Chapter 5

GENETIC/EVOLUTIONARY ALGORITHMS AND APPLICATION TO POWER SYSTEMS

Vladimiro Miranda, Luís Miguel Proença

INESC Porto, Instituto Engª Sist. Computadores, PORTUGAL
and FEUP - Fac. Engenharia Universidade do Porto
vmiranda@inescn.pt

also editing as support material the FAQ for comp.ai.genetic

by Joerg Heitkoetter and David Beasley
with permission of the authors

http://www.cis.ohio-state.edu/hypertext/faq/usenet/ai-faq/genetic/top.html

1. PREFACE

This chapter is intended to serve as base for understanding the universe of Evolutionary Computation (EC), only a small footpath to a more complex scientific universe, incorporating Fuzzy Systems and Artificial Neural Networks, sometimes referred to as Computational Intelligence (CI), and only part of an even more complex universe, incorporating Artificial Life, Fractal Geometry, and other Complex Systems Sciences, which might someday be referred to as Natural Computation (NC).

Over the course of the past years, Global Optimization algorithms imitating certain principles of nature have proved their usefulness in various domains of applications. Especially worth copying are those principles where nature has found "stable islands" in a "turbulent ocean" of solution possibilities. Such phenomena can be found in annealing processes, central nervous systems and biological evolution, and have lead to the following *optimization* methods: Simulated Annealing (SA), Artificial Neural Networks (ANNs) and the field of Evolutionary Computation (EC).

EC may currently be characterized by the following pathways: Genetic Algorithms (GA), Evolutionary Programming (EP), Evolution Strategies (ES), Classifier Systems (CFS), Genetic Programming (GP), and several other problem solving strategies, based upon biological observations dating back to Charles Darwin's discoveries in the 19th century - the means of natural Selection and survival of the fittest, and theories of Evolution. The inspired algorithms are thus termed Evolutionary Algorithms (EA).

The chapter will provide a basic introduction to EC and, particularly, with special focus on Power Systems. We are grateful to *Joerg Heitkoetter* and *David Beasley* for allowing us to use some interesting material they have compiled and edited with remarkable quality.

2. INTRODUCTION

2.1. What are Evolutionary Algorithms (EAs)?

Evolutionary Algorithm is an umbrella term used to describe computer-based problem solving systems which use computational models of some of the known Evolution mechanisms as key elements in their design and implementation. A variety of evolutionary algorithms have been proposed; they all share a common conceptual base of simulating the evolution of individual structures via processes of Selection, Mutation, and Reproduction. The processes depend on the perceived Performance of the individual structures as defined by an Environment.

More precisely, EAs maintain a Population of structures, that evolve according to rules of Selection and other operators, that are referred to as "search operators", (or Genetic Operators), such as Recombination and Mutation. Each *Individual* in the population receives a measure of its *Fitness* in the *Environment*. Reproduction focuses attention on high fitness individuals, thus exploiting the available fitness information. *Recombination* and *Mutation* perturb those individuals, providing general heuristics for Exploration. Although simplistic from a biologist's viewpoint, these algorithms are sufficiently complex to provide robust and powerful adaptive search mechanisms.

2.2. Biological Basis

To understand EAs, it is necessary to have some appreciation of the biological processes on which they are based.

Firstly, we should note that Evolution (in nature or anywhere else) is not a purposive or directed process. That is, there is no evidence to support the assertion that the goal of evolution is to produce Mankind. Indeed, the processes of nature seem to boil down to a haphazard

Generation of biologically diverse organisms. Some of evolution is determined by natural Selection or different Individuals competing for resources in the Environment. Some are better than others. Those that are better are more likely to survive and propagate their genetic material.

In nature, we see that the encoding for genetic information (Genome) is done in a way that admits asexual Reproduction. Asexual reproduction typically results in offspring that are genetically identical to the Parent. (Large numbers of organisms reproduce asexually; this includes most bacteria which some biologists hold to be the most successful Species known). Sexual Reproduction allows some shuffling of Chromosomes, producing offspring that contain a combination of information from each Parent. At the molecular level what occurs (wild oversimplification alert!) is that a pair of almost identical chromosomes bump into one another, exchange chunks of genetic information and drift apart. This is the Recombination operation, which is often referred to as Crossover because of the way that biologists have observed strands of chromosomes crossing over during the exchange.

Recombination happens in an Environment where the Selection of who gets to mate is largely a function of the Fitness of the Individual, i.e. how good the individual is at competing in its environment. Some "luck" (random effect) is usually involved too. Some EAs use a simple function of the fitness measure to select individuals (probabilistically) to undergo genetic operations such as Crossover or asexual Reproduction (the propagation of genetic material unaltered). This is fitness-proportionate Selection. Other implementations use a model in which certain randomly selected individuals in a subgroup compete and the fittest is selected. This is called Tournament Selection and is the form of Selection we see in nature when stags rut to vie for the privilege of mating with a herd of hinds.

Much EA research has assumed that the two processes that most contribute to Evolution are Crossover and Fitness based Selection/reproduction. As it turns out, there are mathematical proofs that indicate that the process of fitness proportionate Reproduction is, in fact, near optimal in some senses.

Evolution, by definition, absolutely requires diversity in order to work. In nature, an important source of diversity is Mutation. In an EA, a large amount of diversity is usually introduced at the start of the algorithm, by randomizing the Genes in the Population. The importance of Mutation, which introduces further diversity while the algorithm is running, therefore continues to be a matter of debate. Some refer to it as a background operator, simply replacing some of the original diversity which has been lost, while others view it as playing the dominant role in the evolutionary process.

It cannot be stressed too strongly that an evolutionary algorithm (as a Simulation of a genetic process) is not a random search for a solution to a problem (highly fit Individual). EAs use stochastic processes, but the result is distinctly non-random (better than random).

```
Algorithm EA is
                                    // start with an initial time
t := 0;
  // initialize a usually random population of individuals
initpopulation P (t);
 // evaluate fitness of all initial individuals in population
evaluate P (t);
       // test for termination criterion (time, fitness, etc.)
while not done do
                                    // increase the time counter
    t := t + 1;
        // select sub-population for offspring production
    P' := selectparents P (t);
            // recombine the "genes" of selected parents
    recombine P' (t);
          // perturb the mated population stochastically
    mutate P' (t);
                                    // evaluate it's new fitness
    evaluate P' (t);
               // select the survivors from present fitness
    P := survive P,P' (t);
                                    // done
end EA.
```

2.3. Genetic Algorithms (GA)

The Genetic Algorithm is a model of machine learning which derives its behavior from a metaphor of some of the mechanisms of Evolution in nature. This is done by the creation within a machine of a Population of Individuals represented by Chromosomes, in essence a set of character strings that are analogous to the base-4 chromosomes that we see in our own DNA. The individuals in the population then go through a process of simulated "evolution".

Genetic Algorithms are used for a number of different application areas. An example of this would be multidimensional optimization problems in which the character string of the Chromosome can be used to

encode the values for the different parameters being optimized.

In practice, therefore, we can implement this genetic model of computation by having arrays of bits or characters to represent the Chromosomes. Simple bit manipulation operations allow the implementation of Crossover, Mutation and other operations. Although a substantial amount of research has been performed on variable- length strings and other structures, the majority of work with Genetic Algorithms is focused on fixed-length character strings. We should focus on both this aspect of fixed-length and the need to encode the representation of the solution being sought as a character string, since these are crucial aspects that distinguish Genetic Programming, which does not have a fixed length representation and there is typically no encoding of the problem.

When the Genetic Algorithm is implemented it is usually done in a manner that involves the following cycle: Evaluate the Fitness of all of the Individuals in the Population. Create a new population by performing operations such as Crossover, fitness-proportionate Reproduction and Mutation on the individuals whose fitness has just been measured. Discard the old population and iterate using the new population.

One iteration of this loop is referred to as a Generation. There is no theoretical reason for this as an implementation model. Indeed, we do not see this punctuated behaviour in Populations in nature as a whole, but it is a convenient implementation model.

The first Generation (generation 0) of this process operates on a Population of randomly generated Individuals. From there on, the genetic operations, in concert with the Fitness measure, operate to improve the population.

In point 3 we'll refer to the Canonical Genetic Algorithm.

2.4. Positioning within optimization theory

GA have certain characteristics that differentiate them from traditional optimization methods. Here's a summary of the differences:

• GA code parameters in a bit string and not in the values of the parameters. The meaning of the bits is completely transparent for the GA.

• GA search from a population of points and not from a single point.

• GA use only the fitness function and don't need knowledge about derivatives or problem structure.

• GA use transition probabilistic rules (represented by the operators Selection, Crossover and Mutation) instead of deterministic rules.

2.5. References

D. E. Goldberg, "Genetic algorithms in Search, Optimization and Machine Learning", 1989 Addison-Wesley

Heitkoetter, Joerg and Beasley, David, eds. (1998) "The Hitch-Hiker's Guide to Evolutionary Computation: A list of Frequently Asked Questions (FAQ)", USENET: comp.ai.genetic. Available via anonymous FTP from rtfm.mit.edu/pub/usenet/news.answers/ai-faq/genetic/. About 110 pages.

3. HISTORICAL PERSPECTIVE

In this point we refer to the main steps in the field of Genetic Algorithms since their birth in 1962 by the hand of J. Holland.

Before that date some attempts were made in modeling genetic systems in computer systems (Barricelli 1957, 1962, Fraser 1960, 1962, Martin e Cockerham, 1960). However these studies' fundamental objective was to understand some biological phenomena.

John Holland and his students in the University of Michigan were the first to recognize the usefulness of using Genetic Operators in artificial adaptation problems. But it was Bagley in 1967 that first mentioned the expression "Genetic Algorithm" and the first to present a practical application of this knowledge.

In 1971, John Holland sets the basis for the theory behind the use of Genetic Algorithms: The Schema Theorem. In 1975 two important works were published: "Adaptation in Natural and Artificial Systems" of J. Holland and "An Analysis of the behavior of a class of Genetic Adaptive Systems" of De Jong. These works served as a base for the vast majority of subsequent studies.

In 1989, Goldberg published "Genetic Algorithms in Search, Optimization and Machine Learning" a reference book for Genetic Algorithms.

In the present Genetic Algorithms are reaching their adult phase, being used in several applications in different fields especially where conventional methods are not applicable.

3.1. References

J. H. Holland, " Adaption in Natural and Artificial Systems", Univ. of Michigan Press, 1975 Ann Arbor

D. E. Goldberg, "Genetic algorithms in Search, Optimization and Machine Learning", 1989 Addison-Wesley

4. CANONICAL GENETIC ALGORITHM

4.1. Introduction

Although there are many forms [Grefestette 91] for Genetic Algorithms, we will only refer the *canonical* algorithm. This means that we will be dealing with three genetic operators (Selection, Crossover and Mutation) and linear, binary, fixed-size chromosomes. Canonical GA use a fixed-size, non-overlapping population scheme and each new generation is created by the Selection operator and altered by Crossover and Mutation. The first population is generated at random.

4.2. Coding

Each chromosome represents a potential solution for the problem to solve and must be expressed in binary form. For instance, , in the integer interval $I=[0,31]$, we could simply code x in binary base, using 5 bits (such as 10010 or 00101). If we have a set of binary variables, each variable will be represented by a bit. For a multivariable problem, each variable has to be coded in the chromosome. As described below, all of the search process takes place at the coding level.

4.3. Fitness

Each solution must be evaluated by the fitness function to produce a value. This objective function characterizes the problem to be solved and, playing the role of environment, establishes the basis for Selection. If we want to maximize the function $f(x)=x^2$, with the above coding, the chromosome 11011 would receive the fitness value $27^2=729$, while the chromosome 00111 would receive the value $7^2=49$. The pair (chromosome, fitness) represents an *individual*.

In most cases, the fitness function can be assimilated to the objective function of a classical optimization problem. It will also include penalties for violated constraints.

The fitness function does not necessarily have to be in a mathematical form. It can be expressed also in qualitative form, and there are examples of GA models, in Power Systems, with fuzzy fitness function, such as in [Miranda 95]. It is traditional to assume that the fitness function is a monotone increasing function with the desirability of the solutions.

4.4. Selection

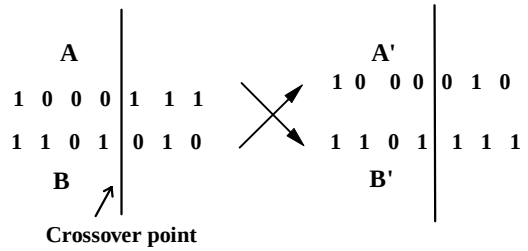
The *Selection* operator creates a new population (or *generation*) by selecting individuals from the old population, biased towards the best. This means that there will be more copies of the best individuals, although there may be some copies of the worst. This operator can be implemented in a variety of ways, although the most used techniques are those known as Stochastic Tournament and Roulette [Goldberg 91].

Stochastic Tournament - This implementation is suited to distributed implementations and is very simple: every time we want to select an individual for reproduction, we choose two, at random, and the best wins with some fixed probability, typically 0.8. This scheme can be enhanced by using more individuals on the competition [Goldberg 91] or even by considering evolving winning probability, eventually leading to *Boltzman Tournament* [Goldberg 91], generalizing the *Simulated Annealing* paradigm [Kirkpatrick 83].

Roulette – In this process, the individuals of each generation are selected for survival into the next generation according to a probability value proportional to the ratio of individual fitness over total population fitness; this means that on average the next generation will receive copies of an individual in proportion to the importance of its fitness value.

4.5. Crossover

The recombination operator used in the Canonical Genetic Algorithm is called Single Point Crossover. Individuals are paired at random with a high probability that Crossover will take place. In affirmative case, a Crossover point is selected at random and, say, the rightmost segments of each individual are exchanged to produce two offspring as illustrated in the next figure, where two 7-bit chromosomes **A** and **B** exchange parts (the Crossover point is p=4) resulting in the chromosomes **A'** and **B'**:



4.6. Mutation

In the Canonical Genetic Algorithm Mutation consists of simply flipping each individual bit with a very low probability (A typical value would be $P_m = 0.001$). This background operator is used to ensure that the probability of searching a particular subspace of the problem space is never zero, thereby tending to inhibit the possibility of ending the search at a local, rather than a global optimum.

