The Hitch-Hiker's Guide to Evolutionary Computation

(FAQ for comp.ai.genetic)

edited by

Jörg Heitkötter
UUNET Deutschland GmbH
Emil-Figge-Str. 80
D-44227 Dortmund, Germany
<joke@de.uu.net>
or <joke@santafe.edu>

and

David Beasley
Deloitte Consulting
20 Manvers Street,
Bath, United Kingdom BA1 1PX
<david.beasley@iee.org>

PLEASE:
Search this document first if you have a question
and
If someone posts a question to the newsgroup which is answered in here
DON'T POST THE ANSWER TO THE NEWSGROUP:
POINT THE ASKER TO THIS FAQ
and finally

DON'T PANIC!

Copyright (c) 1993-1998 by Jörg Heitkötter and David Beasley, all rights reserved.

This FAQ may be posted to any USENET newsgroup, on-line service, or BBS as long as it is posted in its entirety and includes this copyright statement. This FAQ may not be distributed for financial gain. This FAQ may not be included in commercial collections or compilations without express permission from the author.

FAQ /F-A-Q/ or /fak/ **[USENET]** n. 1. A Frequently Asked Question. 2. A compendium of accumulated lore, posted periodically to high-volume newsgroups in an attempt to forestall such questions. Some people prefer the term 'FAQ list' or 'FAQL' /fa'kl/, reserving 'FAQ' for sense 1.

RTFAQ /R-T-F-A-Q/ [USENET: primarily written, by analogy with *RTFM*] imp. Abbrev. for 'Read the FAQ!', an exhortation that the person addressed ought to read the newsgroup's *FAQ list* before posting questions.

RTFM /R-T-F-M/ [UNIX] imp. Acronym for 'Read The Fucking Manual'. 1. Used by *guru*s to brush off questions they consider trivial or annoying. Compare *Don't do that, then!* 2. Used when reporting a problem to indicate that you aren't just asking out of *randomness*. "No, I can't figure out how to interface UNIX to my toaster, and yes, I have RTFM." Unlike sense 1, this use is considered polite. ...

−−− "The on-line hacker Jargon File, version 3.0, 29 July 1993"

PREFACE

This guide is intended to help, provide basic information, and serve as a first straw for individuals, i.e. uninitiated *hitch-hikers*, who are stranded in the mindboggling universe of *Evolutionary Computation* **(EC);** that in turn is only a small footpath to an even more mindboggling scientific universe, that, incorporating *Fuzzy Systems,* and *Artificial Neural Networks,* is sometimes referred to as *Computational Intelligence* (CI); that in turn is only part of an even more advanced scientific universe of mindparalysing complexity, that incorporating *Artificial Life, Fractal Geometry,* and other *Complex Systems Sciences* 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,** which in turn 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, that are based upon biological observations, that date back to Charles Darwin's discoveries in the 19th century: the means of *natural selection* and the *survival of the fittest,* and theories of evolution. The inspired algorithms are thus termed *Evolutionary Algorithms* **(EA).**

Moreover, this guide is intended to help those who are just beginning to read the **comp.ai.genetic** newsgroup, and those who are new "on" **USENET.** It shall help to avoid lengthy discussions of questions that usually arise for beginners of one or the other kind, and which are boring to read again and again by **comp.ai.genetic** "old-timers."

You will see this guide popping up periodically in the Usenet newsgroup **comp.ai.genetic** (and also **comp.answers** , and **news.answers** , where it should be locatable at any time).

ORIGIN

This guide was produced by Jörg Heitkötter (known as *Joke*) in early 1993, using material from many sources (see Acknowledgements), mixed with his own brand of humour. Towards the end of 1993, Jörg handed over editorial responsibility to David Beasley . He reorganised the guide in various ways, and generally attempted to inject his own brand of orderliness. Thus, any jokes are the work of *Joke*. The mundane bits are David's responsibility.

The guide is kept up to date, as far as possible, and new versions are issued several times a year. However, we do rely on useful information being sent to us for inclusion in the guide (we dont always have time to read **comp.ai.genetic** , for example). Contributions, additions, corrections, cash, etc. are therefore always welcome. Send e-mail to the address at the beginning of the guide.

DISCLAIMER

This periodic posting is not meant to discuss any topic exhaustively, but should be thought of as a *list of reference pointers,* instead. This posting is provided on an "as is" basis, NO WARRANTY whatsoever is expressed or implied, especially, NO WARRANTY that the information contained herein is up-to-date, correct or useful in any way, although all this is intended.

Moreover, please note that the opinions expressed herein do not necessarily reflect those of the editors' institutions or employers, neither as a whole, nor in part. They are just the amalgamation of the editors' collections of ideas, and contributions gleaned from other sources.

NOTE: some portions of this otherwise rather dry guide are intended to be satirical. If you do not recognize it as such, consult your local doctor or a professional comedian.

HOW TO USE THIS GUIDE

HITCH-HIKING THE FAQNIVERSE

This guide is big. Really big. You just won't believe how hugely, vastly, mindbogglingly big it is. That's why it has been split into a "trilogy" -- which, like all successful trilogies, eventually ends up consisting of more than three parts.

Searching for answers

To find the answer of question number *x*, just search for the string "Q*x*:". (So the answer to question 42 is at "Q42:"!)

What does [xxxx99] mean?

Some books are referenced again and again, that's why they have this kind of "tag", that an experienced *hitch-hiker* will search for in the list of books (see Q10 and Q12 and other places) to dissolve the riddle. Here, they have a ":" appended, thus you can search for the string "[ICGA85]:" for example.

Why all this UPPERCASING in running text?

Words written in all uppercase letters are cross-references to entries in the *Glossary* (see Q99). Again, they have a ":" appended, thus if you find, say **EVOLUTION,** you can search for the string "EVOLUTION:" in the *Glossary*.

