must in any case be explored. From such explicit knowledge, one can explicitly derive a population composed of the desired amount of explorers and followers (individuals respectively belonging to regions to be explored, or to be exploited).

Third, induction is a deterministic way to build an accurate memory. But evolution could as well be used to optimize the content of its memory ... We shall then investigate a co-evolution frame, concerned with optimizing both individuals and operators to be *not* applied on them. Experimentations will tell whether evolution is the best way to take care of evolution — as claimed by fans of *Nature Only* — and whether the specific learning task is better or faster achieved by a specific deterministic algorithm.

8 Conclusion

We have presented in this dissertation some applications of Evolutionary Computation to difficult optimization problems of Numerical Analysis and Structural Mechanics. Those problems could hardly be solved by standard deterministic numerical methods, and often resisted other stochastic or heuristic-based optimization methods. We feel that one key of our successes using EC lies in the careful adaptation of the algorithm to the problem at hand. The design of the search space (the genotype space, defined by the representation) is the first crucial step, as demonstrated on different applications: the choice of a model for function identification problems (section 3), the choice of the inputs for a prescribed NN model (section 3.4.1, or the variety of possible shape representations and the respective performances of the corresponding evolutionary algorithms (section 4.3).

But on the same search space, the choice of evolution operators can also have a critical impact on both the quality of the solution and the computing time to find it (sections 4.2.1 and 3.4.2).

Finally, the definition of the fitness function (or of the successive fitness functions in the frame of the Behavioral Memory paradigm) can also turn successful evolutionary optimization into disasters: see the problem of identification of constitutive laws of hyper-elastic materials (section 3.4.3), or all applications where standard GA failed until a Behavioral Memory-based process was used (section 6).

Following [85], we would compare the present situation of EC to that of AI regarding problem solvers some years ago: during a first enthusiastic phase, people had been looking for *the general problem solver* that would address all possible problems; it progressively occurred that this was merely a mirage, and that taking into account the specificities of the problem at hand could indeed be beneficial.

Recent works in Evolutionary Computation, including the ones presented in this dissertation, witness the same phenomenon: the idea of a general evolutionary optimizer, mainly present in canonical GAs, has to be abandoned. When facing an optimization problem resisting classical deterministic methods, whatever domain knowledge is available should be sought for and used to design and improve an evolutionary algorithm.

Being in daily contact with actual practitioners (from both CMAP, the Applied Maths Center, and LMS, the Solid Mechanics Laboratory, at Ecole Polytechnique) is an asset not only to discover new application fields, but also to successfully tackle difficult optimization problems by a tight collaboration with the experts in the field.

References

[1] G. Allaire, E. Bonnetier, G. Francfort, and F. Jouve. Shape optimization by the homogenization method. *To appear in Nümerische Mathematik*, 1996.

[2] G. Allaire and R. V. Kohn. Optimal design for minimum weight and compliance in plane stress using extremal microstructures. *European Journal of Mechanics, A/Solids*, 12(6):839–878, 1993.

[3] J.-M. Alliot, H. Gruber, M. Schoenauer, and G. Joly. Genetic algorithms for solving air traffic control conflicts. In *Proceedings of the 9^{th} IEEE Conference on Artificial Intelligence Applications*. IEEE, IEEE Press, May 1993.

[4] J.-M. Alliot, E. Lutton, E. Ronald, and M. Schoenauer, editors. *Actes de la Conférence Evolution Artificielle*. Toulouse, Cepadues, September 1994.

[5] J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors. *Artificial Evolution*. Brest, Springer Verlag, 1995.

[6] P. J. Angeline, G. M. Saunders, and J. B. Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5(2):86–91, 1993.

[7] J. Antonisse. A new interpretation of schema notation that overturns the binary encoding constraint. In J. D. Schaffer, editor, *Proceedings of the 3^{rd} International Conference on Genetic Algorithms*, pages 86–91. Morgan Kaufmann, June 1989.

[8] T. Bäck. *Evolutionary Algorithms in theory and practice*. New-York:Oxford University Press, 1995.

[9] T. Bäck, G. Rudolph, and H.-P. Schwefel. Evolutionary programming and evolution strategies: Similarities and differences. In D. B. Fogel and W. Atmar, editors, *Proceedings of the 2^{nd} Annual Conference on Evolutionary Programming*, pages 11–22. Evolutionary Programming Society, 1993.