4.7. Parameters

Like other optimization methods, GA have certain parameters like, for example:

- Population size;

- Genetic Operations Probabilities;

- Number of individuals involved in the Selection procedure, etc.

These parameters must be selected with maximum care, for the performance of GA depends largely on the values used. Normally, it's recommended the use of a relatively low population number, high Crossover and low Mutation probabilities. Goldberg [Goldberg 89] analyses the effect of these parameters in the algorithms.

4.8. How does a Genetic Algorithm work?

A canonical GA is a very simple process: we first generate a random initial population, evaluate it and start creating new populations by applying genetic operators. This high-level behavior can be depicted on the following piece of pseudo-C:

```
main()
{
     int gen;
     generate(oldpop);
     for(gen = 0; gen < MAXGEN;
gen++)
     {
          evaluate(oldpop);
          newpop = select(oldpop);
          Crossover(newpop);
          Mutation(oldpop);
          oldpop = newpop;
     }
}
```

Obviously, there is the need for some bookkeeping functions, for statistics and so on, but they are not central to this explanation.

This very simple behavior hides a powerful processing, done by the GA. In fact, the combination of Selection and Crossover leads to a proliferation of individuals that possess small, tightly coupled *blocks* of bits leading to good performance. These blocks, usually called *schemata* [Holland75], are replicated through Selection and combined or separated by Crossover.

And Mutation, what is its job? Mutation works as a kind of "life insurance". Some important bit values (genes) may be lost during Selection and Mutation can bring them back, if necessary. Nevertheless, too much Mutation can be harmful: a Mutation probability of 0.5 always leads to random search [Goldberg 89], independently of Crossover probability.

So, GA tends to select individuals with good performance and recombine some of their building blocks, creating more and more copies of good schemata, simply by the use of Selection and Crossover. This hidden processing is called *implicit parallelism* because the number of schemata processed in each generation is typically $\mathbf{O(}\boldsymbol{N}^{\mathbf{3}}\mathbf{)}$, being $\boldsymbol{N}$ the population size [Holland 75]. This compares very well with the number of fitness function evaluations, N. This characteristic is distinctive of Genetic Algorithms [Grefenstette 91].

It is very important to stress that a GA is not a "random search" for a solution to a problem (highly fit individual). The GA uses stochastic processes, but the result is distinctly non-random (better than random).

4.9. The Schemata Theorem

Why do GA work? There is an elegant answer to this question, based on the concept of *schema* (pl. *schemata*). A schema is a similarity template describing a subset of strings with similarities at certain string positions. If, without loss of generality, we consider only chromosomes represented with binary genes in {0,1}, a schema could be H = *001*1, where the character * is a “wild card”, meaning that the value of 0 or 1 at such position is undefined. The strings A = 100101 and B = 000111 include both the schema H because the string alleles match the schema positions 2,3,4 and 6.

For binary strings or chromosomes of length L (number of bits or alleles), the number of schemata is 3^L. But the schemata have distinct relevance; a schema 0***** is more vague than 011*1* in representing similarities between chromosomes; and a schema 1****0 spans a larger portion of the string than the schema 1**0**.

A schema H maybe characterized by its order o(H), which is the number of its fixed positions, and by its

defining length d(H), which is the distance between its first and its last fixed position. For the schema G = 1**0**, we have o(G) = 2 and d(G) = 3.

Let's now reason about the effect of reproduction the expected number of different schemata in a population. Suppose that at a given time step t (a given generation) there are *m* examples of a particular schema H in a population P(t); we have m = m(H,t). Reproduction generates a copy of a string A_i with probability $p_i = \frac{f_i}{\sum f_i}$ (assuming a sampling process known as "roulette"). After the process of retaining from the population **A**(t) a non-overlapping population of size n with replacement, there is an expectation of having in the population **A**(t+1), at time t+1, a number m(H, t+1) of representatives of the schema H given by

$$m(H, t+1) = m(H, t)\ n\ \frac{f(H)}{\sum f_i}$$

where f(H) is the average fitness of the chromosomes including schema H at time t. If we introduce the average fitness of the entire population as $\bar{f} = \frac{1}{n}\sum f_i$, we can write

$$m(H, t+1) = m(H, t)\ \frac{f(H)}{\bar{f}}$$

This means that a particular schema replicates in the population proportionally to the ratio of the average fitness of the schema by the average fitness of the population. So, schemata that have associated an average fitness above the population average will have more copies in the following generation, while those with average fitness below the population average will have a smaller number of copies.

Suppose now that a given schema remains with average fitness above the population average by an amount $c\bar{f}$, with c a constant; we could then re-write the above equation as

$$m(H, t+1) = m(H, t)\ \frac{\bar{f} + c\bar{f}}{\bar{f}} = (1+c)\ m(H, t)$$

Assuming a stationary value of c, we obtain

$$m(H, t) = m(H,0)(1+c)^t$$

The effect is clear: an exponential replication in a population of above-average schemata.

A schema may be disrupted by crossover, if the crossover point falls within the defining length spanned by the schema (let's reason with single point crossover, to keep it simple). The survival probability of a schema under a crossover operation performed with probability p_c is

$$p_s \geq 1 - p_c \frac{d(H)}{L-1}$$

Combining reproduction and crossover, we can write the following estimation

$$m(H, t+1) \geq m(H, t)\ \frac{f(H)}{\bar{f}}\left[1 - p_c \frac{d(H)}{L-1}\right]$$

We see now that the survival of a schema under reproduction and crossover depends on whether it is above or below the population average and whether it has a short or long definition length.

To add the effect of mutation, admitted affecting randomly a single position with probability p_m, we must notice that a schema survives if each of its o(H) fixed positions remain unaffected by mutation. Therefore, the probability of surviving mutation is $(1-p_m)^{o(H)}$, which can be approximated by 1 - o(H) p_m for $p_m << 1$.

We can conclude that in a process with reproduction, crossover and mutation, we can expect that a particular schema will have a number of copies in generation t+1 given approximately by

$$m(H, t+1) \geq m(H, t)\ \frac{f(H)}{\bar{f}}\left[1 - p_c \frac{d(H)}{L-1} - o(H)p_m\right]$$

This constitutes the *Schemata Theorem*, which is at the root of the "building block" hypothesis - that highly fit, short (low order) schemata form partial solutions to a problem, and that a GA will combine these building blocks leading to a better performance and to the optimum of the problem.

4.10. References

D. E Goldberg, "A Note on Boltzman Tournament Selection for Genetic Algorithms and Population-Oriented Simulated Annealing", 1991, Complex Systems 3

D. E Goldberg, K. Deb, "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms", Genetic algorithms in Search, Optimization and Machine Learning Summer School, 1991 Stanford

D. E. Goldberg, "Genetic algorithms in Search, Optimization and Machine Learning", 1989 Addison-Wesley

J. Grefenstette, "Conditions for Implicit Parallelism", Navy Center for Applied Research in Artificial Intelligence, Internal Report, 1991 Washington

J. H. Holland, " Adaption in Natural and Artificial Systems", The University of Michigan Press, 1975 Ann Arbor

Miranda, V., Proença, L.M., "A General Methodology For Distribution Planning Under Uncertainty, Including Genetic Algorithms And Fuzzy Models In A Multi-Criteria Environment", Proceedings of IEEE/KTH Stockholm Power Tech Conference, Stockholm, Sweden, June 18-22, 1995, pp. 832-837

S. Kirkpatrick et al. "Optimization by Simulated Annealing" 1983 Science

5. MAIN CHARACTERISTICS OF GENETIC ALGORITHMS

5.1. Advantages of Genetic Algorithms over traditional methods

The main advantages that GA present in comparison with conventional methods are:

- Since GA perform a search in a population of points and are based on probabilistic transition rules, they are less likely to converge to local minima (or maxima).

- GA do not require "well behaved" objective functions, easily tolerating discontinuities.

- GA are well adapted to distributed or parallel implementations.

Nevertheless, the power of conventional methods is recognized. GA should only be used only when it is impossible (or very difficult) to obtain efficient solutions by these traditional approaches.

5.2. Continuous and Discrete Variables

Real values can be approximated to the necessary degree by using a fixed point binary representation. However, when the relative precision of the parameters is more important than the absolute precision the logarithm of the parameters should be used instead. Alternatively, a floating-point binary representation can be used instead.

Discrete decision variables can be handled directly through binary (or n-ary) encoding. When functions can be expected to be locally monotone the use of Gray coding is known to better exploit that monotony.

5.3. Constraints

Most optimization problems are constrained in some way. GA can handle constraints in 2 ways, the most efficient of which is by embedding these in the coding of the chromosomes. When this is not possible, the performance of invalid individuals should be calculated according to a penalty function, which ensures that these individuals are, indeed, poor performers. appropriate penalty functions for a particular problem are not necessarily easy to design, since they may considerably affect the efficiency of the genetic search.

5.4. Multiobjective decision problems

Optimization problems very seldom require the optimization of a single objective function. Instead, there are often competing objectives which should be optimized simultaneously. In opposition to single objective optimization problems, the solution for a multiobjective optimization problem is not a single solution but a set of non-dominated solutions. The task of finding this set of solutions is not always an easy one. GA have the potential to become a powerful method for multiobjective optimization, keeping a population of solutions, and being able to search for non-dominated solution in parallel.

5.5. Other GA variants.

From *Genetic Algorithms in Control Systems Engineering*. Flemig P.J.; Fonseca, C. M., 1993

The simple Genetic Algorithm has been improved in several ways. Different Selection methods have been proposed (Baker, 1987), which reduce the stochastic errors associated with roulette wheel Selection. Ranking (Baker, 1985) has been introduced as an alternative to proportional fitness assignment, and shown to help avoidance of premature convergence and to speed up the search when the population approaches convergence (Whitley, 1989). Other recombination operators have been proposed, such as the multiple point and reduced-surrogate Crossover (Booker, 1987). The Mutation operator has remained more or less unaltered, but the use of real-coded chromosomes requires alternative Mutation operators (Davis, 1991). It has also motivated the development of non-conventional recombination operators, such as intermediate Crossover. Also, several models of parallel GA have been proposed improving the performance and allowing the implementation of

concepts like the one of "Geographical Isolation".

5.6. References

Baker, J. E. et al.(1985) *Adaptive Selection Methods for Genetic Algorithms*, in Proc. 1st Int. Conference on Genetic Algorithms.

Baker, J. E. et al.(1987) *Reducing bias and inefficiency in the Selection algorithm* in Proc. 2nd Int. Conference on Genetic Algorithms.

Booker, L. (1987) *Improving search in Genetic Algorithms*. In : Genetic Algorithms and Simulated annealing (L. Davis Ed.)

Davis L. Ed. *Handbook of Genetic Algorithms*. van Nostrand Reinhold, New York.

Whitley, D (1989) *The GENITOR algorithm and Selection pressure: Why rank-based allocation of reproductive trials is best.* in Proc. 3rd Int. Conference on Genetic Algorithms

6. EA APPLICATIONS

6.1. Introduction

Although Evolutionary Algorithms and, in particularly, Genetic Algorithms are a relatively new technique, there are applications in several fields and, in particular, in engineering.

In principle, EAs can compute any computable function, i.e. everything a normal digital computer can do. But EAs are especially badly suited for problems where efficient ways of solving them are already known, (unless these problems are intended to serve as benchmarks). Special purpose algorithms, i.e. algorithms that have a certain amount of problem domain knowledge hard coded into them, will usually outperform EAs, so there is no black magic in EC. EAs should be used when there is no other known problem solving strategy, and the problem domain is NP-complete. That's where EAs come into play: heuristically finding solutions where all else fails.

Some of the most successful EA applications are listed below.

6.2. Control System Engineering

One of the fields in which GA have been applied with considerable success is Control Engineering. In this field, the problems that are presented to engineers are of high complexity and, almost always, multiobjective. Genetic Algorithms are a powerful toll especially when used together with other existing tools. See [Fleming1993] for more details.

6.3. Timetabling

This has been addressed quite successfully with GAs. A very common manifestation of this kind of problem is the timetabling of exams or classes in Universities, etc.

The first application of GAs to the timetabling problem was to build the schedule of the teachers in an Italian high school. The research, conducted at the Department of Electronics and Information of Politecnico di Milano, Italy, showed that a GA was as good as Taboo Search, and better than simulated annealing, at finding teacher schedules satisfying a number of hard and soft constraints. The software package developed is now in current use in some high schools in Milano. (Colorni et al 1990)

At the Department of Artificial Intelligence, University of Edinburgh, timetabling the MSc exams is now done using a GA (Corne et al. 93, Fang 92). An example of the use of GAs for timetabling classes is (Abramson & Abela 1991).

In the exam timetabling case, the Fitness function for a Genome representing a timetable involves computing degrees of punishment for various problems with the timetable, such as clashes, instances of students having to take consecutive exams, instances of students having (e.g.) three or more exams in one day, the degree to which heavily-subscribed exams occur late in the timetable (which makes marking harder), overall length of timetable, etc. The modular nature of the fitness function has the key to the main potential strength of using GAs for this sort of thing as opposed to using conventional search and/or constraint programming methods. The power of the GA approach is the ease with which it can handle arbitrary kinds of constraints and objectives; all such things can be handled as weighted components of the fitness function, making it easy to adapt the GA to the particular requirements of a very wide range of possible overall objectives. Very few other timetabling methods, for example, deal with such objectives at all, which shows how difficult it is (without GAs) to graft the capacity to handle arbitrary objectives onto the basic "find shortest- length timetable with no clashes" requirement. The proper way to weight/handle different objectives in the fitness function in relation to the general GA dynamics remains, however, an important research problem!

GAs thus offer a combination of simplicity, flexibility & speed which competes very favorably with other

approaches, but are unlikely to outperform knowledge-based (etc.) methods if the best possible solution is required at any cost. Even then, however, hybridization may yield the best of both worlds; also, the ease (if the hardware is available!) of implementing GAs in parallel enhances the possibility of using them for good, fast solutions to very hard timetabling and similar problems.