FTP and HTTP naming conventions

A file available on an **FTP** server will be specified as: **<ftp-site-name>/<the-complete-filename>** So for example, the file **bar.tar.gz** in the directory **/pub/foo** on the ftp server **ftp.certain.site** would be specified as: **ftp.certain.site/pub/foo/bar.tar.gz**

A specification ending with a "/" is a reference to a whole directory, e.g. **ftp.certain.site/pub/foo/**

HTTP files are specified in a similar way, but with the prefix: **http://**

WHERE TO FIND THIS GUIDE

Between postings to **comp.ai.genetic** , this **FAQ** is available on the World Wide Web. Get it from any **ENCORE** site (See Q15.3). The following Encore sites can be accessed by HTTP. If you use the one closest to you, you should get the best speed of service.

- UUNET Deutschland GmbH (Germany): **http://research.de.uu.net:8080/encore/www/**
- The University of Girona (Spain) **http://gnomics.udg.es/˜encore/www/**
- The University of Granada (Spain): **http://krypton.ugr.es/˜encore/www/**
- The University of Oviedo (Spain): **http://www.etsimo.uniovi.es/ftp/pub/EC/FAQ/www/**
- The University of Birmingham (UK) **http://www.cs.bham.ac.uk/Mirrors/ftp.de.uu.net/EC/clife/www/**
- The Santa Fe Institute (USA): **http://alife.santafe.edu/˜joke/encore/www/**
- Purdue University, West Lafayette, IN (USA): **http://www.cs.purdue.edu/coast/archive/clife/FAQ/www/**
- The Chinese University of Hong Kong: **http://www.cs.cuhk.hk/pub/EC/FAQ/www/top.htm**

Other Encore sites can be accessed by **FTP,** and the FAQ can be found in the file **FAQ/www/top.htm** or something similar. The FAQ is also available in plain text format on Encore, and from **rtfm.mit.edu/pub/usenet/news.answers/ai-faq/genetic/** as the files: part1 to part6. The FAQ may also be retrieved by e-mail from <mail-server@rtfm.mit.edu>. Send a message to the mail-server with "help" and "index" in the body on separate lines for more information.

A PostScript version is also available. This looks really crisp (using **boldface**, *italics*, etc.), and is available for those who prefer offline reading. Get it from Encore in file **FAQ/hhgtec.ps.gz** (the plain text versions are in the same directory too).

"As a net is made up of a series of ties, so everything in this world is connected by a series of ties. If anyone thinks that the mesh of a net is an independent, isolated thing, he is mistaken. It is called a net because it is made up of a series of interconnected meshes, and each mesh has its place and responsibility in relation to other meshes."

——— Buddha

Referencing this Guide

If you want to reference this guide it should look like:

Heitkötter, Jörg 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.

Or simply call it *"the Guide"*, or *"HHGTEC"* for acronymaniacs.

The ZEN Puzzle

For some weird reason this guide contains some puzzles which can only be solved by cautious readers who have (1) a certain amount of a certain kind of humor, (2) a certain amount of patience and time, (3) a certain amount of experience in **ZEN NAVIGATION,** and (4) a certain amount of books of a certain author.

Usually, puzzles search either for certain answers (more often, ONE answer) to a question; or, for the real smartasses, sometimes an answer is presented, and a certain question is searched for. ZEN puzzles are even more challenging: you have to come up with an answer to a question, both of which are not *explicitly*, rather *implicitly* stated somewhere in this **FAQ.** Thus, you are expected to give an answer AND a question!

To give an impression *what this is all about*, consider the following, submitted by *Craig W. Reynolds.* The correct question is: "Why is Fisher's ‘improbability quote’ (cf EPILOGUE) included in this FAQ?", Craig's correct answer is: *‘This is a GREAT quotation, it sounds like something directly out of a turn of the century Douglas Adams: Natural Selection:*
the original "Infinite Improbability Drive"’ Got the message? Well, this was easy and very obvious. The other puzzles are more challenging...

However, all this is just for fun (mine and hopefully yours), there is nothing like the $100 price, some big shots in computer science, e.g. *Don Knuth* usually offer; all there is but a honorable mentioning of the ZEN navigator, including the puzzle s/he solved. It's thus like in real life: don't expect to make money from your time being a scientist, it's all just for the fun of it...

Enjoy the trip!

TABLE OF CONTENTS

Q0: How about an introduction to comp.ai.genetic?

Certainly. See below.

What is comp.ai.genetic all about?

The newsgroup **comp.ai.genetic** is intended as a forum for people who want to use or explore the capabilities of *Genetic Algorithms* **(GA),** *Evolutionary Programming* **(EP),** *Evolution Strategies* **(ES),** *Classifier Systems* **(CFS),** *Genetic Programming* **(GP),** and some other, less well-known problem solving algorithms that are more or less loosely coupled to the field of *Evolutionary Computation* **(EC).**

How do I get started? What about USENET documentation?

The following guidelines present the essentials of the **USENET** online documentation, that is posted each month to **news.announce.newusers**

If you are already familiar with "netiquette" you can skip to the end of this answer; if you don't know *what the hell this is all about,* proceed as follows: (1) carefully read the following paragraphs, (2) read all the documents in **news.announce.newusers** before posting any article to USENET. At least you should give the introductory stuff a try, i.e. files "news-answers/introduction" and "news-answers/news-newusers-intro". Both are survey articles, that provide a short and easy way to get an overview of the interesting parts of the online docs, and thus can help to prevent you from drowning in the megabytes to read. Both can be received either by subscribing to **news.answers** , or sending the following message to <mail-server@rtfm.mit.edu>:

```
send usenet/news.answers/introduction
send usenet/news.answers/news-newusers-intro
quit
```

Netiquette

"Usenet is a convention, in every sense of the word."

Although **USENET** is usually characterized as "an anarchy, with no laws and no one in charge" there have "emerged" several rules over the past years that shall facilitate *life* within newsgroups. Thus, you will probably find the following types of articles:

1. **Requests**

   Requests are articles of the form "I am looking for X" where X is something public like a book, an article, a piece of software.

   If multiple different answers can be expected, the person making the request should prepare to make a summary of the answers he/she got and announce to do so with a phrase like "Please e-mail, I'll summarize" at the end of the posting.