[10] T. Bäck and M. Schütz. Evolution strategies for mixed-integer optimization of optical multilayer systems. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Proceedings of the 4^{th} Annual Conference on Evolutionary Programming*. MIT Press, March 1995.

[11] T. Bäck, M. Schütz, and S. Khuri. A comparative study of a penalty function, a repair heuristic and stochastic operators with set covering problem. In J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Artificial Evolution*, number 1063 in LNCS. Springer Verlag, 1995.

[12] T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

[13] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithms. In A. Prieditis and S. Russel, editors, *Proceedings of ICML95*, pages 38–46. Morgan Kaufmann, 1995.

[14] M. F. Barnsley. *Fractals everywhere*. Academic Press, 1988.

[15] M. Bendsoe and N. Kikushi. Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and Engineering*, 71:197–224, 1988.

[16] H.-G. Beyer. Toward a theory of evolution strategies: Some asymptotical results for the $(1,+\lambda)$-theory. *Evolutionary Computation*, 1(2):165–188, 1993.

[17] H.-G. Beyer. Toward a theory of evolution strategies: The (μ,λ)-theory. *Evolutionary Computation*, 2(4):381–407, 1994.

[18] H.-G. Beyer. Toward a theory of evolution strategies: On the benefit of sex - the $(\mu/\mu, \lambda)$-theory. *Evolutionary Computation*, 3(1):81–111, 1995.

[19] H.-G. Beyer. Toward a theory of evolution strategies: Self-adaptation. *Evolutionary Computation*, 3(3):311–347, 1995.

[20] C. Bishop. Curvature-driven smoothing: a learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, 4(5):882–884, 1993.

[21] J.-D. Boissonnat and M. Yvinec. *Géométrie algorithmique*. Ediscience International, 1995.

[22] M. Bonnet. Identifications élastiques. Ecole d'automne – Problèmes inverses en mécanique. LMS, Ecole Polytechnique, Palaiseau, 1995.

[23] Y. Brenier. Personal communication, 1992.

[24] Huntley Browne and Spillane. A parallel genetic heuristic for the quadratic assignement problem. In J. D. Schaffer, editor, *Proceedings of the* 3^{rd} *International Conference on Genetic Algorithms*. Morgan Kaufmann, 1989.

[25] H. D. Bui. *An introduction to inverse problems in Structural Mechanics*. CRC Press, Boca Raton, 1994.

[26] H. D. Bui. Sur quelques problèmes inverses élastiques en mécanique de l'endommagement. In *Deuxième Colloque National en Calcul des Structures*, 1995.

[27] J. Cea. Problems of shape optimum design. In E. J. Haug and J. Cea, editors, *Optimization of distributed parameter structures - Vol. II*, volume 50, pages 1005–1088. NATO Series, Series E, 1981.

[28] R. Cerf. *Une théorie assymptotique des algorithmes génétiques*. PhD thesis, Université de Montpellier II, March 1994.

[29] R. Cerf. An asymptotic theory of genetic algorithms. In J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Artificial Evolution*, volume 1063 of *LNCS*. Springer Verlag, 1996.

[30] C. D. Chapman and M. J. Jakiela. Genetic algorithm-based structural topology design with compliance and topology simplification considerations. *Journal of Mechanical Design*, 1995. Submitted.

[31] C. D. Chapman, K. Saitou, and M. J. Jakiela. Genetic algorithms as an approach to configuration and topology design. *Journal of Mechanical Design*, 116:1005–1012, 1994.

[32] C. Charalambous. Conjugate gradient algorithm for efficient training of artificial neural networks. *IEE Proc.*, 139(3):301–310, 1992.

[33] P. G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland, Amsterdam, 1988.

[34] F. Coffin. Programmation génétique pour un problème de contrôle. Rapport de stage d'option B4 de l'Ecole Polytechnique. Palaiseau, Juin 1995.

[35] A. Constantinescu. *Sur l'identification des modules élastiques*. PhD thesis, Ecole Polytechnique, June 1994.