6.4. Job-Shop Scheduling

The Job-Shop Scheduling Problem (JS) is a very difficult NP-complete problem which, so far, seems best addressed by sophisticated branch and bound search techniques. GA researchers, however, are continuing to make progress on it. (Davis 85) started off GA research on the JS, (Whitley 89) reports on using the edge Recombination operator (designed initially for the TSP) on JSSPs too. More recent work includes (Nakano 91),(Yamada & Nakano 92), (Fang et al. 93). The latter three report increasingly better results on using GAs on fairly large benchmark JSSPs (from Muth & Thompson 63); neither consistently outperform branch & bound search yet, but seem well on the way. A crucial aspect of such work (as with any GA application) is the method used to encode schedules. An important aspect of some of the recent work on this is that better results have been obtained by rejecting the conventional wisdom of using binary representations (as in (Nakano 91)) in favor of more direct encoding. In (Yamada & Nakano 92), for example, a Genome directly encodes operation completion times, while in (Fang et al. 93) genomes represent implicit instructions for building a schedule. The success of these latter techniques, especially since their applications are very important in industry, should eventually spawn advances in GA theory.

Concerning the point of using GAs at all on hard job-shop scheduling problems, the same goes here as suggested above for `Timetabling': The GA approach enables relatively arbitrary constraints and objectives to be incorporated painlessly into a single optimization method. It is unlikely that GAs will outperform specialized knowledge-based and/or conventional OR-based approaches to such problems in terms of raw solution quality, however GAs offer much greater simplicity and flexibility, and so, for example, may be the best method for quick high-quality solutions, rather than finding the best possible solution at any cost. Also, of course, hybrid methods will have a lot to offer, and GAs are far easier to parallelize than typical knowledge-based/OR methods.

Similar to the JS is the Open Shop Scheduling Problem (OSSP). (Fang et al. 93) reports an initial attempt at using GAs for this. Ongoing results from the same source shows reliable achievement of results within less than 0.23% of optimal on moderately large OSSPs (so far, up to 20x20), including an improvement on the previously best known solution for a benchmark 10x10 OSSP. A simpler form of job shop problem is the Flow-Shop Sequencing problem; recent successful work on applying GAs to this includes (Reeves 93)."

In contrast to job shop scheduling some maintenance scheduling problems consider which activities to schedule within a planned maintenance period, rather than seeking to minimize the total time taken by the activities. The constraints on which parts may be taken out of service for maintenance at particular times may be very complex, particularly as they will in general interact. Some initial work is given in (Langdon, 1995).

6.5. Management Sciences

A good reference on this subject is the work from Volker Nissen: *Evolutionary Algorithms in Management Science - An overview and list of references*. Se also *Boulding 1991*.

Also, Genetic Algorithms have been successful used in market forecasting. The most well known system is PREDICTO (http://www.quote.com/newsletters/predicto/).

Also on the subject,:

Bauer, R. Jr. (1994) *Genetic Algorithms and Investment Strategies*. New York: John Wiley and Sons.

6.6. Game Playing

GAs can be used to evolve behaviors for playing games. Work in evolutionary Game Theory typically surrounds the Evolution of a Population of players who meet randomly to play a game in which they each must adopt one of a limited number of moves. (Maynard-Smith 1982). Let's suppose it is just two moves, X and Y. The players receive a reward, analogous to Darwinian Fitness, depending on which combination of moves occurs and which move they adopted. In more complicated models there may be several players and several moves.

The players iterate such a game a series of times, and then move on to a new partner. At the end of all such moves, the players will have a cumulative payoff, their Fitness. This fitness can then be used as a means of conducting something akin to Roulette-Wheel Selection to generate a new Population.

The real key in using a GA is to come up with an encoding to represent player's strategies, one that is amenable to Crossover and to Mutation. possibilities are to suppose at each iteration a player adopts X with some probability (and Y with one minus such). A player can thus be represented as a real number, or a bit-string by interpreting the decimal value of the bit string as the inverse of the probability.

6.7. General References.

Abramson & Abela (1991) "A Parallel Genetic Algorithm for Solving the School Timetabling Problem", Technical Report, Division of I.T., C.S.I.R.O, April 1991. (Division of Inf. Technology, C.S.I.R.O., c/o Dept. of Communication & Electronic Engineering, Royal Melbourne Institute of Technology, PO BOX 2476V, Melbourne 3001, Australia)

Boulding, K.E. (1991) "What is evolutionary economics?", Journal of Evolutionary Economics, 1, 9-17.

Colorni A., M. Dorigo & V. Maniezzo (1990). A Genetic Algorithm to Solve the Timetable Problem. Technical Report No. 90-060, Politecnico di Milano, Italy.

Colorni A., M. Dorigo & V. Maniezzo (1990). Genetic Algorithms And Highly Constrained Problems: The Time-Table Case. Proceedings of the First International Workshop on Parallel Problem Solving from Nature, Dortmund, Germany, Lecture Notes in Computer Science 496, Springer- Verlag, 55-59.

lorni A., M. Dorigo & V. Maniezzo (1990). Genetic Algorithms: A New Approach to the Time-Table Problem. NATO ASI Series, Vol.F 82, COMBINATORIAL Optimisation, (Ed. M.Akguel and others), Springer- Verlag, 235-239. http://iridia.ulb.ac.be/dorigo/dorigo/conferences/IC.02-NATOASI90.ps.gz

Corne, D. Fang, H.-L. & Mellish, C. (1993) "Solving the Modular Exam Scheduling Problem with Genetic Algorithms". Proc. of 6th Int'l Conf. on Industrial and Engineering Applications of Artificial Intelligence & Expert Systems, ISAI.

Davis, L. (1985) "Job-Shop Scheduling with Genetic Algorithms", [ICGA85], 136-140.

Fang, H.-L. (1992) "Investigating GAs for scheduling", MSc Dissertation, University of Edinburgh Dept. of Artificial Intelligence, Edinburgh, UK.

Fang, H.-L., Ross, P., & Corne D. (1993) "A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling & Open-Shop Scheduling Problems", [ICGA93], 375-382.

Fleming Peter J., Fonseca C. M., "Genetic algorithms in control systems engineering", 1993

Fogel (1993) "Evolving Behaviour in the Iterated Prisoner's Dilemma" Evolutionary Computation 1:1, 77-97

Fogel, D.B. and Harrald, P. (1994) ``Evolving Complex Behaviour in the Iterated Prisoner's Dilemma," Proceedings of the Fourth Annual Meetings of the Evolutionary Programming Society, L.J. Fogel and A.W. Sebald eds., World Science Press.

Holland, J.H and John Miller (1990) ``Artificially Adaptive Agents in Economic Theory," American Economic Review: Papers and Proceedings of the 103rd Annual Meeting of the American Economics Association: 365--370.

http://iridia.ulb.ac.be/dorigo/dorigo/conferences/IC.01-PPSN1.ps.gz

Huberman, Bernado, and Natalie S. Glance (1993) "Diversity and Collective Action" in H. Haken and A. Mikhailov (eds.) Interdisciplinary Approaches to Nonlinear Systems, Springer.

Lindgren, K. (1991) ``Evolutionary Phenomena in Simple Dynamics," in [ALIFEI].

Lindgren, K. and Nordahl, M.G. "Cooperation and Community Structure in Artificial Ecosystems", Artificial Life, vol 1:1&2, 15-38

Marimon, Ramon, Ellen McGrattan and Thomas J. Sargent (1990) ``Money as a Medium of Exchange in an Economy with Artificially Intelligent Agents" Journal of Economic Dynamics and Control 14, pp. 329--373.

Maynard-Smith, (1982) Evolution and the Theory of Games, CUP.

Muth, J.F. & Thompson, G.L. (1963) "Industrial Scheduling". Prentice Hall, Englewood Cliffs, NJ, 1963.

Nakano, R. (1991) "Conventional Genetic Algorithms for Job-Shop Problems", [ICGA91], 474-479.

Reeves, C.R. (1993) "A Genetic Algorithm for Flowshop Sequencing", Coventry Polytechnic Working Paper, Coventry, UK.

Stanley, E.A., Ashlock, D. and Tesfatsion, L. (1994) "Iterated Prisoners Dilemma with Choice and Refusal of

Partners in [ALIFEIII] 131-178

Whitley, D., Starkweather, T. & D'Ann Fuquay (1989) "Scheduling Problems and Traveling Salesmen: The Genetic Edge Recombination Operator", [ICGA89], 133-140.

Yamada, T. & Nakano, R. (1992) "A Genetic Algorithm Applicable to Large-Scale Job-Shop Problems", [PPSN92], 281-290.

Table 1 - Number of papers including Evolutionary Algorithms in applications to Power Systems, up to 1996, according to a survey (Miranda 96)

AREA	FIELD	N. REFs
A. Expansion or structural planning	A1. Generation-transmission	6
	A2. Transmission-distribution	18
	A3. VAr planning, capacitor placement	5
B. Operation planning	B1. Unit commit., generator scheduling	6
	B2. Load dispatch	21
	B3. React. pw. dispatch, volt. control	7
	B4. Maintenance Scheduling	3
	B5. Security Assessment	5
C. Generation/ transmission and distribution operation	C1. Loss minimization, switching	3
	C2. Alarm processing, fault diagnosis	10
	C3. Service restoration	4
	C4. Load management	1
	C5. Load forecasting	4
	C6. State estimation	3
	C7. FACTS	3
D. Analysis	D1. Power Flow	8
	D2. Harmonics	5
E. Control	E1. Parameter estimation and tuning	6
F. Surveys		3

6.8. Power Systems

There have been some attempts to apply Genetic Algorithms to solve problems in the Power System area. In the references one may find the reports of attempted approaches to problems such as:

- Clustering and Network Reduction;
- Optimal Capacitor Placement;
- Voltage Optimization;
- Harmonics;
- System Observability;
- Reactive Power Control;
- Load Flow Analysis;
- Distribution Network Planning
- Unit Commitment

A recent survey (Miranda 96) has covered the diverse sub-fields of Power System modeling and listed over a hundred important publications on GA and other EC applications in those areas.

Table 1 gives a picture of the broad attempts of use of Evolutionary Computing techniques in Power System applications. The numbers in the column N.REFs refer to the number of publications identified in 1996 in major reviews and conference proceedings - the big majority being with GA. Since that date, many more publications have come to light.

6.9. Some references in Power System applications

Ferreira, J.R., Peças Lopes, J.A., Saraiva, J.T., “Identification of preventive control procedures to avoid voltage collapse using genetic algorithms”, appearing in Proceedings of PSCC’99 – Power System Computation Conference, Trondheim, Norway, 1999

H. Ding et al, "Optimal clustering of power networks using genetic algorithms", Proc. 3rd Biennial Symp. Indust. Elect. Applications, Ruston, LA, USA, LA Tech. Univ., 1992

H. Mori, "A genetic approach to power system topological observability", Proc. IEEE International Symp. on Circuits and Systems, New York, NY, USA, IEEE 1991

H. Yang, "Worst case analysis of distribution system harmonics using genetic algorithms", Proc. IEEE SOUTHEASTCON '92 New York, NY, USA, IEEE 1992

K. Iba, "Reactive Power Optimisation by Genetic Algorithm", PICA'93, Phoenix, AR, USA, May 1993

Langdon, W.B. (1995) "Scheduling Planned Maintenance of the (UK) National Grid", cs.ucl.ac.uk:/genetic/papers/grid_aisb-95.ps

Miranda, V., Proença, L.M., Ranito, J.V., "Genetic Algorithms in Optimal Multistage Distribution Network Planning", IEEE Transactions on Power Systems, November 1994.

Miranda, V., Srinivasan, D., Proença, L.M., “Evolutionary computation in Power Systems”, Proceedings of PSCC 96 - 12th Power System Computation Conference, Dresden, Germany, Aug.19-

23 1996

Miranda, V., Proença, L.M., “Dynamic planning of distribution networks including dispersed generation”, Proceedings of CIRED ARGENTINA’96, Session 4 (Planning), Dec. 2-5, Buenos Aires, Argentina, 1996

T. Haida et al., "Genetic algorithms approach to voltage optimization", Proc. First International Forum on Aplications of Neural Networks to Power Systems, New York, NY, USA, IEEE 1991

V. Ajjarapu et al., "Application of genetic based algorithms to optimal capacitor placement", Proc. First International Forum on Applications of Neural Networks to Power Systems, New York, NY, USA, IEEE 1991

Wang, L.C., Proença, L.M., Miranda, V., “Demonstrating an efficient capacitor location and sizing method for distribution systems with the Macau network”**,** Proceedings of ELAB’96 - Encontro Luso-Afro-Brasileiro para o Planeamento e Exploração de Redes de Energia, Oct. 16-18, Porto, Portugal, 1996

X. Yin, N. Germany, "Investigations on Solving the Load Flow Problem by Genetic Algorithms", Electric Power Systems research,22 (1991), 1991 Elsevier

7. EXAMPLES OF APPLICATION OF GA IN POWER SYSTEMS

In this section we will present a few simplified versions of models applied to Power Systems, with the purpose of helping the readers to understand more clearly the potential behind the GA principles.

7.1. Unit Commitment

The project CARE, developed within the European Union research programme JOULE-THERMIE, aims at developing and testing some new concepts for the operation and control of isolated or weak networks with wind generation.

The project is conducted by a consortium of research institutions from France (Ecole de Mines de Paris), Greece (NTUA), Portugal (INESC) and the UK (RAL), and utilities from Greece (PPC) and Portugal (EDA). One of the modules included is a Unit Commitment/Generator Scheduling application where solutions are searched for by a set of GA models. We will shortly describe the Long Term model, which prepares a schedule for the next 48 hours, in 1 hour steps, and is run (updated) every hour.

The unit commitment procedure (developed at INESC, Portugal) searches for global optimal solutions considering the following costs and constraints:

- unit shutdown and start-up costs
- operation costs (usually, fuel costs)
- ramping costs
- load satisfaction
- ramping constraints
- spinning reserve criteria
- maximum wind penetration
- minimum unit down times
- minimum start up times (as function of down time)

7.1.1. Coding

Usually, in a period of 48 hours, one generator will only change state a few times. Therefore, one would only need a few bits to denote the change of state, instead of 48 bits (one for each hour, if the time step is 1 hour). Moreover, during long periods, the same set of generators may be on or off - so, they can be represented in block.