   The Subject line of the posting should then be something like "Request: X"

2. **Questions**

   As opposed to requests, questions are concerned with something so specific that general interest cannot readily be assumed. If the poster thinks that the topic is of some general interest, he/she should announce a summary (see above).

   The Subject line of the posting should be something like "Question: this-and-that" (Q: this-and-that) or have the form of a question (i.e., end with a question mark)

3. **Answers**

   These are reactions to questions or requests. As a rule of thumb articles of type "answer" should be rare. Ideally, in most cases either the answer is too specific to be of general interest (and should thus be e-mailed to the poster) or a summary was announced with the question or request (and answers should thus be e-mailed to the poster).

The subject lines of answers are automatically adjusted by the news software.

4. Summaries

In all cases of requests or questions the answers for which can be assumed to be of some general interest, the poster of the request or question shall summarize the answers he/she received. Such a summary should be announced in the original posting of the question or request with a phrase like "Please answer by e-mail, I'll summarize"

In such a case answers should NOT be posted to the newsgroup but instead be mailed to the poster who collects and reviews them. After about 10 to 20 days from the original posting, its poster should make the summary of answers and post it to the net.

Some care should be invested into a summary:

a) simple concatenation of all the answers might not be enough; instead redundancies, irrelevances, verbosities and errors should be filtered out (as good as possible),

b) the answers shall be separated clearly

c) the contributors of the individual answers shall be identifiable unless they requested to remain anonymous [eds note: yes, that happens])

d) the summary shall start with the "quintessence" of the answers, as seen by the original poster

e) A summary should, when posted, clearly be indicated to be one by giving it a Subject line starting with "Summary:"

Note that a good summary is pure gold for the rest of the newsgroup community, so summary work will be most appreciated by all of us. *(Good summaries are more valuable than any moderator!)*

5. Announcements

Some articles never need any public reaction. These are called announcements (for instance for a workshop, conference or the availability of some technical report or software system).

Announcements should be clearly indicated to be such by giving them a subject line of the form "Announcement: this-and-that", or "ust "A: this-and-that".

Due to common practice, conference announcements usually carry a "CFP:" in their subject line, i.e. "call for papers" (or: "call for participation").

6. Reports

Sometimes people spontaneously want to report something to the newsgroup. This might be special experiences with some software, results of own experiments or conceptual work, or especially interesting information from somewhere else.

Reports should be clearly indicated to be such by giving them a subject line of the form "Report: this-and-that"

7. Discussions

An especially valuable possibility of **USENET** is of course that of discussing a certain topic with hundreds of potential participants. All traffic in the newsgroup that can not be subsumed under one of the above categories should belong to a discussion.

If somebody explicitly wants to start a discussion, he/she can do so by giving the posting a subject line of the form "Start discussion: this-and-that" (People who react on this, please remove the "Start discussion: " label from the subject line of your replies)

It is quite difficult to keep a discussion from drifting into chaos, but, unfortunately, as many other newsgroups show there seems to be no secure way to avoid this. On the other hand, **comp.ai.genetic** has not had many problems with this effect, yet, so let's just go and hope...

Thanks in advance for your patience!

The Internet

For information on internet services, see Q15.5.

Q1: 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 mechanisms of **EVOLUTION** as key elements in their design and implementation. A variety of **EVOLUTIONARY ALGORITHMs** have been proposed. The major ones are: **GENETIC ALGORITHMs** (see Q1.1), **EVOLUTIONARY PROGRAMMING** (see Q1.2), **EVOLUTION STRATEGIEs** (see Q1.3), **CLASSIFIER SYSTEMs** (see Q1.4), and **GENETIC PROGRAMMING** (see Q1.5). 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 (cf. **EXPLOITATION)** 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.

--- "An Overview of Evolutionary Computation" [ECML93], 442-459.

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 shuffing 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. 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 randomising 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 may have 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).

PSEUDO CODE

```
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 its new fitness
        evaluate P' (t);

        // select the survivors from actual fitness
        P := survive P,P' (t);
    od
end EA.
```

Q1.1: What's a Genetic Algorithm (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 focussed on fixed-length character strings. We should focus on both this aspect of fixed-lengthness 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 behavior 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.

PSEUDO CODE

```
Algorithm GA is

    // start with an initial time
    t := 0;

    // initialize 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

        // increase the time counter
        t := t + 1;

        // select a 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 its new fitness
                evaluate P' (t);
                // select the survivors from actual fitness
                P := survive P,P' (t);
        od
    end GA.
```

Q1.2: What's Evolutionary Programming (EP)?

Introduction

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 (see below).

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. Combinatoric 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.

History

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 (See Q12). 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 its 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.**

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.

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:

1. Selection: EP typically uses stochastic selection via a tournament. Each trial solution in the population faces competition against a preselected 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.
2. **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.

References

Some references which provide an excellent introduction (by no means extensive) to the field, include:

Artificial Intelligence Through Simulated Evolution [Fogel66] (primary)

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

Proceeding of the first [EP92], second [EP93] and third [EP94] Annual Conference on Evolutionary Programming (primary) (See Q12).

PSEUDO CODE

```
Algorithm EP is

    // start with an initial time
    t := 0;

    // initialize 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 its 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;

    od
end EP.
```

[Eds note: An extended version of this introduction is available from **ENCORE** (see Q15.3) in **/FAQ/supplements/what-is-ep**]

Q1.3: What's an Evolution Strategy (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* (cf Q5) 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.

The first attempts to imitate principles of organic evolution on a computer still resembled the iterative optimization methods known up to that time (cf Q5): In a two-membered or (1+1) ES, one **PARENT** generates one **OFFSPRING** per **GENERATION** by applying **NORMALLY DISTRIBUTED** mutations, i.e. smaller steps occur more likely than big ones, until a child performs better than its ancestor and takes its place. Because of this simple structure, theoretical results for **STEPSIZE** control and **CONVERGENCE VELOCITY** could be derived. The ratio between successful and all mutations should come to 1/5: the so-called **1/5 SUCCESS RULE** was discovered. This first algorithm, using mutation only, has then been enhanced to a (*m*+1) strategy which incorporated **RECOMBINATION** due to several, i.e. *m* parents being available. The mutation scheme and the exogenous stepsize control were taken across unchanged from (1+1) ESs.