[36] G. Cretin, E. Lutton, J. Levy-Vehel, P. Glevarec, and C. Roll. Mixed ifs: Resolution of the inverse problem using genetic programming. In J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Artificial Evolution*. Springer Verlag, 1996.

[37] Y. Le Cun. *Modèles connexionnistes de l'apprentissage*. PhD thesis, Université Pierre et Marie Curie, Paris VI, 1987.

[38] L. Davis. Adapting operator probabilities in genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the* 3^{rd} *International Conference on Genetic Algorithms*, pages 61–69. Morgan Kaufmann, 1989.

[39] L. Davis. *Handbook of Genetic Algorithms.* Van Nostram Reinhold, New York, 1991.

[40] T. E. Davis and J. C. Principe. A simulated annealing like convergence theory for simple genetic algorithm. In R. K. Belew and L. B. Booker, editors, *Proceedings of the* 4^{th} *International Conference on Genetic Algorithms*, pages 174–181. Morgan Kaufmann, 1991.

[41] T. E. Davis and J. C. Principe. A markov chain framework for the simple genetic algorithm. *Evolutionary Computation*, 1(3):269–292, 1993.

[42] H. de Garis. Genetic programming : building artificial nervous systems using genetically programmed neural networks modules. In R. Porter and B. Mooney, editors, *Proceedings of the* 7^{th} *International Conference on Machine Learning*, pages 132–139. Morgan Kaufmann, 1990.

[43] K. A. DeJong. Are genetic algorithms function optimizers ? In R. Manner and B. Manderick, editors, *Proceedings of the* 2^{nd} *Conference on Parallel Problems Solving from Nature*, pages 3–13. North Holland, 1992.

[44] K. A. DeJong and J. Sarma. On decentralizing selection algorithms. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 17–23. Morgan Kaufmann, 1995.

[45] K.A. DeJong and W.M. Spears. A formal analysis of the role of multi-point crossover in genetic algorithms. *Artificial Intelligence*, 5:1–26, 1992.

[46] J. Dejonghe. Allègement de platines métalliques par algorithmes génétiques. Rapport de stage d'option B2 de l'Ecole Polytechnique. Palaiseau, Juin 1993.

[47] D. Delahaye. *Optimisation de l'espace aérien par algorithmes génétiques.* PhD thesis, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, 1995.

[48] D. Delahaye, J.-M. Alliot, M. Schoenauer, and J.-J. Farges. Using genetic algorithms for sectoring airspace. In *Proceedings of the* 10^{th} *IEEE Conference on Artificial Intelligence Applications.* IEEE, IEEE Press, May 1994.

[49] D. Delahaye, J.-M. Alliot, M. Schoenauer, and J.-L. Farges. Genetic algorithms for air traffic assignment. In *Proceedings of European Conference on Artificial Intelligence.* Amsterdam, August 1994.

[50] D. Delahaye, J.-M. Alliot, M. Schoenauer, and J.-L. Farges. Genetic algorithms for automatic regrouping of air traffic control sectors. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Proceedings of the* 4^{th} *Annual Conference on Evolutionary Programming*, pages 657–672. MIT Press, March 1995.

[51] C. Desquilbet. Détermination de trajets optimaux par algorithmes génétiques. Rapport de stage d'option B2 de l'Ecole Polytechnique. Palaiseau, Juin 1992.

[52] C. Desquilbet and P. Sassus. Reconnaissance de détails en niveaux de gris par algorithmes génétiques. Rapport de travail personnel. Majeure SICS de l'Ecole Polytechnique. Palaiseau, March 1992.

[53] A.E. Eiben, E.H.L. Aarts, and K.M. Van Hee. Global convergence of genetic algorithms: a markov chain analysis. In Hans-Paul Schwefel and Reinhard Männer, editors, *Proceedings of the* 1^{st} *Parallel Problem Solving from Nature*, pages 4–12. Springer Verlag, 1991.

[54] L. Eshelman, R. A. Caruana, and J. D. Schaffer. Biases in the crossover landscape. In J. D. Schaffer, editor, *Proceedings of the 3^{rd} International Conference on Genetic Algorithms*, pages 10–19. Morgan Kaufmann, 1989.

[55] A. Fadda and M. Schoenauer. Evolutionary chromatographic law identification by recurrent neural nets. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Proceedings of the 4^{th} Annual Conference on Evolutionary Programming*, pages 219–235. MIT Press, March 1995.