We have therefore divided the period of scheduling into a number b of blocks, where we admit that the composition of generators on and off is constant (requiring only a number of bits equal to the number of generators). We have also added to each block a number c of bits defining an advance or delay for each one.

There is also one extra bit per block allowing a block to be disabled, so that the number of active blocks is variable and adapts to the solution.

The number of bits to express a solution is therefore given by bx(n+c+1), and the number b of blocks needed depends on the regularity of the load curve.

For instance, in a problem with 25 generators and a hourly scheduling for 48 hours (a long-term UC), the direct encoding of every hour would require a chromosome with 1200 bits, while the same problem divided into 10 blocks (with 2 bits for advance or delay) would require chromosomes of 280 bits in length only.

This structure is depicted in Figure 1. Each block corresponds to a different number of hours, depending on the shape of the load curve.

Presently, the number of blocks and the relative size of each block are decisions taken at the modeling stage of the problem, by the operator (not done automatically).

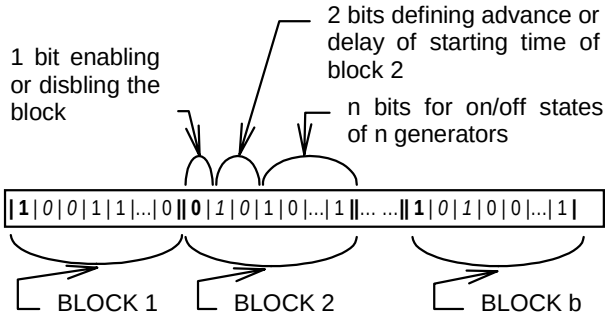


Figure 1 - Structure of chromosome coding in the Unit Commitment module of the CARE project

7.1.2. Selection and deterministic crowding

Selection is directed under the deterministic crowding (DC) approach (Mahfoud 95). The DC will randomly pair all population elements in each generation. Each pair of parents will undergo mutation, crossover and yield two children. The members of each set of two parents and two children are then classified and paired according a measure of similarity — in this case, the Hamming distance between chromosomes (the number of differences in the bits). Finally, competition is established in each pair and the two fittest elements will enter the next generation. DC can successfully perform niching and preserve the diversity in the GA.

7.1.3. Neighborhood digging

At certain generations, a process called "neighborhood digging" (ND) is launched. It tries to displace the start up and shut down times of generators in order to improve solutions, selected with a given probability.

The general ND tests are:

- enabling/disabling some blocks
- shutting down or starting up generators within some blocks
- anticipating/postponing the start of a new block
- avoiding unnecessary shutdowns/start ups of generators

A set of rules have been derived to guide the ND process, for instance for postponement and anticipation of start up and shutdown of generators, in order to improve the computational efficiency of the process. The decision to adopt changes in the solutions may come either from feasibility verification or from fitness calculation. If better solutions are found, their chromosome coding is added to the genetic pool.

The ND procedure has been extensively tested, namely against a Simple Genetic Algorithm (where the neighborhood search is disabled) and its superior performance has been confirmed.

7.1.4. Chromosome repair

Chromosome repair (CR) is applied to unfeasible solutions, in an attempt to bring them into feasibility. It consists of a series of very simple and heuristic tests, so that the process remains computationally efficient. If a feasible solution is built, it replaces the original unfeasible one in the genetic pool.

7.1.5. Crossover

The module adopts 2-point crossover. In this Unit Commitment problem, with the type of chromosome coding adopted, it proved to lead to a superior performance of the algorithm.

7.1.6. Mutation

Low probability random mutation is adopted. The mutation rate may change (increase) for a number of generations if no improvement is found in the best solution available.

7.1.7. Fitness

Fitness is calculated from the sum of all costs and penalties.

The costs are related to start-up and shut down procedures, and to operation costs in each hour. These latter costs depend on the calculation of an optimal dispatch for the generation in each period, subject to constraints such as meeting the load and conforming to operation windows or ramping limits of the generators.

A special attention is devoted to the presence of wind generation. An original model has been developed to include in the classical dispatch algorithm with Lagrange multipliers a fuzzy constraint on wind penetration allowed, to take in account the dynamic security of the system in case of wind perturbations (Miranda 98). The model also includes a cost value for not using all wind available.

Besides costs, the fitness function also includes penalty values for not having all constraints satisfied. The highest penalty is assigned to schedules that do not meet the load. Other penalties apply to schedules that do not respect minimum down times or start up delays. A third set of penalties are included for schedules that do not satisfy a spinning reserve criterion, taken as a fuzzy constraint.

The application allows the user to select two spinning reserve criteria — either a percentage of the load or a probabilistic risk value.

The best fitness is therefore assigned to solutions with lowest value of cost+penalties.

7.1.8. Results

The CARE model has been extensively tested and is in demonstration in the island of Crete in Greece - a system composed of 17 different generators including 6 steam turbines, 4 diesel, 7 gas turbines and 3 wind parks.

The results in Figure 2 are for a problem encoded in 16 blocks, with 2 bits for block flexibility, with population size as 200 and the maximum number of generations set to 500.

Load and wind prediction curves for 48 hours were used, according to the parameters defined in CARE. The results presented in Figure 1 verify a spinning reserve criterion of at least 10% of load and a maximum fuzzy wind penetration of 20-25%. The penalties for violating constraints such as spinning reserve were decreased for the latest time steps of the scheduling period, given the increased uncertainty in load and wind predictions.

A series of tests to check on the effect and relevance of the Neighborhood Digging (ND) procedure were performed:

- SGA – Running the algorithm without ND
- NDlast – ND routine only in the last generation
- ND150 – ND routine every 40 generations, only until generation 160
- ND350 – ND routine only in the last 150 generations, every 40 generations
- GAND – ND routine every 40 generations.

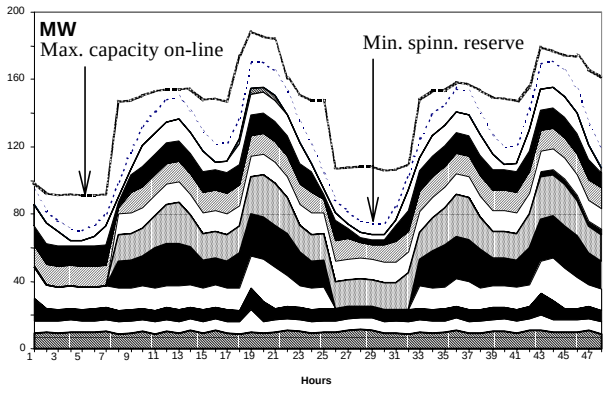


Figure 2 – Solution for a 48-hour UC. The base strip corresponds to wind generation. The spinning reserve criterion is satisfied (as well as all other constraints).

For each of these trials, 30 runs were repeated with random initialization of chromosomes. The best solution found was not the same in all runs, and the operation+scheduling cost varied in different degree, depending on the trial set.

The following table illustrates the aggregate results obtained:

Table 2 - Results of algorithm performance tests

	Best cost	St. dev. from the mean	St. dev. in %	Time (s)
SGA	59320	1419	2.39	74.8
Ndlast	58864	1126	1.91	
ND150	59139	1147	1.94	
ND350	58757	1083	1.84	
GAND	58255	795	1.36	181.8

It is clear that the GAND configuration of the algorithm gives the best and more robust results: not only it consistently detects better solutions than all other approaches, but also the variance of solutions offered in a number of runs is much narrower. The price to pay for this accuracy is an increase in computing time, compared with the SGA alternative with no ND. However, the computed time for GAND (3 minutes, in a PC Pentium 350 MHz with Windows NT) is still totally acceptable.

Figure 3 shows two samples for the evolution of the fitness function for a GAND and a SGA run. It clearly shows that it took GAND more or less half the number of generations (250) to reach solutions of the same quality of the ones obtained by SGA in 500 generations. It is not feasible to admit that one will do 30 runs in a real operation environment – but even so, the worst results of GAND will be better than the best of SGA.

It can also be seen that the ND procedure is more beneficial at a later stage of the genetic algorithm than in the first iterations.

This example illustrates that building an efficient GA

for a practical application requires careful modeling and understanding of how the evolutionary process acts.

This model has been integrated in the EMS set of applications in service in the island of Crete, Greece, and serves as demonstration of the potential of a set of advanced techniques.

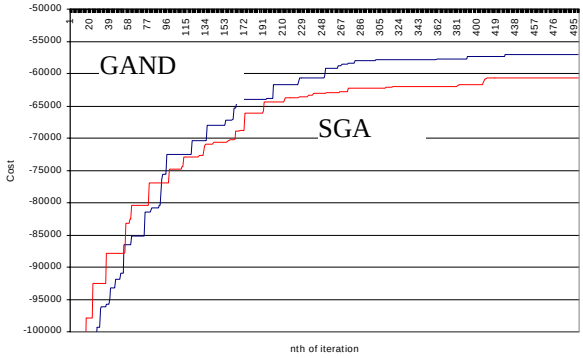


Figure 3 – Evolution of fitness function – comparison of GAND with SGA

7.1.9. Some references on the Unit Commitment problem

"Optimization of the unit commitment problem by a coupled gradient network and a genetic algorithm", Technical Report TR-103697, EPRI, 1994.

A. J. Adderman (1985) "Treatment of Integral Contingent Conditions in Optimal Power Plant Commitment", in H. J. Wacker (ed.) : Applied Optimization Techniques in Energy Problems, Stuttgart.

D. Dasgupta, D. R. McGregor, "Thermal unit commitment using genetic algorithms", IEE Proceedings Part C - Generation, Transmission and Distribution, Vol. 141, 1994, pp. 459-465.

G. B. Sheble, T. T. Maifeld, "Unit commitment by genetic algorithm and expert system", Electric Power Systems Research, Vol. 30, 1994, pp. 115-121.

H. Saitoh, K. Inoue, J. Toyoda, "Genetic algorithm approach to unit commitment problem", Proceedings of International Conference on Intelligent Systems Applications to Power Systems (ISAP'94), France, September 1994, pp. 583-589.

H. Sonnenschein (1982) "A modular Optimization calculation Method of Power station Energy balance and Plant Efficiency", Journal of engineering for Power, vol. 104 255-259

H. T. Yang, P. C. Yang, C. L. Huang, "A parallel genetic algorithm approach to solving the unit commitment problem: Implementation on the transputer networks", p. n.: 96 WM 292-3-PWRS, IEEE-PES Wt P. M., Baltimore, USA, Jan 21-25, 96.

H. T. Yang, P. C. Yang, C. L. Huang, "Applications of the genetic algorithm to the unit commitment problem in power generation industry", Proceedings of the Fourth IEEE International Conference on Fuzzy Systems, Part 1, Yokohama, Japan, 1995, pp. 267-274.

H. Wagner , "Procedures for the Solution of the Unit commitment Problem", in H. J. Wacker (ed.) : Applied Optimization Techniques in Energy Problems, Stuttgard, 1985.

J. H. Park, S. O. Yang, H. S. Lee, Y. M. Park, "Economic Load Dispatch Using Evolutionary Algorithms", Proc. of ISAP'96, Orlando, USA, 29 January-2 February, 1996, pp. 441-447.

L. Wei, Q. Zeng, T. Jiang, J. Yu, D. Huang, "New heuristic genetic algorithm and its application to electric power system unit combination", Proceedings of the Chinese Society of Electrical Engineering, Vol. 14, 1994, pp. 67-72, (in Chinese).

Mahfoud, S., "Niching methods for Genetic Algorithms", IlliGAL Report n. 95001, Illinois Genetic Algorithm Laboratory, May 1995 (as collected from the web page)

Miranda, V. et al, "Unit Commitment module — the CARE project", Internal Report of JOR3-CT96-0119 Project CARE, INESC Porto, Portugal, 1998

S. J. Huang, C. L. Huang, "Application of genetic based neural networks to thermal unit commitment", paper no. 96 WM 203-0 PWRS, presented at IEEE-PES Winter Power Meeting, Baltimore, USA, Jan 21-25, 1996.

S. Kazzrlis, A. Bakirtzis, V. Petridis, "A genetic algorithm solution to the unit commitment problem", IEEE PES Winter 94-95 Meeting, 1994, New York.

T. Ohta, T. Matsui, T. Makata, M. Kato, M. Aoyagi, M. Kunugi, K. Shimada, J. Nagata, "Practical Approach to Unit Commitment Problem using genetic Algorithm and Lagrangian Relaxation Method", Proc.of ISAP'96, Orlando, USA, 29 January-2 February, 1996, pp. 434-440.

T. T. Maifeld, G. B. Sheble, "Genetic-based unit commitment algorithm", paper no.: 95 SM 548-8-PWRS, presented at IEEE-PES Summer Meeting at Portland, USA, July 23-27, 1995.

X. Ma, A. A. El-Keib, R. E. Smith, H. Ma, "Genetic algorithm based approach to thermal unit commitment of electric power systems", Electric Power Systems Research, Vol. 34, No. 1, July 1995, pp.

7.2. Load Shedding

7.2.1. A model with GA

The following example is extracted from recent experiments (Ferreira 99).

Under-frequency load shedding is used to alleviate load-generation imbalance following a sudden loss of generation or loss of interconnection. A large frequency reduction happens when generators are not able to supply the connected system load or when generator governors are not fast enough to respond to such disturbances.

In these latter situations, although the system may remain stable, very large frequency deviations may cause the actuation of frequency protection relays (such as the ones installed in thermal power station auxiliaries) leading the system to collapse.

To avoid this situation, utilities usually have shedding schemes that disconnect selected loads throughout the entire power system; besides protecting against frequency collapse, this technique is also used to prevent deep drops in system frequency. However, sometimes the amount of disconnected load is far more than the necessary, which may also lead to large over frequency oscillations.