Schwefel later generalized these strategies to the multimembered ES now denoted by (*m+l*) and (*m,l*) which imitates the following basic principles of organic evolution: a **POPULATION,** leading to the possibility of recombination with random mating, mutation and selection. These strategies are termed **PLUS STRATEGY** and **COMMA STRATEGY,** respectively: in the plus case, the parental generation is taken into account during selection, while in the comma case only the offspring undergoes selection, and the parents die off. *m* (usually a lowercase *mu*, denotes the population size, and *l*, usually a lowercase *lambda* denotes the number of offspring generated per generation). Or to put it in an utterly insignificant and hopelessly outdated language:

```
(define (Evolution-strategy population)
  (if (terminate? population)
    population
    (evolution-strategy
      (select
        (cond (plus-strategy?
                (union (mutate
                         (recombine population))
                       population))
              (comma-strategy?
                (mutate
                  (recombine population))))))))
```

However, dealing with ES is sometimes seen as "strong tobacco," for it takes a decent amount of probability theory and applied **STATISTICS** to understand the inner workings of an ES, while it navigates through the hyperspace of the usually n-dimensional problem space, by throwing hyperelipses into the deep...

Luckily, this medium doesn't allow for much mathematical ballyhoo; the author therefore has to come up with a simple but brilliantly intriguing explanation to save the reader from falling asleep during the rest of this section, so here we go:

Imagine a black box. A large black box. As large as, say for example, a Coca-Cola vending machine. Now, [..] (to be continued)

A single **INDIVIDUAL** of the ES' population consists of the following **GENOTYPE** representing a point in the **SEARCH SPACE:**

OBJECT VARIABLES

Real-valued x_i have to be tuned by recombination and mutation such that an objective function reaches its global optimum. Referring to the metaphor mentioned previously, the x_i represent the regulators of the alien Coka-Cola vending machine.

STRATEGY VARIABLEs

Real-valued s_i (usually denoted by a lowercase sigma) or mean stepsizes determine the mutability of the x_i. They represent the **STANDARD DEVIATION** of a (0, s_i) **GAUSSIAN DISTRIBUTION** (GD) being added to each x_i as an undirected mutation. With an "expectancy value" of 0 the parents will produce offspring similar to themselves on average. In order to make a doubling and a halving of a stepsize equally probable, the s_i mutate log-normally, distributed, i.e. exp(GD), from generation to generation. These stepsizes hide the internal model the population has made of its **ENVIRONMENT,** i.e. a **SELF-ADAPTATION** of the stepsizes has replaced the exogenous control of the (1+1) ES.

This concept works because selection sooner or later prefers those individuals having built a good model of the objective function, thus producing better offspring. Hence, learning takes place on two levels: (1) at the genotypic, i.e. the object and strategy variable level and (2) at the phenotypic level, i.e. the **FITNESS** level.

Depending on an individual's x_i, the resulting objective function value f(x), where x denotes the vector of objective variables, serves as the **PHENOTYPE** (fitness) in the selection step. In a plus strategy, the *m* best of all (*m*+*l*) individuals survive to become the parents of the next generation. Using the comma variant, selection takes place only among the *l* offspring. The second scheme is more realistic and therefore more successful, because no individual may survive forever, which could at least theoretically occur using the plus variant. Untypical for conventional optimization algorithms and lavish at first sight, a comma strategy allowing intermediate deterioration performs better! Only by *forgetting* highly fit individuals can a permanent adaptation of the stepsizes take place and avoid long stagnation phases due to misadapted s_i's. This means that these individuals have built an internal model that is no longer appropriate for further progress, and thus should better be discarded.

By choosing a certain ratio *m/l*, one can determine the convergence property of the evolution strategy: If one wants a fast, but local convergence, one should choose a small **HARD SELECTION,** ratio, e.g. (5,100), but looking for the global optimum, one should favour a softer selection (15,100).

Self-adaptation within ESs depends on the following agents (Schwefel 87):

Randomness: One cannot model mutation
as a purely random process. This would mean that a child is completely independent of its parents.

Population size: The population has to be sufficiently large. Not only
the current best should be allowed to reproduce, but a set of good individuals. Biologists have coined the term "requisite variety" to mean the genetic variety necessary to prevent a **SPECIES** from becoming poorer and poorer genetically and eventually dying out.

COOPERATION:
In order to exploit the effects of a population (m > 1), the individuals should recombine their knowledge with that of others (cooperate) because one cannot expect the knowledge to accumulate in the best individual only.

Deterioration: In order to allow better internal models (stepsizes)
to provide better progress in the future, one should accept deterioration from one generation to the next. A limited life-span in nature is not a sign of failure, but an important means of preventing a species from freezing genetically.

ESs prove to be successful when compared to other iterative methods on a large number of test problems (Schwefel 77). They are adaptable to nearly all sorts of problems in optimization, because they need very little information about the problem, especially no derivatives of the objective function. For a list of some 300 applications of **EAs,** see the SyS-2/92 report (cf Q14). ESs are capable of solving *high dimensional, multimodal, nonlinear problems* subject to *linear and/or nonlinear constraints.* The objective function can also, e.g. be the result of a **SIMULATION,** it does not have to be given in a closed form. This also holds for the constraints which may represent the outcome of, e.g. a finite elements method (FEM). ESs have been adapted to **VECTOR OPTIMIZATION** problems (Kursawe 92), and they can also serve as a heuristic for NP-complete combinatorial problems like the **TRAVELLING SALESMAN PROBLEM** or problems with a noisy or changing response surface.

References

Kursawe, F. (1992) " Evolution strategies for vector optimization", Taipei, National Chiao Tung University, 187-193.

Kursawe, F. (1994) " Evolution strategies: Simple models of natural processes?", Revue Internationale de Systémique, France (to appear).