[56] D. B. Fogel. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence.* IEEE Press, Piscataway, NJ, 1995.

[57] D. B. Fogel, L. J. Fogel, W. Atmar, and G. B. Fogel. Hierarchic methods of evolutionary programming. In D. B. Fogel and W. Atmar, editors, *Proceedings of the 1^{st} Annual Conference on Evolutionary Programming*, pages 175–182, La Jolla, CA, 1992. Evolutionary Programming Society.

[58] D.B. Fogel and L.C. Stayton. On the effectiveness of crossover in simulated evolutionary optimization. *BioSystems*, 32:171–182, 1994.

[59] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution.* New York: John Wiley, 1966.

[60] S. Forrest and M. Mitchell. What makes a problem hard for a genetic algorithms : Some anomalous results and their explanation. *Machine Learning*, pages 285–319, 1993.

[61] R. Garigliano, A. Purvis, P. A. Giles, and D. J. Nettleton. Genetic algorithms and shape representation. In D. B. Fogel and W. Atmar, editors, *Proceedings of the 2^{nd} Annual Conference on Evolutionary Programming*, pages 40–47. Evolutionary Programming Society, 1993.

[62] P.L. George. *Automatic mesh generation, application to Finite Element Methods.* Wiley & Sons, 1991.

[63] C. Ghaddar, Y. Maday, and A. T. Patera. Analysis of a part design procedure. *Submitted to Nümerishe Mathematik*, 1995.

[64] A. Giordana, L. Saitta, and F. Zini. Learning disjunctive concepts by means of genetic algorithms. In Cohen W. and Hirsh H., editors, *Proceedings of ICML-94, International Conference on Machine Learning*, pages 96–104. Morgan Kaufmann, 1994.

[65] J. Gittus and J. Zarka, editors. *Modelling small deformations of polycrystals.* Elsevier Applied Sciences Publishers, 1985.

[66] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning.* Addison Wesley, 1989.

[67] D. E. Goldberg and K. Deb. A comparative study of selection schemes used in genetic algorithms. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, 1991.

[68] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multi-modal function optimization. In J. J. Grefenstette, editor, *Proceedings of the 2^{nd} International Conference on Genetic Algorithms*, pages 41–49. Lawrence Erlbaum Associates, 1987.

[69] Y. Grandvalet and S. Canu. Comments on "noise injection into inputs in back propagation learning" [111]. *IEEE transactions on Systems, Man and Cybernetics*, 1995.

[70] J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-16, 1986.

[71] J. J. Grefenstette. Virtual genetic algorithms: First results. Technical Report AIC-95-013, Navy Center for Applied Research in Artificial Intelligence, February 1995.

[72] L. Gueguen. Algorithmes génétiques et l-systèmes pour l'optimisation de formes, Sept. 1995. Rapport de DEA d'Informatique de l'Université de Paris XI.

[73] P. Harouni. Optimisation par réseaux neuro-génétiques, Juin 1992. Rapport de DEA d'Analyse Numérique de l'Université de Paris VI.

[74] R. Hecht-Nielsen. *Neuro-computing*. Addison Wesley, 1989.

[75] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation*. Addison Wesley, 1991.

[76] W. D. Hillis. Co-evolving parasites improves simulated evolution as an optimization procedure. In C. G. Langton, editor, *Proceedings Artificial Life II*, pages 313–324. Addison Wesley, 1992.

[77] D.M. Himmelblau, J.T. Hsiung, and W. Suewatanakul. Should back propagation be replaced by more effective opimization algorithms ? *IEEE transactions on Neural Networks*, 1991.

[78] J. H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, 1975.

[79] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, pages 359–366, 1989.

[80] K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3:551–560, 1990.

[81] L. Ingber. Simulated annealing: Practive versus theory. *Math. Comput. Modelling*, 18:29–57, 1993.

[82] F. James. *Sur la modélisation mathématique des équilibres diphasiques et des colonnes de chromatographie*. PhD thesis, Ecole Polytechnique - France, 1990.

[83] F. James and M. Sepulveda. Parameter identification for a hyperbolic system arising in chemical engineering. *Inverse Problems*, 10(6):1299–1314, 1994.