Load shedding is accomplished by using frequency sensitive relays that detect the onset of decay in the system frequency. Both frequency and rate-of-change of frequency are measured. Load shedding is usually implemented in stages with each stage triggered at a different frequency level or df/dt setting. This staging is performed to prioritize the shedding of the loads, where the least important loads are to be shed first.

Usually, this priority policy is associated to the social impact of the disconnection action, without any other concern related to the load behavior during the emergency conditions. Specifically, frequency dependent loads present better dynamic characteristics during the power imbalance phenomenon and may be kept connected

The following paragraphs describe a new approach based on GA that aims at determining, for a given loss of generation, the minimum amount of load to be disconnected and the step-actuation factor used for that purpose (minimum frequency relay setting level and/or *df/dt* operation). This calculation takes also into consideration the dynamic characteristics of loads (namely, frequency dependence) and its location on the network.

The minimization of the load curtailment is subjected to a set of constraints, such as:

1. prioritary loads should be kept in operation;
2. minimum system frequency deviation;
3. maximum residual system frequency value;
4. maximum system frequency excursion.

An optimal load shedding scheme of this type can then be formally considered as a large, non linear, discrete optimization problem subject to multiple constrains.

To implement the load shedding strategies, some assumptions must be introduced for the modeling of load buses. In the example and results below, without loss of generality, it has been assumed that there are four feeders in each load bus. These feeders are sharing the bus load such that, in each feeder, the combination of load types is different.

Load shedding is triggered in the example by two actuation types: *frequency level*, and *frequency rate-of-change*. In this work, 49, 48.5 and 48 Hz are respectively the first, second and third levels of frequency setting for load shedding. Simultaneously, the setting of frequency change rate is fixed at 0.5Hz/sec.

This means that whenever system frequency reaches one of three levels of frequency setting or if system frequency declines more than 0.5Hz per second, a scheduled amount of load will be disconnected. Time delays in the actuation of the frequency relays have also been considered.

Coding - The choice of how to encode solutions in a chromosome is of primary importance to the success of the genetic algorithm approach. In this work, each bit corresponds to one of the feeders of the system (possible location for a disconnecting device operated by a relay). The chromosome is divided into three parts, each part representing a shedding stage. Obviously, those feeders that are assigned to one stage would not be available in others.

Thus, each load bus must be coded using 12 bits of the chromosome. This coding example also shows that some bits may become irrelevant – the meaning of the chromosome depends entirely on the fitness function, i.e., on the interpretation of the user.

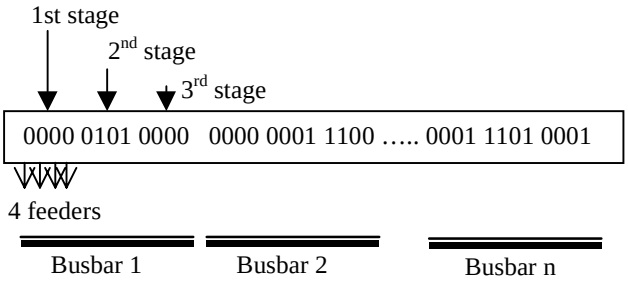


Figure 4 – One individual in the load shedding problem

Fitness - A fitness function (**fit**) defined as follows is used to assess the quality of each shedding solution coded in an individual chromosome:

$$fit = 100(1-\frac{|f_{mn}|+m}{A})(1-\frac{|f_{mx}|}{B})(1-\frac{|f_{fin}|}{C})(\frac{D.P_{los}-P_{shed}}{E.P_{los}})$$

where A, B, C, D and E are weight parameters dependent on the system under analysis and on the importance attributed to each frequency deviation, and m=-1(offset value). For each *individual* (solution), P_{los} is the amount of generation lost and P_{shed} is the amount of load to be shed.

The values f_{min}, f_{max} and f_{final} are respectively the minimum, maximum and final frequency deviation values obtained from a step-by-step numerical solution of the differential equation that characterizes the system dynamic behavior.

To speed up the procedure several conditions were included during the fitness assessment: solutions are discarded as soon as found unfeasible and other filtering procedures are included to avoid the need of evaluating the full dynamic behavior of the system.

Initial population - If ***1s*** and ***0s*** in each chromosome are chosen with equal probability, approximately half of feeders will be set ready for shedding. In other words, half of loads would be ready to be shed regardless of the amount of generation lost. This type of approach for choosing the initial population makes the optimization process faster and more efficient.

Genetic Operators - The work described in (Ferreira 99) uses the canonical Genetic Algorithm using a fixed-size, non-overlapping population scheme with each new generation created by the selection operator and altered by single point crossover and mutation, according to fixed operator probabilities.

Results - Here are some results from a test system with 16 buses and 9 generators that was used to evaluate the feasibility of this approach.

Figure 5 shows the evolution of the best solution for the total amount of load shedded with the successive generations of the genetic algorithm, for 100 generations.

Figure 6 is a graph of the deviation in frequency experienced by the test system for the best GA solution (upper line) and for another solution designed by hand. It shows how the solution found by the GA is of better quality: the GA solution requires less load shedded – however, its action is much more effective than a solution demanding a higher level of load disconnected.

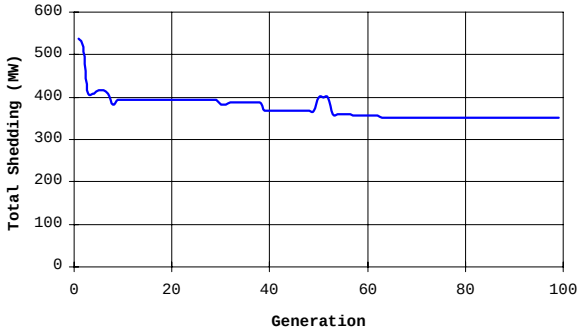


Figure 5 - Evolution of load to be shed suggested by the GA along generations

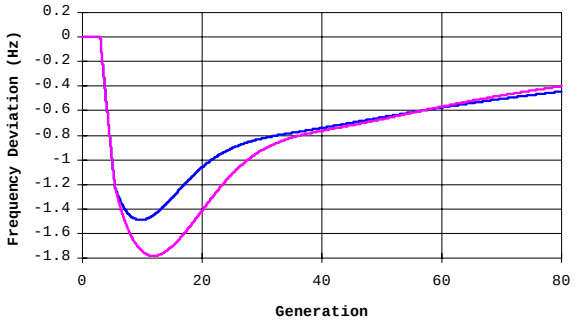


Figure 6 - System dynamic behavior for the GA solution (upper curve) and a larger shedding level (lower curve)

7.2.2. References for the load shedding problem

Peças Lopes, J.A., Wa, W.C., Proença, L.M., "Genetic Algorithms In The Definition Of Optimum Load Shedding Strategies", to appear in PSCC'99 – Power System Computation Conference, Trondheim, Norway, 1999

8. EVOLUTIONARY PROGRAMMING (EP) AND EVOLUTION STRATEGIES (ES)

8.1. Concepts

Evolutionary Programming, originally conceived by Lawrence J. Fogel in 1960, is a stochastic optimization strategy similar to Genetic Algorithms, but instead places emphasis on the behavioral linkage between parents and their offspring, rather than seeking to emulate specific Genetic Operators as observed in nature. Evolutionary Programming is similar to Evolution Strategies, although the two approaches developed independently.

Like both ES and GAs, EP is a useful method of optimization when other techniques such as gradient descent or direct, analytical discovery are not possible. Combinatorial and real-valued Function optimization in which the optimization surface or Fitness landscape is "rugged", possessing many locally optimal solutions, are well suited for Evolutionary Programming.

The 1966 book, "Artificial Intelligence Through Simulated Evolution" by Fogel, Owens and Walsh is the landmark publication for EP applications, although many other papers appear earlier in the literature. In the book, finite state automata were evolved to predict symbol strings generated from Markov processes and non- stationary time series. Such evolutionary prediction was motivated by a recognition that prediction is a keystone to intelligent behavior (defined in terms of adaptive behavior, in that the intelligent organism must anticipate events in order to adapt behavior in light of a goal).

In 1992, the First Annual Conference on Evolutionary Programming was held in La Jolla, CA. Further conferences have been held annually. The conferences attract a diverse group of academic, commercial and military researchers engaged in both developing the theory of the EP technique and in applying EP to a wide range of optimization problems, both in engineering and biology.

Rather than list and analyze the sources in detail, several fundamental sources are listed below which should serve as good pointers to the entire body of work in the field.

The Process For EP, like GAs, there is an underlying assumption that a Fitness landscape can be characterized in terms of variables, and that there is an optimum solution (or multiple such optima) in terms of those variables. For example, if one were trying to find the shortest path in a Traveling Salesman Problem, each solution would be a path. The length of the path could be expressed as a number, which would serve as the solution's fitness. The fitness landscape for this problem could be characterized as a hypersurface proportional to the path lengths in a space of possible paths. The goal would be to find the globally shortest path in that space, or more practically, to find very short tours very quickly.

The basic EP method involves 3 steps (Repeat until a threshold for iteration is exceeded or an adequate solution is obtained):

(1) Choose an initial Population of trial solutions at random. The number of solutions in a population is highly relevant to the speed of optimization, but no definite answers are available as to how many solutions are appropriate (other than >1) and how many solutions are just wasteful.

(2) Each solution is replicated into a new Population. Each of these offspring solutions are mutated according to a distribution of Mutation types, ranging from minor to extreme with a continuum of Mutation types between. The severity of Mutation is judged on the basis of the functional change imposed on the parents.

(3) Each offspring solution is assessed by computing it's Fitness. Typically, a stochastic tournament is held to determine N solutions to be retained for the Population of solutions, although this is occasionally performed deterministically. There is no requirement that the Population Size be held constant, however, nor that only a single offspring be generated from each Parent.

It should be pointed out that EP typically does not use any Crossover as a Genetic Operator.

8.2. EP and GAs

There are two important ways in which EP differs from GAs.

First, there is no constraint on the representation. The typical GA approach involves encoding the problem solutions as a string of representative tokens, the GENOME. In EP, the representation follows from the problem. A neural network can be represented in the same manner as it is implemented, for example, because the Mutation operation does not demand a linear encoding. (In this case, for a fixed topology, real-valued weights could be coded directly as their real values and Mutation operates by perturbing a weight vector with a zero mean multivariate Gaussian perturbation. For variable topologies, the architecture is

also perturbed, often using Poisson distributed additions and deletions.)

Second, the Mutation operation simply changes aspects of the solution according to a statistical distribution which weights minor variations in the behavior of the offspring as highly probable and substantial variations as increasingly unlikely. Further, the severity of Mutations is often reduced as the global optimum is approached. There is a certain tautology here: if the global optimum is not already known, how can the spread of the Mutation operation be damped as the solutions approach it? Several techniques have been proposed and implemented which address this difficulty, the most widely studied being the "Meta-Evolutionary" technique in which the variance of the Mutation distribution is subject to Mutation by a fixed variance Mutation operator and evolves along with the solution.

8.3. EP and ES

The first communication between the Evolutionary Programming and Evolution Strategy groups occurred in early 1992, just prior to the first annual EP conference. Despite their independent development over 30 years, they share many similarities. When implemented to solve real-valued Function Optimization problems, both typically operate on the real values themselves (rather than any coding of the real values as is often done in GAs). Multivariate zero mean Gaussian Mutations are applied to each Parent in a Population and a Selection mechanism is applied to determine which solutions to remove (i.e., "cull") from the population. The similarities extend to the use of self-adaptive methods for determining the appropriate Mutations to use -- methods in which each Parent carries not only a potential solution to the problem at hand, but also information on how it will distribute new trials (offspring). Most of the theoretical results on Convergence (both asymptotic and velocity) developed for ES or EP also apply directly to the other.

The main differences between ES and EP are:

Selection: EP typically uses Stochastic Selection via a tournament. Each trial Solution in the Population faces competition against a pre-selected number of opponents and receives a "win" if it is at least as good as its opponent in each encounter. Selection then eliminates those Solutions with the least wins. In contrast, ES typically uses deterministic Selection in which the worst Solutions are purged from the Population based directly on their function evaluation.

Recombination: EP is an abstraction of Evolution at the level of reproductive Populations (i.e., Species) and thus no Recombination mechanisms are typically used because Recombination does not occur between Species (by definition: see Mayr's biological species concept). In contrast, ES is an abstraction of Evolution at the level of Individual behavior. When self-adaptive information is incorporated this is purely Genetic information (as opposed to Phenotypic) and thus some forms of Recombination are reasonable and many forms of Recombination have been implemented within ES. Again, the effectiveness of such operators depends on the problem at hand.

```
Algorithm EP is
                                        // start with an initial time
t := 0;
  // initialise a usually random population of individuals
initpopulation P (t);
 // evaluate fitness of all initial individuals of population
evaluate P (t);
        // test for termination criterion (time, fitness, etc.)
while not done do
            // perturb the whole population stochastically
   P'(t) := mutate P (t);
                                    // evaluate it's new fitness
   evaluate P' (t);
  // stochastically select the survivors from actual fitness
   P(t+1) := survive P(t),P'(t);
                                  // increase the time counter
   t := t + 1;
                                                        // done
end EP.
```

9. EXAMPLES OF APPLICATION OF EP IN POWER SYSTEMS

The number of examples of application of EP in Power System problems is much smaller that of GA. However, EP models are receiving increasing attention from the community, because they seem to be more efficient for certain types of problems than GA.

9.1. Load dispatch

One of the tasks of the CARE project mentioned above was to compare the efficiency of a set of dispatch techniques, using either conventional mathematical programming models either novel approaches. We will describe a simple didactic test done for the application of Evolutionary Programming to the Load Dispatch

problem, discussed in (Proença 99).

Evolutionary Programming algorithms in Economic Dispatch (ED) provide an edge over common GA mainly because:

- They don't require any special coding of individuals. In the case of ED, since the desired outcome is the operating point of each of the dispatched units (a real number), each of the individuals can be directly presented as a set of real numbers, each one being the produced power of the unit it concerns.
- Since each of the individuals codes within itself its own mutation rate, and since it is itself mutated, the algorithms provide a self-regulating adaptive scheme.

On the other hand, no special requirements are made regarding the objective function and constraints, which is a very interesting feature of Evolutionary Programming algorithms (and Genetic Algorithms) as compared to traditional methods.