Rechenberg, I. (1973) "Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution", Stuttgart: Fromman-Holzboog.

Schwefel, H.-P. (1977) "Numerische Optimierung von Computermodellen mittels der Evolutionsstrategie", Basel: Birkhäuser.

Schwefel, H.-P. (1987) "Collective Phaenomena in Evolutionary Systems", 31st Annu. Meet. Inter'l Soc. for General System Research, Budapest, 1025-1033.

Q1.4: What's a Classifier System (CFS)?

The name of the Game

First, a word on naming conventions is due, for no other paradigm of **EC** has undergone more changes to its *name space* than this one. Initially, Holland called his cognitive models "Classifier Systems" abbrv. with **CS,** and sometimes **CFS,** as can be found in [GOLD89].

Whence Riolo came into play in 1986 and Holland added a reinforcement component to the overall design of a CFS, that emphasized its ability to learn. So, the word "learning" was prepended to the name, to make: "Learning Classifier Systems" (abbrv. to **LCS).** On October 6-9, 1992 the "1st Inter'l Workshop on Learning Classifier Systems" took place at the NASA Johnson Space Center, Houston, TX. A summary of this "summit" of all leading researchers in LCS can be found on **ENCORE** (See Q15.3) in file **CFS/papers/lcs92.ps.gz**

Today, the story continues, LCSs are sometimes subsumed under a "new" machine learning paradigm called "Evolutionary Reinforcement Learning" or ERL for short, incorporating LCSs, "Q-Learning", devised by Watkins (1989), and a paradigm of the same name, devised by Ackley and Littman [ALIFEIII].

And then, this latter statement is really somewhat confusing, as Marco Dorigo has pointed out in a letter to editors of this guide, since Q-Learning has no evolutionary component. So please let the Senior Editor explain: When I wrote this part of the guide, I just had in mind that Q-Learning would make for a pretty good replacement of Holland's bucket-brigade algorithm, so I used this litte speculation to see what comes out of it; in early December 95, almost two years later, it has finally caught Marco's attention. But meanwhile, I have been proven right: Wilson has suggested to use Q-Learning in **CLASSIFIER SYSTEM** (Wilson 1994) and Dorigo & Bersini (1994) have shown that Q-Learning and the bucket-brigade are truly equivalent concepts.

We would therefore be allowed to call a CFS that uses Q-Learning for rule discovery, rather than a bucket-brigade, a Q-CFS, Q-LCS, or Q-CS; in any case would the result be subsumed under the term ERL, even if Q-Learning itself is not an **EVOLUTIONARY ALGORITHM!**

Interestingly, Wilson called his system **ZCS** (apparantly no "Q" inside), while Dorigo & Bersini called their system a **D-Max-VSCS,** or "discounted max very simple classifier system" (and if you know Q-Learning at least the **D-Max** part of the name will remind you of that algorithm). The latter paper can be found on Encore (see Q15.3) in file **CFS/papers/sab94.ps.gz**

And by the way in [HOLLAND95] the term "classifier system" is not used anymore. You cannot find it in the index. Its a gone! Holland now stresses the adaptive component of his invention, and simply calls the resulting systems **adaptive agents.** These agents are then implemented within the framework of his recent invention called **ECHO.**

(See **http://alife.santafe.edu/alife/software/echo.html** for more.)

On Schema Processors and ANIMATS

So, to get back to the question above, "What are CFSs?", we first might answer, "Well, there are many interpretations of Holland's ideas...what do you like to know in particular?" And then we'd start with a recitation from [HOLLAND75], [HOLLAND92], and explain all the **SCHEMA** processors, the broadcast language, etc. But, we will take a more comprehensive, and intuitive way to understand what **CLASSIFIER SYSTEMs** are all about.

The easiest road to explore the very nature of classifier systems, is to take the animat (ANIMAl + **ROBOT** = ANIMAT) "lane" devised by Booker (1982) and later studied extensively by Wilson (1985), who also coined the term for this approach. Work continues on animats but is often regarded as **ARTIFICIAL LIFE** rather than **EVOLUTIONARY COMPUTATION.** This thread of research has even its own conference series: "Simulation of Adaptive Behavior (SAB)" (cf Q12), including other notions from machine learning, connectionist learning, evolutionary robotics, etc. [NB: the latter is obvious, if an animat lives in a digital microcosm, it can be put into the real world by implantation into an autonomous robot vehicle, that has sensors/detectors (camera eyes, whiskers, etc.) and effectors (wheels, robot arms, etc.); so all that's needed is to use our algorithm as the "brain" of this vehicle, connecting the hardware parts with the software learning component. For a fascinating intro to the field see, e.g. Braitenberg (1984)]

classifier systems, however, are yet another offspring of John Holland's aforementioned book, and can be seen as one of the early applications of **GAs,** for **CFSs** use this **EVOLUTIONARY ALGORITHM** to adapt their behavior toward a changing **ENVIRONMENT,** as is explained below in greater detail.

Holland envisioned a cognitive system capable of classifying the goings on in its environment, and then reacting to these goings on appropriately. So what is needed to build such a system? Obviously, we need (1) an environment; (2) receptors that tell our system about the goings on; (3) effectors, that let our system manipulate its environment; and (4) the system itself, conveniently a "black box" in this first approach, that has (2) and (3) attached to it, and "lives" in (1).

In the animat approach, (1) usually is an artificially created digital world, e.g. in Booker's Gofer system, a 2 dimensional grid that contains "food" and "poison". And the Gofer itself, that walks across this grid and tries (a) to learn to distinguish between these two items, and (b) survive well fed.

Much the same for Wilson's animat, called "∗". Yes, its just an asterisk, and a "Kafka-esque naming policy" is one of the sign posts of the whole field; e.g. the first implementation by Holland and Reitmann 1978 was called CS-1, (cognitive system 1); Smith's Poker player LS-1 (1980) followed this "convention". Riolo's 1988 **LCS** implementations on top of his CFS-C library (cf Q20), were dubbed FSW-1 (Finite State World 1), and LET-SEQ-1 (LETter SEQuence predictor 1).