[84] F. James and P. Valentin. *Récents progrès en Génie des Procédés*, chapter An analytical model of multicomponent isothermal adsorption equilibrium, pages 49–56. Adsorption Processes for Gas Separation, CNRS-NSF Gif 91, 1991.

[85] C. Z. Janikow. A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning*, 13:189–228, 1993.

[86] C. Z. Janikow and Z. Michalewicz. An experimental comparison of binary and floating point representations in genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of 4th International Conference on Genetic Algorithms*, pages 31–36. Morgan Kaufmann, July 1991.

[87] E. Jensen. *Topological Structural Design using Genetic Algorithms*. PhD thesis, Purdue University, November 1992.

[88] T. Jones. Crossover, macromutation and population-based search. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 73–80. Morgan Kaufmann, 1995.

[89] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann, 1995.

[90] F. Jouve. *Modélisation mathématique de l'œil en élasticité non-linéaire*, volume RMA 26. Masson Paris, 1993.

[91] B. A. Julstrom. What have you done for me lately? adapting operator probabilities in a steady-state genetic algorithm on genetic algorithms. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 81–87. Morgan Kaufmann, 1995.

[92] A. B. Kahng and B. R. Moon. Toward more powerful recombinations. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 96–103. Morgan Kaufmann, 1995.

[93] C. Kane. *Algorithmes génétiques et Optimisation topologique.* PhD thesis, Université de Paris VI, July 1996.

[94] C. Kane, F. Jouve, and M. Schoenauer. Structural topology optimization in linear and nonlinear elasticity using genetic algorithms. In *Proceedings of the ASME 21st Design Automation Conference*. ASME, Boston, Sept. 1995.

[95] C. Kane and M. Schoenauer. Genetic operators for two-dimensional shape optimization. In J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Artificial Evolution*, number 1063 in LNCS. Springer Verlag, Septembre 1995.

[96] M. J. Keith and M. C. Martin. *Advances in Genetic Programming*, chapter Genetic Programming in C++: Implementation Issues, pages 285–310. MIT Press, Cambridge, MA, 1994.

[97] J. Kichenin, K. Dang-Van, and K. Boytard. Finite element simulation of new two dissipative mechanism models for bulk medium-density polyethylene. *J. of Mat. Sc.*, 31:1653–1661, 1996.

[98] J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs.* MIT Press, Massachussetts, 1994.

[99] J. R. Koza. *Genetic Programming: On the Programming of Computers by means of Natural Evolution.* MIT Press, Massachussetts, 1994.

[100] I. Langmuir. . *Jour. Am. Chem. Soc.*, 38:2221, 1916.

[101] M.A. Lee and H. Takagi. Dynamic control of genetic algorithms using fuzzy logic techniques. In S. Forrest, editor, *Proceedings of the* 5^{th} *International Conference on Genetic Algorithms*, pages 76–83. Morgan Kaufmann, 1993.

[102] R. G. Leriche, C. Knopf-Lenoir, and R. T. Haftka. A segragated genetic algorithm for constrained structural optimization. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 558–565, 1995.

[103] J. R. Levenick. Inserting introns improves genetic algorithm success rate : Taking a cue from biology. In R. K. Belew and L. B. Booker, editors, *Proceedings of the* 4^{th} *International Conference on Genetic Algorithms*, pages 123–127, 1991.

[104] H.-S. Lin, J. Xiao, and Z. Michalewicz. Evolutionary algorithm for path planning in mobile robot environment. In Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel, and H. Kitano, editors, *Proceedings of the First IEEE International Conference on Evolutionary Computation.* IEEE Press, June 1994.

[105] E. Lutton and P. Martinez. A genetic algorithm with sharing for the detection of 2d primitives in images. In J.-M. Alliot, E. Lutton, E. Ronald, and M. Schoenauer, editors, *Actes de la Conférence Evolution Artificielle.* Cepadues, Septembre 1994.

[106] S. W. Mahfoud. A comparison of parallel and sequential niching techniques. In L. J. Eshelman, editor, *Proceedings of the 6^{th} International Conference on Genetic Algorithms*, pages 136–143. Morgan Kaufmann, 1995.