The algorithm was developed in an Object Oriented fashion, in the C++ programming language. This option was made given the high flexibility and ease of reconfiguration given by this approach.

9.1.1. Fitness

The fitness function that was used for evaluation of the individuals (and the one the optimum or near-optimum solution was to be found), was based on the following cost function:

$$C_T = \sum_{i=1}^{NGen} C_i(P_i) + L * W_L + \left[(D - \sum_{i=1}^{NGen} P_i) \middle| \text{if} > 0 \right]^2 * W_V$$

C_T	=	Total Cost
NGen	=	Number of dispatched units
C_i	=	Operation cost function of unit i
P_i	=	Active power output of generator i
L	=	Total losses
W_L	=	Per – unit cost of losses
D	=	Total demand
W_V	=	Underdispatch penalty

Constraints were implemented as to assure that $P_{i_{min}} < P_i < P_{i_{max}}$ - generator operating limits.

9.1.2. Mutation and Replacement Schemes

Each of the elements of the population consists in a list of records, each record assigned to one unit. Each of those records has two fields:

- Field 1 with the current dispatched power of the station.
- Field 2 with the current mutation standard deviation.

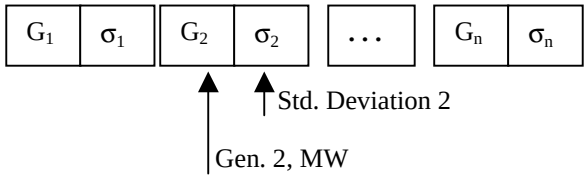


Figure 7 – Composition of one element of the population

In each of the generations, and for each of the elements of the population, the value in field 1 for each generator is changed (mutated), adding a random amount with normal distribution, zero average and standard deviation given in field 2. The individual is then evaluated. If its new fitness is better, the "mutated" individual replaces the former one. If fitness is worse, the "mutated" individual is accepted to replace the former one with a given probability. Following this procedure, field 2 is also mutated, being added a random normal distributed number with 0 average and with a constant value of standard deviation that is an input to the algorithm.

After mutation is applied to all the individuals, the new population is sorted and the worse individual is replaced with a copy of the best one.

9.1.3. Algorithm Flowchart

The evaluation process, were the fitness of each of the individuals is calculated, requires running a load flow routine to evaluate the losses associated with each individual dispatch and to calculate the dispatch of the compensation bar generator. The chosen load flow method was the Newton-Raphson method. The flow chart depicting the steps of the evaluation of each individual is presented in Figure 8.

9.1.4. Algorithm Input Data

The input information of this module includes both algorithm parameters and network data. Since the evaluation function needs a load flow to be run, this module needs all of the electrical parameter information required for the load flow concerning its elements.

In what concerns the algorithm itself, the user can set the following parameters:

- Size of the population.
- Number of generations.
- Overload penalty (the parameter W_V
- Losses penalty (the parameter W_L)

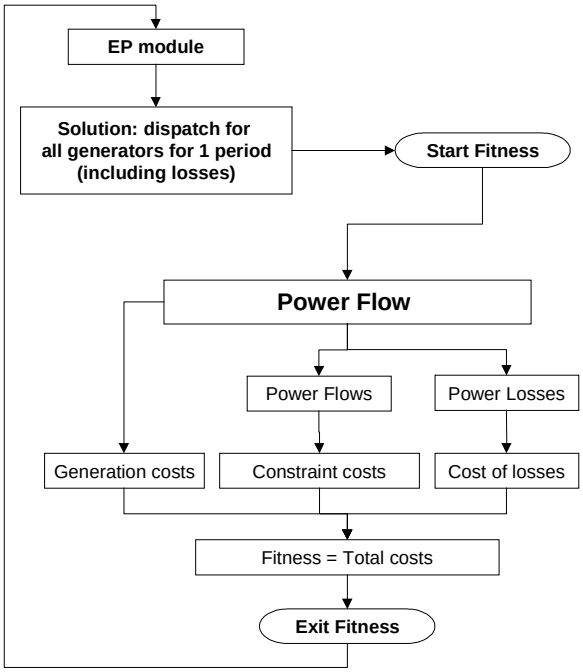


Figure 8 - Evaluation Sequence Flowchart

-

9.1.5. Algorithm Testing

The algorithm was tested on the network in Figure 9, consisting in 25 buses, 5 synchronous generators (on bars 0, 1, 2, 3, 4, 5, 7), 8 asynchronous generators (on bars 17, 18, 19, 20, 21, 22, 23, 34), each with capacitor bank (not shown) and 6 transmission lines. Loads are also assigned to bars 6, 8, 13, 14, and 15.

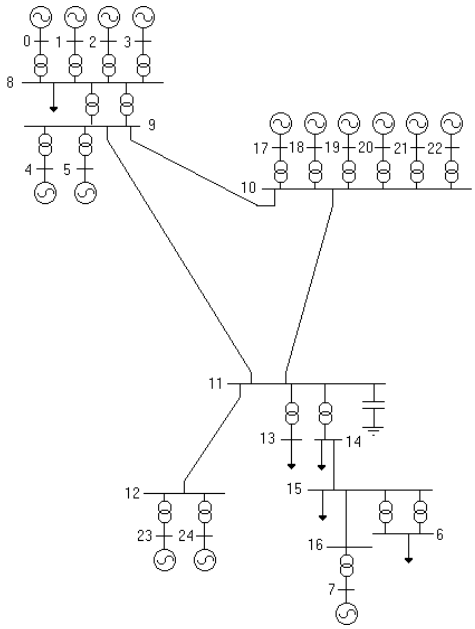


Figure 9 - Test Network

All asynchronous generators were considered to have cost functions of the form:

$$GC = 0.5 * P_i^2 + 100 * P_i + 25$$

In the case of synchronous generators, the cost functions were considered to be:

$$GC = P_i^2 + 85 * P_i + 400$$

The population sizes tested were of 5, 10 and 20 individuals.

Figure 10 shows the evolution of the fitness of the best EP solution in each generation for different population sizes (5, 10 and 20 individuals). The run time for 1000 generations and a population of 10 was 71 s, all tests made with an Intel Pentium II 333MHz computer with 64Mb memory.

We may notice that apparently the size of the population had a small influence on the final solution. But this point must be analyzed carefully, because different tuns with the same population size may yeld slightly different results, due to the probabilistic nature of the process.

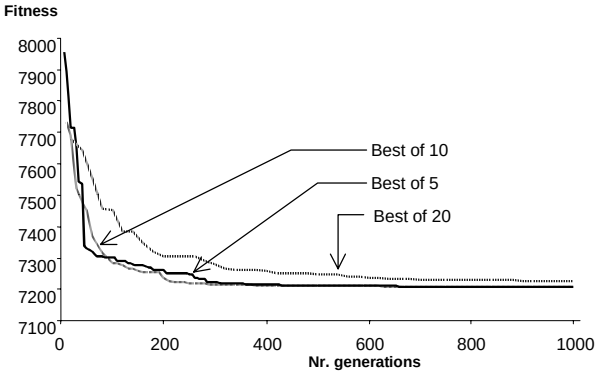


Figure 10 – Evolution of EP for different population sizes

9.2. References for the Load Dispatch problem and other EP/ES references

L. M. Proença, J. L. Pinto, M. Matos, "INESC Developments on the Economic Dispatch Module", Internal Report of JOR3-CT96-0119 Project CARE, INESC Porto, Portugal, Feb 1999

D. E. Bouchard, M. M. A. Salama, A. Y. Chikhani, "Optimal Distribution Feeder Routing And Optimal Substation Sizing And Placement Using Evolutionary Strategies", Proceedings of the 1994 Canadian Conference on Electrical and Computer Engineering, Vol. 2, Halifax, Canada, 1994, pp. 661-664.

J. H. Park, S. O. Yang, H. S. Lee, Y. M. Park, "Economic Load Dispatch Using Evolutionary Algorithms", Proc. of ISAP'96, Orlando, USA, 29 January-2 February, 1996, pp. 441-447

J. T. Ma, L. L. Lai, "Optimal reactive power dispatch using evolutionary programming", Proceedings of IEEE/KTH Stockholm Power Tech Conference, Stockholm, Sweden, June 18-22, 1995, pp. 662-667.

10. OTHER ASPECTS OF EVOLUTIONARY COMPUTATION

10.1. Evolution Strategies (ES).

In 1963 two students at the Technical University of Berlin (TUB) met and were soon to collaborate on experiments which used the wind tunnel of the Institute of Flow Engineering. During the search for the optimal shapes of bodies in a flow, which was then a matter of laborious intuitive experimentation, the idea was conceived of proceeding strategically. However, attempts with the coordinate and simple gradient strategies were unsuccessful. Then one of the students, Ingo Rechenberg, now Professor of Bionics and Evolutionary Engineering, hit upon the idea of trying random changes in the parameters defining the shape, following the example of natural Mutations. The Evolution Strategy was born. A third student, Peter Bienert, joined them and started the construction of an automatic experimenter, which would work according to the simple rules of Mutation and Selection. The second student, Hans-Paul Schwefel, set about testing the efficiency of the new methods with the help of a Zuse Z23 computer; for there were plenty of objections to these "random strategies."

In spite of an occasional lack of financial support, the Evolutionary Engineering Group which had been formed held firmly together. Ingo Rechenberg received his doctorate in 1970 (Rechenberg 73). It contains the theory of the two membered Evolution Strategy and a first proposal for a multimembered strategy which in the nomenclature introduced here is of the (m+1) type. In the same year financial support from the Deutsche Forschungsgemeinschaft (DFG, Germany's National Science Foundation) enabled the work, that was concluded, at least temporarily, in 1974 with the thesis "Evolutionsstrategie und numerische Optimierung" (Schwefel 77).

Thus, Evolution Strategies were invented to solve technical optimization problems (TOPs) like e.g. constructing an optimal flashing nozzle, and until recently ES were only known to civil engineering folks, as an alternative to standard solutions. Usually no closed form analytical objective function is available for TOPs and hence, no applicable optimization method exists, but the engineer's intuition.

10.2. Classifier Systems (CFS)

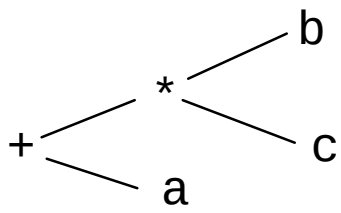
A Classifier System is system which takes a set of inputs, and produces a set of outputs which indicate some classification of the inputs. An example might take inputs from sensors in a chemical plant, and classify them in terms of: 'running ok', 'needs more water', 'needs less water', 'emergency'. These systems are rule-based machine-learning systems capable of learning by examples. See Niessen 1993 for a small introduction to CFS. For a more detailed report see:

John Holland et al. *Induction: Processes of Inference, Learning, and Discovery*, Cambridge Mass: MIT Press.

D. E. Goldberg, *Genetic algorithms in Search, Optimization and Machine Learning*, 1989 Addison-Wesley

10.3. Genetic Programming (GP)

Genetic Programming is the extension of the genetic model of learning into the space of programs. That is, the objects that constitute the Population are not fixed-length character strings that encode possible solutions to the problem at hand, they are programs that, when executed, "are" the candidate solutions to the problem. These programs are expressed in genetic programming as parse trees, rather than as lines of code. Thus, for example, the simple program "a + b * c" would be represented as:



or, to be precise, as suitable data structures linked together to achieve this effect. Because this is a very simple thing to do in the programming language Lisp, many GPers tend to use Lisp. However, this is simply an implementation detail. There are straightforward methods to implement GP using a non-Lisp programming environment.

The programs in the Population are composed of elements from the Function Set and the Terminal Set, which are typically fixed sets of symbols selected to be appropriate to the solution of problems in the domain of interest.

In GP the Crossover operation is implemented by taking randomly selected subtrees in the Individuals (selected according to Fitness) and exchanging them.

It should be pointed out that GP usually does not use any Mutation as a Genetic Operator.

11. WHO IS INTERESTED IN EC?

11.1. . Who indeed?

Evolutionary Computation attracts researchers and people of quite dissimilar disciplines, i.e. EC is a interdisciplinary research field.

Computer scientists

Want to find out about the properties of sub-symbolic information processing with EAs and about learning, i.e. adaptive systems in general.

They also build the hardware necessary to enable future EAs (precursors are already beginning to emerge) to huge real world problems, i.e. the term "massively parallel computation", springs to mind.

Engineers

Of many kinds want to exploit the capabilities of EAs on many areas to solve their application, esp. optimization problems.

Roboticists

Want to build MOBOTs (MOBile ROBOTs, i.e. R2D2's and #5's cousins) that navigate through uncertain Environments, without using built-in "maps". The MOBOTS thus have to adapt to their surroundings, and learn what they can do "move-through-door" and what they can't "move- through-wall" on their own by "trial-and-error".

Cognitive scientists

Might view CFS as a possible apparatus to describe models of thinking and cognitive systems.

Physicists

Use EC hardware, e.g. Hillis' (Thinking Machine Corp.'s) Connection Machine to model real world problems which include thousands of variables, that run "naturally" in parallel, and thus can be modeled more easily and esp. "faster" on a parallel machine, than on a serial "PC" one.

Biologists

In fact many working biologists are hostile to modeling, but an entire community of Population Biologists arose with the 'evolutionary synthesis' of the 1930s created by the polymaths R.A. Fisher, J.B.S. Haldane, and S. Wright. Wright's Selection in small Populations, thereby avoiding local optima) is of current interest to both biologists and ECers -- populations are naturally parallel.

A good exposition of current Population Biology modeling is J. Maenad Smith's text Evolutionary Genetics. Richard Dawn’s Selfish Gene and Extended Phenotype are unparalleled (sic!) prose expositions of evolutionary processes. Rob Collins' papers are excellent parallel GA models of evolutionary processes (available in [ICGA91] and by FTP from ftp.cognet.ucla.edu:/pub/alife/papers/).

As fundamental motivation, consider Fisher's comment: "No practical biologist interested in (e.g.) sexual Reproduction would be led to work out the detailed

consequences experienced by organisms having three or more sexes; yet what else should [s/]he do if [s/]he wishes to understand why the sexes are, in fact, always two?" (Three sexes would make for even weirder grammar, [s/]he said...)