So from the latter paragraph we can conclude that environment can also mean something completely different (e.g. an infinite stream of letters, time serieses, etc.) than in the animat approach, but anyway; we'll stick to it, and go on.

Imagine a very simple animat, e.g. a simplified model of a frog. Now, we know that frogs live in (a) Muppet Shows, or (b) little ponds; so we chose the latter as our demo environment (1); and the former for a non-Kafka-esque name of our model, so let's dub it "Kermit".

Kermit has eyes, i.e. sensorial input detectors (2); hands and legs, i.e. environment-manipulating effectors (3); is a spicy-fly-detecting-and-eating device, i.e. a frog (4); so we got all the 4 pieces needed.

Inside the Black Box

The most primitive "black box" we can think of is a computer. It has inputs (2), and outputs (3), and a message passing system inbetween, that converts (i.e., *computes*), certain input messages into output messages, according to a set of rules, usually called the "program" of that computer. From the theory of computer science, we now borrow the simplest of all program structures, that is something called "production system" or PS for short. A PS has been shown to be computationally complete by Post (1943), that's why it is sometimes called a "Post System", and later by Minsky (1967). Although it merely consists of a set of if-then rules, it still resembles a full-fledged computer.

We now term a single "if-then" rule a "classifier", and choose a representation that makes it easy to manipulate these, for example by encoding them into binary strings. We then term the set of classifiers, a "classifier population", and immediately know how to breed new rules for our system: just use a **GA** to generate new rules/classifiers from the current **POPULATION!**

All that's left are the messages floating through the black box. They should also be simple strings of zeroes and ones, and are to be kept in a data structure, we call "the message list".

With all this given, we can imagine the goings on inside the black box as follows: The input interface (2) generates messages, i.e., 0/1 strings, that are written on the message list. Then these messages are matched against the condition-part of all classifiers, to find out which actions are to be triggered. The message list is then emptied, and the encoded actions, themselves just messages, are posted to the message list. Then, the output interface (3) checks the message list for messages concerning the effectors. And the cycle restarts.

Note, that it is possible in this set-up to have "internal messages", because the message list is not emptied after (3) has checked; thus, the input interface messages are *added* to the initially empty list. (cf Algorithm **CFS, LCS** below)

The general idea of the CFS is to start from scratch, i.e., from *tabula rasa* (without any knowledge) using a randomly generated classifier population, and let the system learn its program by induction, (cf Holland et al. 1986), this reduces the input stream to recurrent input patterns, that must be repeated over and over again, to enable the animat to classify its current situation/context and react on the goings on appropriately.

What does it need to be a frog?

Let's take a look at the behavior emitted by Kermit. It lives in its digital microwilderness where it moves around randomly. [NB: This seemingly "random" behavior is not that random at all; for more on instinctive, i.e., innate behavior of non-artificial animals see, e.g. Tinbergen (1951)]

Whenever a small flying object appears, that has no stripes, Kermit should eat it, because its very likely a spicy fly, or other flying insect. If it has stripes, the insect is better left alone, because Kermit had better not bother with wasps, hornets, or bees. If Kermit encounters a large, looming object, it immediately uses its effectors to jump away, as far as possible.

So, part of these behavior patterns within the "pond world", in **AI** sometimes called a "frame," from traditional knowledge representation techniques, Rich (1983), can be expressed in a set of "if <condition> then <action>" rules, as follows:

IF	**THEN**
small, flying object to the left	send @
small, flying object to the right	send %
small, flying object centered	send $

large, looming object	send !
no large, looming object	send *
* and @	move head 15 degrees left
* and %	move head 15 degrees right
* and $	move in direction head pointing
!	move rapidly away from direction head pointing

Now, this set of rules has to be encoded for use within a **CLASSIFIER SYSTEM.** A possible encoding of the above rule set in CFS-C (Riolo) classifier terminology. The condition part consists of two conditions, that are combined with a logical AND, thus must be met both to trigger the associated action. This structure needs a NOT operation, (so we get NAND, and know from hardware design, that we can build any computer solely with NANDs), in CFS-C this is denoted by the '~' prefix character (rule #5).

IF		**THEN**
0000,	00 00	00 00
0000,	00 01	00 01
0000,	00 10	00 10
1111,	01 ##	11 11
~1111,	01 ##	10 00
1000,	00 00	01 00
1000,	00 01	01 01
1000,	00 10	01 10
1111,	## ##	01 11

Obviously, string '0000' denotes *small,* and '00' in the fist part of the second column, denotes *flying.* The last two bits of column #2 encode the direction of the object approaching, where '00' means *left,* '01' means *right,* etc.

In rule #4 a the "don't care symbol" '#' is used, that matches '1' and '0', i.e., the position of the large, looming object, is completely arbitrary. A simple fact, that can save Kermit's (artificial) life.

PSEUDO CODE (Non-Learning CFS)

Algorithm CFS **is**

// start with an initial time
t := 0;

// an initially empty message list
initMessageList ML (t);

// and a randomly generated population of classifiers
initClassifierPopulation P (t);

// test for cycle termination criterion (time, fitness, etc.)
while not done **do**

// increase the time counter
t := t + 1;

// 1. detectors check whether input messages are present
ML := readDetectors (t);

// 2. compare ML to the classifiers and save matches
ML' := matchClassifiers ML,P (t);

// 3. process new messages through output interface
ML := sendEffectors ML' (t);

od

end CFS.

To convert the previous, non-learning **CFS** into a learning **CLASSIFIER SYSTEM, LCS,** as has been proposed in Holland (1986), it takes two steps:

(1) the major cycle has to be changed such that the activation of each classifier depends on some additional parameter, that can be modified as a result of experience, i.e. reinforcement from the **ENVIRONMENT;**

(2) and/or change the contents of the classifier list, i.e., generate new classifiers/rules, by removing, adding, or combining condition/action-parts of existing classifiers.