[107] S. W. Mahfoud and D. E. Goldberg. A genetic algorithm for parallel simulated annealing. In R. Manner and B. Manderick, editors, *Proceedings of the 2^{nd} Conference on Parallel Problems Solving from Nature*, pages 301–310. Springer Verlag, 1992.

[108] S. Martin, A. Brunet-Bruneau, J. Rivory, and M. Schoenauer. Darwinian evolution of homogeneous multi-layer systems: a new method for optical coatings design. In *Proceedings of EUROPTO 94*. SPIEE, June 1994.

[109] S. Martin, J. Rivory, and M. Schoenauer. Simulated darwinian evolution of homogeneous multi-layer systems: a new method for optical coatings design. *Optics Communications*, 110(1):503–506, September 1994.

[110] S. Martin, J. Rivory, and M. Schoenauer. Synthesis of optical multi-layer systems using genetic algorithms. *Applied Optics*, 34:2267, 1995.

[111] K. Matsuoka. Noise injection into inputs in back propagation learning. *IEEE transactions on Systems, Man and Cybernetics*, 22(3):436–440, 1992.

[112] E. Mazer, M. Ahuactzin, E. Talbi, and P. Bessiere. The ariadne's clew algorithm. In J.-A. Meyer, H. L. Roitblat, and S. W. Wilson, editors, *From aniamls to animats 2, Proceedings of the second International Conference on Simulated Adaptive Behavior*, pages 182–188. MIT Press, 1993.

[113] Z. Michalewicz. A hierarchy of evolution programs: An experimental study. *Evolutionary Computation*, 1(1):51–76, 1993.

[114] Z. Michalewicz. A survey of constraint handling techniques in evolutionary computation methods. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Proceedings of the 4^{th} Annual Conference on Evolutionary Programming*, 1995.

[115] Z. Michalewicz. *Genetic Algorithms+Data Structures=Evolution Programs*. Springer Verlag, New-York, 1996. 3rd edition.

[116] Z. Michalewicz, D. Dasgupta, R. Leriche, and M. Schoenauer. Evolutionary algorithms for constrained engineering problems. *Computers & Industrial Engineering Journal*, 30(2), April 1996.

[117] Z. Michalewicz and C. Z. Janikow. Handling constraints in genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the 4^{th} International Conference on Genetic Algorithms*, pages 151–157. Morgan Kaufmann, 1991.

[118] Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel, and H. Kitano, editors. *Proceedings of the First IEEE International Conference on Evolutionary Computation*. IEEE Press, June 1994.

[119] M. Minsky and S. Papert. *On perceptrons (New edition)*. MIT Press, 1988.

[120] M. Mitchell, S. Forrest, and J.H. Holland. The Royal Road for genetic algorithms : Fitness landscapes and GA performance. In F. J. Varela and P. Bourgine, editors, *Proceedings of the First European Conference on Artificial Life-93*, pages 245–254. MIT Press/Bradford Books, 1993.

[121] M. Mitchell and J.H. Holland. When will a genetic algorithm outperform hill-climbing ? In S. Forrest, editor, *Proceedings of the 5^{th} International Conference on Genetic Algorithms*, page 647, 1993.

[122] S. Muggleton. Inverse entailment and PROGOL. *New Gen. Comput.*, 13:245–286, 1995.

[123] D. Nguyen and B. Widrow. *Neural networks for control*, chapter The truck backer-upper: an example of self-learning in Neural Networks, pages 287–300. MIT Press, 1990.

[124] A.E. Nix and M.D. Vose. Modeling genetic algorithms with markov chains. *Annals of Mathematics and Artificial Intelligence*, 5(1):79–88, 1992.

[125] P. Nordin. *Advances in Genetic Programming*, chapter A Compiling Genetic Programming System that Directly Manipulates the Machine Code, pages 311–331. MIT Press, Cambridge, MA, 1994.

[126] R. W. Ogden. *Non-linear elastic deformations.* Ellis Horwood Ltd., 1984.

[127] U.-M. O'Reilly. *An Analysis of Genetic Programming.* PhD thesis, Carleton University, Ottawa, Ontario, Canada, 1995.

[128] J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, 1993.

[129] B. A. Pearlmutter. Gradient calculations for dynamic recurrent neural networks: A survey. *IEEE Transactions on Neural Networks*, 6:1212–1228, 1995.