Economists

EC has been used extensively in Management Sciences. See Nissen93 for more details.

Philosophers...

...and some other really curious people may also be interested in EC for various reasons.

11.2. . References

Artificial Intelligence Through Simulated Evolution [Fogel66] Birkhaeuser.

Booker, L.B. (1982) "Intelligent behavior as an adaptation to the task environment" PhD Dissertation, Univ. of Michigan, Logic of Computers Group, Ann Arbor, MI.

Braitenberg, V. (1984) "Vehicles: Experiments in Synthetic Psychology" Boston, MA: MIT Press.

Dorigo M. & H. Bersini (1994). "A Comparison of Q-Learning and Classifier Systems." Proceedings of From Animals to Animats, Third International Conference on Simulation of Adaptive Behavior (SAB94), Brighton, UK, D.Cliff, P.Husbands, J.-A.Meyer and S.W.Wilson (Eds.), MIT Press, 248-255.

Fogel DB (1995) "Evolutionary Computation: Toward a New Philosophy of Machine Intelligence," IEEE Press, Piscataway, NJ.

Holland, J.H. & Reitman, J.S. (1978) "Cognitive Systems based on Adaptive Algorithms" In D.A. Waterman & F.Hayes-Roth, (eds) Pattern- directed inference systems. NY: Academic Press.

Holland, J.H. (1986) "Escaping Brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems". In: R.S. Michalski, J.G. Carbonell & T.M. Mitchell (eds), Machine Learning: An Artificial Intelligence approach, Vol II, 593-623, Los Altos, CA: Morgan Kaufman.

Holland, J.H. (1992) "Adaptation in natural and artificial systems" Boston, MA: MIT Press.

Holland, J.H. (1995) "Hidden Order: How adaptation builds complexity" Reading, MA: Addison-Wesley.

Holland, J.H., et al. (1986) "Induction: Processes of Inference, Learning, and Discovery", Cambridge, MA: MIT Press.

Kursawe, F. (1992) " Evolution strategies for vector optimization"

Minsky, M.L. (1961) "Steps toward Artificial Intelligence" Proceedings IRE, 49, 8-30. Reprinted in E.A. Feigenbaum & J. Feldman (eds) Computers and Thought, 406-450, NY: McGraw-Hill, 1963.

Minsky, M.L. (1967) "Computation: Finite and Infinite Machines" Englewood Cliffs, NJ: Prentice-Hall.

Post, Emil L. (1943) "Formal reductions of the general combinatorial decision problem" American Journal of Mathematics, 65, 197-215.

Proceeding of the first [EP92], second [EP93] and third [EP94] Annual Conference on Evolutionary Programming (primary)

Rechenberg, I. (1973) "Evolutionsstrategie: Optimierung Research, Budapest, 1025-1033.

Rich, E. (1983) "Artificial Intelligence" NY: McGraw-Hill.

Schwefel, H.-P. (1977) "Numerische Optimierung von technischer Systeme nach Prinzipien der biologischen Evolution",

Schwefel, H.-P. (1987) "Collective Phaenomena in Evolutionary Systems", 31st Annu. Meet. Inter'l Soc. for General System

Tinbergen, N. (1951) "The Study of Instinct" NY: Oxford Univ. Press.

Watkins, C. (1989) "Learning from Delayed Rewards" PhD Dissertation, Department of Psychology, Cambridge Univ., UK.

Wilson, S.W. (1985) "Knowledge growth in an artificial animal" in [ICGA85], 16-23.

Wilson, S.W. (1994) "ZCS: a zeroth level classifier system" in EC 2(1), 1-18.

12. GLOSSARY

1/5 Success Rule:

Derived by I. Rechenberg, the suggestion that when Gaussian Mutations are applied to real-valued vectors in searching for the minimum of a function, a rule-of-thumb to attain good rates of error convergence is to adapt the Standard Deviation of Mutations to generate one superior solution out of every five attempts.

Adaptive Behavior:

Underlying mechanisms that allow animals, and potentially, Robots to adapt and survive in uncertain environments.

Allele :

(biol) Each Gene is able to occupy only a particular region of a Chromosome, it's locus. At any given locus there may exist, in the Population, alternative forms of the gene. These alternative are called alleles of one another.

Each gene may have an Allele of 0 or 1.

Artificial Intelligence:

"...the study of how to make computers do things at which, at the moment, people are better" --- Elaine Rich (1988)

Artificial Life:

Term coined by Christopher G. Langton for his 1987 [ALIFEI] conference. In the preface of the proceedings he defines ALIFE as "...the study of simple computer generated hypothetical life forms, i.e. life-as-it-could-be."

Building Block:

(EC) A small, tightly clustered group of Genes which have co-evolved in such a way that their introduction into any Chromosome will be likely to give increased Fitness to that chromosome.

The "building block hypothesis" [GOLD89] states that GAs find solutions by first finding as many Building Blocks as possible, and then combining them together to give the highest Fitness.

Central Dogma:

(biol) The dogma that nucleic acids act as templates for the synthesis of proteins, but never the reverse. More generally, the dogma that Genes exert an influence over the form of a body, but the form of a body is never translated back into genetic code: acquired characteristics are not inherited. cf Lamarckism.

(GA) The dogma that the behavior of the algorithm must be analyzed using the Schema Theorem.

Chromosome:

(biol) One of the chains of DNA found in cells. Chromosomes contain Genes, each encoded as a subsection of the DNA chain. Chromosomes are usually present in all cells in an organism, even though only a minority of them will be active in any one cell.

(EC) A data structure which holds a `string' of task parameters, or Genes. This may be stored, for example, as a binary bit- string, or an array of integers.

Classifier System:

A system which takes a (set of) inputs, and produces a (set of) outputs which indicate some classification of the inputs. An example might take inputs from sensors in a chemical plant, and classify them in terms of: 'running ok', 'needs more water', 'needs less water', 'emergency'.

Combinatorial Optimization:

Some tasks involve combining a set of entities in a specific way (e.g. the task of building a house). A general combinatorial task involves deciding (a) the specifications of those entities (e.g. what size, shape, material to make the bricks from), and (b) the way in which those entities are brought together (e.g. the number of bricks, and their relative positions). If the resulting combination of entities can in some way be given a Fitness score, then Combinatorial Optimization is the task of designing a set of entities, and deciding how they must be configured, so as to give maximum fitness. cf Order-Based Problem.

Comma Strategy:

Notation originally proposed in Evolution Strategies, when a Population of μ parents generates λ offspring and the μ parents are discarded, leaving only the λ Individuals to compete directly. Such a process is written as a (μ, λ) search. The process of only competing offspring then is a "comma strategy."

Converged:

A Gene is said to have Converged when 95% of the Chromosomes in the Population all contain the same Allele for that gene. In some circumstances, a population can be said to have converged when all genes have converged. (However, this is not true of populations containing multiple Species, for example.)

Most people use "convergence" fairly loosely, to mean "the GA has stopped finding new, better solutions". Of course, if you wait long enough, the GA will eventually find a better solution (unless you have already found the global optimum). What people really mean is "I'm not willing to wait for the GA to find a new, better solution, because I've already waited longer than I wanted to and it hasn't improved in ages."

An interesting discussion on convergence by Michael Vose can be found in GA-Digest v8n22, available from ftp.aic.nrl.navy.mil:/pub/galist/digests/v8n22

Convergence Velocity:

The rate of error reduction.

Cooperation:

The behavior of two or more individuals acting to increase the gains of all participating individuals.

Crossover:

(EC) A Reproduction Operator which forms a new Chromosome by combining parts of each of two 'parent' chromosomes. The simplest form is single-point Crossover, in which an arbitrary point in the chromosome is picked. All the information from Parent A is copied from the start up to the

Crossover point, then all the information from parent B is copied from the Crossover point to the end of the chromosome. The new chromosome thus gets the head of one parent's chromosome combined with the tail of the other. Variations exist which use more than one Crossover point, or combine information from parents in other ways.

(biol) A complicated process which typically takes place as follows: Chromosomes, while engaged in the production of Gametes, exchange portions of genetic material. The result is that an almost infinite variety of gametes may be produced. Subsequently, during sexual Reproduction, male and female gametes (i.e. sperm and ova) fuse to produce a new Diploid cell with a pair of chromosomes. In [HOLLAND92] the sentence "When sperm and ova fuse, matching Chromosomes line up with one another and then cross over partway along their length, thus swapping genetic material" is thus wrong, since these two activities occur in different parts of the life cycle.

Darwinism:

(biol) Theory of Evolution, proposed by Darwin, that evolution comes about through random variation of heritable characteristics, coupled with natural Selection (survival of the fittest). A physical mechanism for this, in terms of Genes and Chromosomes, was discovered many years later. Darwinism was combined with the Selectionism of Weismann and the genetics of Mendel to form the Neo-Darwinian Synthesis during the 1930s-1950s by T. Dobzhansky, E. Mayr, G. Simpson, R. Fisher, S. Wright, and others.

(EC) Theory which inspired all branches of EC.

Deception:

The condition where the combination of good Building Blocks leads to reduced Fitness, rather than increased fitness. Proposed by [GOLD89] as a reason for the failure of GAs on many tasks.

Diploid:

(biol) This refers to a cell which contains two copies of each Chromosome. The copies are homologous i.e. they contain the same Genes in the same sequence. In many sexually reproducing Species, the genes in one of the sets of chromosomes will have been inherited from the father's Gamete (sperm), while the genes in the other set of chromosomes are from the mother's gamete (ovum).

DNA:

(biol) Deoxyribonucleic Acid, a double stranded macromolecule of helical structure (comparable to a spiral staircase). Both single strands are linear, unbranched nucleic acid molecules build up from alternating deoxyribose (sugar) and phosphate molecules. Each deoxyribose part is coupled to a nucleotide base, which is responsible for establishing the connection to the other strand of the DNA. The 4 nucleotide bases Adenine (A), Thymine (T), Cytosine (C) and Guanine (G) are the alphabet of the genetic information. The sequences of these bases in the DNA molecule determines the building plan of any organism. [eds note: suggested reading: James D. Watson (1968) "The Double Helix", London: Weidenfeld and Nicholson]

Elitism:

Elitism (or an elitist strategy) is a mechanism which is employed in some EAs which ensures that the Chromosomes of the most highly fit member(s) of the Population are passed on to the next Generation without being altered by Genetic Operators. Using elitism ensures that the maximum Fitness of the population can never reduce from one generation to the next. Elitism usually brings about a more rapid convergence of the population. In some applications elitism improves the chances of locating an optimal Individual, while in others it reduces it.

Environment:

(biol) That which surrounds an organism. Can be 'physical' (abiotic), or biotic. In both, the organism occupies a Niche which influences its Fitness within the total Environment. A biotic environment may present frequency-dependent fitness functions within a, that is, the fitness of an organism's behavior may depend upon how many others are also doing it. Over several, biotic environments may foster co-evolution, in which fitness is determined withpartly by other.

Epistasis:

(biol) A "masking" or "switching" effect among Genes. A biology textbook says: "A gene is said to be epistatic when its presence suppresses the effect of a gene at another locus. Epistatic genes are sometimes called inhibiting genes because of their effect on other genes which are described as hypostatic."

(EC) When EC researchers use the term Epistasis, they are generally referring to any kind of strong interaction among Genes, not just masking effects. A possible definition is:

Epistasis is the interaction between different Genes in a Chromosome. It is the extent to which the contribution to Fitness of one gene depends on the values of other genes. Problems with little or no Epistasis are trivial to solve (hillclimbing is sufficient). But highly epistatic problems are difficult to solve, even for GAs. High epistasis means that Building Blocks cannot form, and there will be Deception.

Evolution:

That process of change which is assured given a reproductive Population in which there are (1) varieties of Individuals, with some varieties being (2) heritable.

Evolution Strategy:

A type of evolutionary algorithm developed in the early 1960s in Germany. It employs real-coded parameters, and in its original form, it relied on Mutation as the search operator, and a Population size of one. Since then it has evolved to share many features with Genetic Algorithms.

Evolutionarily Stable Strategy:

A strategy that does well in a Population dominated by the same strategy. (cf Maynard Smith, 1974) Or, in other words, "An 'ESS'... is a strategy such that, if all the members of a population adopt it, no mutant strategy can invade." (Maynard Smith "Evolution and the Theory of Games", 1982).

Evolutionary Computation:

Encompasses methods of simulating Evolution on a computer. The term is relatively new and represents an effort bring together researchers who have been working in closely related fields but following different paradigms.

The field is now seen as including research in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Artificial Life, and so forth. For a good overview see the editorial introduction to Vol. 1, No. 1 of "Evolutionary Computation" (MIT Press, 1993). That, along with the papers in the issue, should give you a good idea of representative research.

Evolutionary Programming:

An evolutionary algorithm developed in the mid 1960s. It is a stochastic Optimization strategy, which is similar to Genetic Algorithms, but dispenses with both "genomic" representations and with Crossover as a Reproduction Operator.

Evolutionary Systems:

A process or system which employs the evolutionary dynamics of Reproduction, Mutation, competition and Selection. The specific forms of these processes are irrelevant to a system being described as "evolutionary."

Exploitation:

When traversing a Search Space, Exploitation is the process of using information gathered from previously visited points in the search space to determine which places might be profitable to visit next.

An example is hill-climbing, which investigates adjacent points in the search space, and moves in the direction giving the greatest increase in Fitness. Exploitation techniques are good at finding local maxima.

Exploration:

The process of visiting entirely new regions of a Search Space, to see if anything promising may be found there. Unlike Exploitation, Exploration involves leaps into the unknown. Problems which have many local maxima can sometimes only be solved by this sort of random search.

Fitness:

(biol) Loosely: adaptedness. Often measured as, and sometimes equated to, relative reproductive success. Also proportional to expected time to extinction. "The fit are those who fit their existing ENVIRONMENTs and whose descendants will fit future environments." (J. Thoday, "A Century of Darwin", 1959). Accidents of history are relevant.