The algorithm thus changes accordingly:

PSEUDO CODE (Learning CFS)

```
Algorithm LCS is
        // start with an initial time
        t := 0;

        // an initially empty message list
        initMessageList ML (t);

        // and a randomly generated population of classifiers
        initClassifierPopulation P (t);

        // test for cycle termination criterion (time, fitness, etc.)
        while not done do
                // increase the time counter
                t := t + 1;

                // 1. detectors check whether input messages are present
                ML := readDetectors (t);

                // 2. compare ML to the classifiers and save matches
                ML' := matchClassifiers ML,P (t);

                // 3. highest bidding classifier(s) collected in ML' wins the
                // "race" and post the(ir) message(s)
                ML' := selectMatchingClassifiers ML',P (t);

                // 4. tax bidding classifiers, reduce their strength
                ML' := taxPostingClassifiers ML',P (t);

                // 5. effectors check new message list for output msgs
                ML := sendEffectors ML' (t);

                // 6. receive payoff from environment (REINFORCEMENT)
                C := receivePayoff (t);

                // 7. distribute payoff/credit to classifiers (e.g. BBA)
                P' := distributeCredit C,P (t);

                // 8. Eventually (depending on t), an EA (usually a GA) is
                // applied to the classifier population
                if criterion then
                        P := generateNewRules P' (t);
                else
                        P := P'
        od
end LCS.
```

What's the problem with CFSs?

Just to list the currently known problems that come with **CFSs,** would take some additional pages; therefore only some interesting papers dealing with unresolved riddles are listed; probably the best paper containing most of these is the aforementioned summary of the **LCS** Workshop:

Smith, R.E. (1992) "A report on the first Inter'l Workshop on LCSs" avail. from **ENCORE** (See Q15.3) in file **CFS/papers/lcs92.ps.gz**

Other noteworthy critiques on LCSs include:

Wilson, S.W. (1987) "Classifier Systems and the Animat Problem" Machine Learning, 2.

Wilson, S.W. (1988) "Bid Competition and Specificity Reconsidered" Complex Systems, 2(5):705-723.

Wilson, S.W. & Goldberg, D.E. (1989) "A critical review of classifier systems" [ICGA89], 244-255.

Goldberg, D.E., Horn, J. & Deb, K. (1992) "What makes a problem hard for a classifier system?" (containing the Goldberg citation below) is also available from Encore (See Q15.3) in file **CFS/papers/lcs92-2.ps.gz**

Dorigo, M. (1993) "Genetic and Non-genetic Operators in ALECSYS" Evolutionary Computation, 1(2):151-164. The technical report, the journal article is based on is avail. from Encore (See Q15.3) in file **CFS/papers/icsi92.ps.gz**

Smith, R.E. Forrest, S. & Perelson, A.S. (1993) "Searching for Diverse, Cooperative **POPULATIONs** with Genetic Algorithms" Evolutionary Computation, 1(2):127-149.

Conclusions?

Generally speaking: **"There's much to do in CFS** research!"

No other notion of **EC** provides more space to explore and if you are interested in a PhD in the field, you might want to take a closer look at CFS. However, be warned!, to quote Goldberg: *"classifier systems are a quagmire−−−a glorious, wondrous, and inventing quagmire, but a quagmire nonetheless."*

References

Booker, L.B. (1982) "Intelligent behavior as an adaption 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. **http://iridia.ulb.ac.be/dorigo/dorigo/conferences/IC.11-SAB94.ps.gz**

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

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. [HOLLAND95]:

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.

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.

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

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.

Q1.5: What's 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:

```
 +
/ \
a  *
  / \
  b  c
```

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.**

More information is available in the GP mailing list **FAQ.** (See Q15.2) and from **http://www-cs-faculty.stanford.edu/˜koza/**

Q2: What applications of EAs are there?

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.

Following is an incomplete *(sic!)* list of successful EA applications:

BIOCOMPUTING

Biocomputing, or Bioinformatics, is the field of biology dedicated to the automatic analysis of experimental data (mostly sequencing data). Several approaches to specific biocomputing problems have been described that involve the use of **GA, GP** and simulated annealing. General information about biocomputing (software, databases, misc.) can be found on the server of the European Bioinformatics Institute: **http://www.ebi.ac.uk/ebi_home.html ENCORE** has a good selection of pointers related to this subject. VSCN provides a detailed online course on bioinformatics: **http://www.techfak.uni-bielefeld.de/bcd/Curric/welcome.html**

There are three main domains to which GA have been applied in Bioinformatics: protein folding, RNA folding, sequence alignment.

Protein Folding

Proteins are one of the essential components of any form of life. They are made of twenty different types of amino acid. These amino acids are chained together in order to form the protein that can contain from a few to several thousands residues. In most of the cases, the properties and the function of a protein are a result of its three dimensional structure. It seems that in many cases this structure is a direct consequence of the sequence. Unfortunately, it is still very difficult/impossible to deduce the three dimensional structure, knowing only the sequence. A part of the VSCN on-line bioinformatics course is dedicated to the use of GAs in Protein Folding Prediction. It contains an extensive bibliography and a detailed presentation of the subject with LOTS of explanations and on-line papers. The URL is: **http://www.techfak.uni-bielefeld.de/bcd/Curric/ProtEn/contents.html**

Koza [KOZA92] gives one example of GP applied to Protein Folding. Davis [DAVIS91] gives an example of **DNA** conformation prediction (a closely related problem) in his *Handbook of GAs.*

RNA Folding

Describing the tertiary structure of an RNA molecule, is about as hard as for a protein, but describing the intermediate structure (secondary structure) is somehow easier because RNA molecules are using the same pairing rules as DNA, (Watson and Crick base pairing). There exist deterministic algorithms that given a set of constraints (rules), compute the more stable structure, but: (a) their time and memory requirement increase quadratically or more with the length of the sequences, and (b) they require simplified rules. Lots of effort has recently been put into applying GAs to this problem, and several papers can be found (on-line if your institute subscribes to these journals):

A genetic Algorithm Based Molecular Modelling Technique For RNA Stem-loop Structures H. Ogata, Y. Akiyama and M Kanehisa, Nucleic Acid Research, 1995, vol 23,3 419-426