[130] B. Persoz. *Introduction à la rhéologie.* Dunod, Paris, 1960.

[131] F. P. Preparata and M. I. Shamos. *Computational Geometry: an introduction.* Springer Verlag, 1985.

[132] P. Prusinkiewicz and A. Lindenmeyer. *The Algorithmic Beauty of Plants.* Springer Verlag, New York, 1990.

[133] Yinglin Y. Qing J., Wei Z. Approximation property of multilayer neural network and its application in nonlinear simulation. *IEEE transactions on Neural Networks*, 1991.

[134] J.R. Quinlan and R.M. Cameron-Jones. FOIL : A midterm report. In P. Bradzil, editor, *Proceedings of ECML93*, pages 3–20. Springer-Verlag, 1993.

[135] N. J. Radcliffe. Equivalence class analysis of genetic algorithms. *Complex Systems*, 5:183–20, 1991.

[136] N. J. Radcliffe and P. D. Surry. Formal memetic algorithms. In T.C. Fogarthy, editor, *Evolutionary Computing: AISB Workshop*, pages 1–16. Springer Verlag LNCS 865, 1994.

[137] N. J. Radcliffe and P. D. Surry. Fitness variance of formae and performance prediction. In L. D. Whitley and M. D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 51–72. Morgan Kaufmann, 1995.

[138] I. Rechenberg. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution.* Fromman-Holzboog Verlag, Stuttgart, 1973.

[139] E. Ronald. When selection meets seduction. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 167–173. Morgan Kaufmann, 1995.

[140] E. Ronald and M. Schoenauer. Genetic lunar lander : an experiment in accurate neuro-control. In Y. Davidor, H.-P. Schwefel, and R. Manner, editors, *Proceedings of the* 3^{rd} *Conference on Parallel Problems Solving from Nature*, pages 452–461. Springer-Verlag, LNCS 866, 1994.

[141] P. Rouchon, M. Fliess, J. Lévine, and P. Martin. Flatness and motion planning: the car with n trailers. In *Proc. ECC'93, Groningen*, pages 1518–1522, 1993.

[142] P. Rouchon, M. Schoenauer, P. Valentin, and G. Guiochon. *Numerical Simulation of Band Propagation in Nonlinear Chromatography*, volume 46 of *Chromatographic Science*, chapter 1, pages 1–43. Marcel Dekker Inc., 1984.

[143] G. Rudolph. Convergence analysis of canonical genetic algorithm. *IEEE Transactions on Neural Networks*, 5(1):96–101, 1994.

[144] G. Rudolph. Convergence of non-elitist strategies. In Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel, and H. Kitano, editors, *Proceedings of the First IEEE International Conference on Evolutionary Computation*, pages 63–66. IEEE Press, 1994.

[145] D.E. Rumelhart and J.L. McClelland. *Parallel Distributed Processing*. MIT Press, 1986.

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

[147] J.D. Schaffer and A. Morishima. An adaptive crossover distribution mechanism for genetic algorithms. In J. J. Grefenstette, editor, *Proceedings of the* 2^{nd} *International Conference on Genetic Algorithms*, pages 36–40. Morgan Kaufmann, 1987.

[148] M. Schoenauer. Representations for evolutionary optimization and identification in structural mechanics. In J. Périaux and G. Winter, editors, *Genetic Algorithms in Engineering and Computer Sciences*, pages 443–464. John Wiley, 1995.

[149] M. Schoenauer and E. Ronald. Genetic extension of neural net training: transfer functions and renormalisation coefficients. In J.-M. Alliot, E. Lutton, E. Ronald, and M. Schoenauer, editors, *Actes de la Conférence Evolution Artificielle*. Toulouse, Cepadues, September 1994.

[150] M. Schoenauer and E. Ronald. Neuro-genetic truck backer-upper controller. In *Proceedings of the First IEEE International Conference on Evolutionary Computation*. IEEE Press, June 1994.

[151] M. Schoenauer, E. Ronald, and S. Damour. Evolving nets for control. In *Proceedings of Neuro-Nimes*. EC2, November 1993.

[152] M. Schoenauer, M. Sebag, F. Jouve, B. Lamy, and H. Maitournam. 23: Evolutionary identification of macro-mechanical models. In P. J. Angeline and Jr K. E. Kinnear, editors, *Advances in Genetic Programming II*, pages 467–488, Cambridge, MA, 1996. MIT Press.