(EC) A value assigned to an Individual which reflects how well the individual solves the task in hand. A "fitness function" is used to map a Chromosome to a Fitness value. A "fitness landscape" is the hypersurface obtained by applying the fitness function to every point in the Search Space.

Function Optimization:

For a function which takes a set of N input parameters, and returns a single output value, F, Function Optimization is the task of finding the set(s) of parameters which produce the maximum (or minimum) value of F. Function Optimization is a type of Value-Based Problem.

Function Set:

(GP) The set of operators used in GP. These functions label the internal (non-leaf) points of the parse trees that represent the programs in the Population. An example Function Set might be

{+, -, *}.

Game Theory:

A mathematical theory originally developed for human games, and generalized to human economics and military strategy, and to Evolution in the theory of Evolutionarily Stable Strategy. Game Theory comes into it's own wherever the optimum policy is not fixed, but depends upon the policy which is statistically most likely to be adopted by opponents.

Gamete:

(biol) Cells which carry genetic information from their parents for the purposes of sexual Reproduction. In animals, male Gametes are called sperm, female gametes are called ova. Gametes have a Haploid number of Chromosomes.

Gene:

(EC) A subsection of a Chromosome which (usually) encodes the value of a single parameter.

(biol) The fundamental unit of inheritance, comprising a segment of DNA that codes for one or several related functions and occupies a fixed position (locus) on Chromosome. However, the term may be defined in different ways for different purposes. For a fuller story, consult a book on genetics.

Gene-Pool:

The whole set of Genes in a breeding Population. The metaphor on which the term is based de-emphasizes the undeniable fact that genes actually go about in discrete bodies, and emphasizes the idea of genes flowing about the world like a liquid.

Generation:

(EC) An iteration of the measurement of Fitness and the

creation of a new Population by means of Reproduction Operators.

Genetic Algorithm:

A type of Evolutionary Computation devised by John Holland [Holland92]. A model of machine learning that uses a genetic/evolutionary metaphor. Implementations typically use fixed-length character strings to represent their genetic information, together with a Population of Individuals which undergo Crossover and Mutation in order to find interesting regions of the Search Space.

Genetic Drift:

Changes in gene/allele frequencies in a Population over many Generations, resulting from chance rather than Selection. Occurs most rapidly in small populations. Can lead to some Alleles becoming `extinct', thus reducing the genetic variability in the population.

Genetic Programming:

Genetic Algorithms applied to programs. Genetic Programming is more expressive than fixed-length character string GAs, though GAs are likely to be more efficient for some classes of problems.

Genetic Operator:

A search operator acting on a coding structure that is analogous to a Genotype of an organism (e.g. a Chromosome).

Genotype:

The genetic composition of an organism: the information contained in the Genome.

Genome:

The entire collection of Genes (and hence Chromosomes) possessed by an organism.

Global Optimization:

The process by which a search is made for the extreme (or extrema) of a functional which, in Evolutionary Computation, corresponds to the Fitness or error function that is used to assess the Performance of any Individual.

Haploid:

(biol) This refers to cell which contains a single Chromosome or set of chromosomes, each consisting of a single sequence of Genes. An example is a Gamete. cf Diploid.

In EC, it is usual for Individuals to be Haploid.

Hard Selection:

Selection acts on competing Individuals. When only the best available individuals are retained for generating future progeny, this is termed "hard Selection." or “deterministic Selection”. In contrast, "soft Selection" offers a probabilistic mechanism for maintaining individuals to be parents of future progeny despite possessing relatively poorer objective values.

Individual:

A single member of a Population. In EC, each Individual contains a Chromosome (or, more generally, a Genome) which represents a possible solution to the task being tackled, i.e. a single point in the Search Space. Other information is usually also stored in each individual, e.g. its Fitness.

Inversion:

(EC) A Reordering operator which works by selecting two cut points in a Chromosome, and reversing the order of all the Genes between those two points.

Lamarckism:

Theory of Evolution which preceded Darwin's. Lamarck believed that evolution came about through the inheritance of acquired characteristics. That is, the skills or physical features which an Individual acquires during its lifetime can be passed on to its offspring. Although Lamarckian inheritance does not take place in nature, the idea has been usefully applied by some in EC. cf DARWINISM.

Learning Classifier System:

A Classifier System which "learns" how to classify its inputs. This often involves "showing" the system many examples of input patterns, and their corresponding correct outputs.

Migration:

The transfer of (the Genes of) an Individual from one Sub-Population to another.

Mutation:

(EC) a Reproduction Operator which forms a new Chromosome by making (usually small) alterations to the values of Genes in a copy of a single, Parent chromosome.

Niche:

(biol) In natural ecosystems, there are many different ways in which animals may survive (grazing, hunting, on the ground, in trees, etc.), and each survival strategy is called an "ecological niche." Species which occupy different Niches (e.g. one eating plants, the other eating insects) may coexist side by side without competition, in a stable way. But if two species occupying the same niche are brought into the same area, there will be competition, and eventually the weaker of the two species will be made (locally) extinct. Hence diversity of species depends on them occupying a diversity of niches (or on geographical separation).

(EC) In EC, we often want to maintain diversity in the Population. Sometimes a Fitness function may be known to be multimodal, and we want to locate all the peaks. We may consider each peak in the fitness function as analogous to a Niche. By applying techniques such as fitness sharing (Goldberg & Richardson, [ICGA87]), the population can be prevented from converging on a single peak, and instead

stable Sub-Populations form at each peak. This is analogous to different Species occupying different niches. See also Species, Speciation.

Object Variables:

Parameters that are directly involved in assessing the relative worth of an Individual.

Offspring:

An Individual generated by any process of Reproduction.

Optimization:

The process of iteratively improving the solution to a problem with respect to a specified objective function.

Order-Based Problem:

A problem where the solution must be specified in terms of an arrangement (e.g. a linear ordering) of specific items, e.g. Traveling Salesman Problem, computer process scheduling. Order-Based Problems are a class of Combinatorial Optimization problems in which the entities to be combined are already determined. cf Value-Based Problem.

Ontogenesis:

Refers to a single organism, and means the life span of an organism from it's birth to death. cf Phylogenesis.

Panmictic Population:

(EC, biol) A mixed Population. A population in which any Individual may be mated with any other individual with a probability which depends only on Fitness. Most conventional evolutionary algorithms have Panmictic Populations.

The opposite is a Population divided into groups known as Sub- Populations, where Individuals may only mate with others in the same sub-population. cf Speciation.

Parent:

An individual which takes part in reproduction to generate one or more other individuals, known as offspring, or children.

Phenotype:

The expressed traits of an individual.

Phylogenesis:

Refers to a population of organisms. The life span of a population of organisms from pre-historic times until today. Cf ontogenesis.

Plus Strategy:

Notation originally proposed in evolution strategies, when a population of μ parents generates λ offspring and all μ and λ individuals compete directly, the process is written as a $(\mu + \lambda)$ search. The process of competing all parents and offspring then is a "plus strategy." Cf. Comma strategy.

Population:

A group of individuals which may interact together, for example by mating, producing offspring, etc. Typical population sizes in EC range from 1 (for certain evolution strategies) to many thousands (for genetic programming). Cf sub- population.

Recombination: cf Crossover.

Reordering:

(EC) a reordering operator is a reproduction operator which changes the order of genes in a chromosome, with the hope of bringing related genes closer together, thereby facilitating the production of building blocks. Cf inversion.

Reproduction:

(biol, EC) the creation of a new individual from two parents (sexual reproduction). Asexual reproduction is the creation of a new individual from a single parent.

Reproduction Operator:

(EC) A mechanism which influences the way in which genetic information is passed on from parent(s) to offspring during reproduction. Reproduction operators fall into three broad categories: Crossover, Mutation and reordering operators.

Requisite Variety:

In genetic algorithms, when the population fails to have a "requisite variety" Crossover will no longer be a useful search operator because it will have a propensity to simply regenerate the parents.

Schema:

A pattern of gene values in a chromosome, which may include `dont care' states. Thus in a binary chromosome, each schema (plural schemata) can be specified by a string of the same length as the chromosome, with each character one of {0, 1, #}. A particular chromosome is said to `contain' a particular schema if it matches the schema (e.G. Chromosome 01101 matches schema #1#0#).

The `order' of a schema is the number of non-don’t-care positions specified, while the `defining length' is the distance between the furthest two non-don’t-care positions. Thus #1##0# is of order 2 and defining length 3.

Schema Theorem:

Theorem devised by Holland [holland92] to explain the behavior of gas. In essence, it says that a GA gives exponentially increasing reproductive trials to above average schemata.

Because each chromosome contains a great many schemata, the rate of schema processing in the population is very high, leading to a phenomenon known as implicit parallelism. This gives a GA with a population of size n a speedup by a factor of n cubed, compared to a random search.

Search Space:

If the solution to a task can be represented by a set of n real-valued parameters, then the job of finding this solution can be thought of as a search in an n-dimensional space. This is referred to simply as the search space. More generally, if the solution to a task can be represented using a representation scheme, r, then the search space is the set of all possible configurations which may be represented in r.

Search Operators:

Processes used to generate new individuals to be evaluated. Search operators in Genetic Algorithms are typically based on Crossover and point Mutation. Search operators in evolution strategies and evolutionary programming typically follow from the representation of a solution and often involve Gaussian or lognormal perturbations when applied to real-valued vectors.

Selection:

The process by which some individuals in a population are chosen for reproduction, typically on the basis of favoring individuals with higher fitness.

Self-Adaptation:

The inclusion of a mechanism not only to evolve the object variables of a solution, but simultaneously to evolve information on how each solution will generate new offspring.

Simulation:

The act of modeling a natural process.

Soft Selection:

The mechanism which allows inferior individuals in a population a non-zero probability of surviving into future generations. See hard Selection.

Speciation:

(biol) the process whereby a new species comes about. The most common cause of speciation is that of geographical isolation. If a sub-population of a single species is separated geographically from the main population for a sufficiently long time, their genes will diverge (either due to differences in Selection pressures in different locations, or simply due to genetic drift). Eventually, genetic differences will be so great that members of the sub-population must be considered as belonging to a different (and new) species.

Speciation is very important in evolutionary biology. Small sub- populations can evolve much more rapidly than a large population (because genetic changes don't take long to become fixed in the population). Sometimes, this evolution will produce superior individuals which can outcompete, and eventually replace the species of the original, main population.

(EC) techniques analogous to geographical isolation are used in a number of gas. Typically, the population is divided into sub- populations, and individuals are only allowed to mate with others in the same sub-population. (a small amount of migration is performed.)

This produces many sub-populations which differ in their characteristics, and may be referred to as different "species". This technique can be useful for finding multiple solutions to a problem, or simply maintaining diversity in the search space.

Most biology/genetics textbooks contain information on Speciation. A more detailed account can be found in "Genetics, Speciation and the Founder Principle", L.V. Giddings, K.Y. Kaneshiro and W.W. Anderson (Eds.), Oxford University Press 1989.

Species:

(biol) There is no universally-agreed firm definition of a Species. A species may be roughly defined as a collection of living creatures, having similar characteristics, which can breed together to produce viable offspring similar to their parents. Members of one species occupy the same ecological Niche. (Members of different species may occupy the same, or different niches.)

(EC) In EC the definition of "species" is less clear, since generally it is always possible for a pair Individuals to breed together. It is probably safest to use this term only in the context of algorithms which employ explicit SPECIATION mechanisms.

(biol) The existence of different Species allows different ecological Niches to be exploited. Furthermore, the existence of a variety of different species itself creates new niches, thus allowing room for further species. Thus nature bootstraps itself into almost limitless complexity and diversity.

Conversely, the domination of one, or a small number of Species reduces the number of viable Niches, leads to a decline in diversity, and a reduction in the ability to cope with new situations.

Step size:

Typically, the average distance in the appropriate space between a Parent and its offspring.

Strategy Variable:

Evolvable parameters that relate the distribution of offspring from a Parent.

Sub-Population:

A Population may be sub-divided into groups, known as Sub-Populations, where Individuals may only mate with others in the same group. (This technique might be chosen for parallel processors). Such sub-divisions may markedly influence the evolutionary dynamics of a population (e.g. Wright's 'shifting balance' model). Sub-populations may be defined by various Migration constraints: islands with limited arbitrary migration; stepping-stones with migration to neighboring

islands; isolation-by-distance in which each individual mates only with near neighbors. cf Panmictic Population, Speciation.

Terminal Set:

(GP) The set of terminal (leaf) nodes in the parse trees representing the programs in the Population. A terminal might be a variable, such as X, a constant value, such as 42, or a function taking no arguments, such as (move-north).

Traveling Salesman Problem:

The traveling salesperson has the task of visiting a number of clients, located in different cities. The problem to solve is: in what order should the cities be visited in order to minimize the total distance traveled (including returning home)? This is a classical example of an Order-Based Problem.

Value-Based Problem:

A problem where the solution must be specified in terms of a set of real-valued parameters. Function Optimization problems are of this type. cf Search Space, Order-Based Problem.

Vector Optimization:

Typically, an Optimization problem wherein multiple objectives must be satisfied.

Vladimiro Miranda received his Licenciado, Ph.D. and Agregado degrees from the Faculty of Engineering of the University of Porto, Portugal (FEUP) in 1977, 1982 and 1991, all in Electrical Engineering.

In 1981 he joined FEUP and currently holds the position of Professor Associado Agregado. In 1985 he joined also INESC, a research and development institute, having held for many years the position of Head of Information and Decision in Energy Systems. In 1996 he was appointed President of the Executive Board of INESC-Macau (South China) and Full Professor in the University of Macau.

He is currently the President of the Scientific Board of INESC Porto and Executive Board Adviser for Power Systems.

He is a member of several Expert Committees in Power Systems and of EAF - Energy Advisory Foundation. He has had responsibility over several research projects within the European Union programmes and also in cooperation with Latin America and Portuguese speaking African countries. He is a consultant of the Innovation Agency of the Portuguese Government for the development of research and innovation projects with China.

He has authored or co-authored many papers, namely in his areas of interest, related with Power System planning and the application of fuzzy sets and other soft computing techniques to Power Systems.