[153] M. Schoenauer and Z. Wu. Discrete optimal design of structures by genetic algorithms. In Bernadou and al., editors, *Conférence Nationale sur le Calcul de Structures*, pages 833–842. Hermes, 1993.

[154] M. Schoenauer and S. Xanthakis. Constrained GA optimization. In S. Forrest, editor, *Proceedings of the* 5^{th} *International Conference on Genetic Algorithms*, pages 573–580. Morgan Kaufmann, 1993.

[155] J. T. Schwartz and M. Sharir. A survey of motion planning and related geometric algorithms. *Artificial Intelligence*, 37:157–169, 1988.

[156] H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, New-York, 1981. 1995 – 2^{nd} edition.

[157] M. Sebag. Using constraints to building version spaces. In L. De Raedt and F. Bergadano, editors, *Proceedings of ECML-94, European Conference on Machine Learning*, pages 257–271. Springer Verlag, April 1994.

[158] M. Sebag. 2^{nd} order understandability of disjunctive version spaces. In *Workshop on Machine Learning and Comprehensibility, IJCAI-95*. Report LRI, Université Paris-Sud, 1995.

[159] M. Sebag, C. Ravisé, and M. Schoenauer. Controlling evolution by means of machine learning. In L. J. Fogel, P. J. Angeline, and T. Bäck, editors, *Proceedings of the* 5^{th} *Annual Conference on Evolutionary Programming*, pages 57–66. MIT Press, 1996.

[160] M. Sebag, C. Rouveirol, and J.F. Puget. Stochastic inductive constraint programming. *Machine Learning*, submitted.

[161] M. Sebag and M. Schoenauer. Controlling crossover through inductive learning. In Y. Davidor, H.-P. Schwefel, and R. Manner, editors, *Proceedings of the 3^{rd} Conference on Parallel Problems Solving from Nature*, pages 209–218. Springer-Verlag, LNCS 866, 1994.

[162] M. Sebag and M. Schoenauer. Contrôle d'un algorithme génétique. *Revue d'Intelligence Artificielle*, 10(2-3):389–428, 1996.

[163] M. Seguin. Optimisation de formes par évolution artificielle. etude de deux représentations, Dec. 1995. Rapport de DEA d'Analyse Numérique de l'Université de Paris VI.

[164] M. Shub and S. Smale. Complexity of bezout's theorem. i: Geometrical aspects. *Journal of the American Mathematical Society*, 6(2):459–501, 1993.

[165] P. Simard, B. Victorri, Y. Le Cun, and J. Denker. Tangent prop – a formalism for specifying selected invariances in a adaptative network. *Advances in Neural Information Processing Systems*, 4:895–903, 1992.

[166] K. Simms. Evolving virtual creatures. In *SIGGRAPH'94 Proceedings*, pages 15–22. Computer Graphics, Annual Conference Series, July 1994.

[167] W. M. Spears. Adapting crossover in a genetic algorithm. In R. K. Belew and L. B. Booker, editors, *Proceedings of the 4^{th} International Conference on Genetic Algorithms*. Morgan Kaufmann, 1991.

[168] L. Taieb. *Optimisation d'interféromètres par algorithmes génétiques*. PhD thesis, Ecole Polytechnique, 1997. To be defended in May.

[169] L. Taïeb and M. Schoenauer. Optimization of direction finders by genetic algorithms. In *Proceedings of GALESIA'95, Genetic Algorithms in Engineering Systems: Innovations and Applications*, pages 23–29. IEE, 1995.

[170] K. I. Trovato. *Application of learning and planning methods*, chapter Differential A^*: an adaptive search method illustrated with robot path planning for moving obstacles and goals in an uncertain environment. World Scientific Publishers, 1991.

[171] H. White. Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings. *Neural Networks*, 3:535–549, 1990.

[172] D. Whitley. Fundamental principles of deception in genetic search. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*. Morgan Kaufmann, 1991.

[173] O. C. Zienkiewicz. *The Finite Element Method in Engineering Science*. McGraw-Hill, New-York, 1st edition, 1967, 3rd edition, 1977.