An Annealing Mutation Operator in the GA for RNA folding B.A Shapiro and J. C. Wu, CABIOS, 1996, vol 12, 3, 171-180

The computer Simulation of RNA Folding Pathway Using a Genetic Algorithm A.P. Gultyaev, F.D.H van Batenburg and C. W. A. Pleij in Journal of Molecular Biology, 1995, vol 250 37-51

Simulated Annealing has also been applied successfully to this problem:

Description of RNA folding by SA M. Schmitz and G. Steger in Journal of Molecular Biology, 1995, 255, 245-266

Sequence Alignments

Sequence Alignment is another important problem of Bioinformatics. The aim is to align together several related sequences (from two to hundreds) given a cost function. For the most widely used cost functions, the problem has been shown to be NP-complete. Several attempts have been made using SA:

Multiple Sequence Alignment Using SA J. Kim, Sakti Pramanik and M.J. Chung, CABIOS, 1994, vol 10, 4, 419-426

Multiple Sequence Alignment by Parallel SA M. Isshikawa, T. Koya and al, CABIOS, 1993,vol 9, 3, 267-273

SAM, software which uses Hidden Markov Models for Multiple Sequence Alignment, can use SA to train the model. Several papers have been published on SAM. The software, documentation and an extensive bibliography can be found in: **http://www.cse.ucsc.edu/research/compbio/sam.html**

More recently, various software using different methods like Gibbs sampling or GAs has been developed:

A Gibbs Sampling Strategy for Multiple Alignment C.E. Lawrence, S. F. Altschull and al, Science, October 1993, vol 262, 208-214

SAGA: Sequence Alignment by Genetic Algorithm C. Notredame and D.G. Higgins, Nucleic Acid Research, 1995, vol 24, 8,
1515-1524

A beta release of SAGA (along with the paper) is available on the European Bioinformatics Institute anonymous **FTP** server: **ftp.ebi.ac.uk/pub/software/unix/saga.tar.Z**

CELLULAR PROGRAMMING: Evolution of Parallel Cellular Machines

Nature abounds in systems involving the actions of simple, locally-interacting components, that give rise to coordinated global behavior. These collective systems have evolved by means of natural **SELECTION** to exhibit striking problem-solving capacities, while functioning within a complex, dynamic **ENVIRONMENT.** Employing simple yet versatile parallel cellular models, coupled with **EVOLUTIONARY COMPUTATION** techniques, cellular programming is an approach for constructing man-made systems that exhibit characteristics such as those manifest by their natural counterparts.

Parallel cellular machines hold potential both scientifically, as vehicles for studying phenomena of interest in areas such as complex adaptive systems and **ARTIFICIAL LIFE,** as well as practically, enabling the construction of novel systems, endowed with evolutionary, reproductive, regenerative, and learning capabilities.

Web site: **http://lslwww.epfl.ch/˜moshes/cp.html**

References:

Sipper, M. (1997) "Evolution of Parallel Cellular Machines: The Cellular Programming Approach", Springer-Verlag, Heidelberg.

Sipper, M. (1996) "Co-evolving Non-Uniform Cellular Automata to Perform Computations", Physica D, 92, 193-208.

Sipper, M. and Ruppin, E. (1997) "Co-evolving architectures for cellular machines", Physica D, 99, 428-441.

Sipper, M. and Tomassini, M. (1996) "Generating Parallel Random Number Generators By Cellular Programming", International Journal of Modern Physics C, 7(2), 181-190.

Sipper, M. (1997) "Evolving Uniform and Non-uniform Cellular Automata Networks", in Annual Reviews of Computational Physics, D. Stauffer (ed)

Evolvable Hardware

The idea of evolving machines, whose origins can be traced to the cybernetics movement of the 1940s and the 1950s, has recently resurged in the form of the nascent field of bio-inspired systems and evolvable hardware. The field draws on ideas from the **EVOLUTIONARY COMPUTATION** domain as well as on novel hardware innovations. Recently, the term *evolware* has been used to describe such evolving ware, with current implementations centering on hardware, while raising the possibility of using other forms in the future, such as bioware. The inaugural workshop, *Towards Evolvable Hardware,* took place in Lausanne, in October 1995, followed by the *First International Conference on Evolvable Systems: From Biology to Hardware (ICES96)* held in Japan, in October 1996. Another major event in the field, ICES98, was held in Lausanne, Switzerland, in September 1998.

References:

Sipper, M. et al (1997) "A Phylogenetic, Ontogenetic, and Epigenetic View of Bio-Inspired Hardware Systems", IEEE Transactions on Evolutionary Computation, 1(1).

Sanchez, E. and Tomassini, M. (eds) (1996) "Towards Evolvable Hardware", Springer-Verlag, Lecture Notes in Computer Science, 1062.

Higuchi, T. et al (1997) "Proceedings of First International Conference on Evolvable Systems: From Biology to Hardware (ICES96)", Springer-Verlag, Lecture Notes in Computer Science.

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.

An alternative characterisation is to model the players as Finite State Machines, or Finite Automata (FA). These can be though of as a simple flow chart governing behaviour in the "next" play of the game depending upon previous plays. For example:

```
100 Play X
110 If opponent plays X go to 100
120 Play Y
130 If opponent plays X go to 100 else go to 120
```

Represents a strategy that does whatever its opponent did last, and begins by playing X, known as "Tit-For-Tat." (Axelrod 1982). Such machines can readily be encoded as bit-strings. Consider the encoding "1 0 1 0 0 1" to represent TFT. The first three bits, "1 0 1" are state 0. The first bit, "1" is interpreted as "Play X." The second bit, "0" is interpreted as "if opponent plays X go to state 1," the third bit, "1", is interpreted as "if the opponent plays Y, go to state 1." State 1 has a similar interpretation. Crossing over such bit-strings always yields valid strategies.

SIMULATIONs in the Prisoner's dilemma have been undertaken (Axelrod 1987, Fogel 1993, Miller 1989) of these machines.

Alternative representations of game players include **CLASSIFIER SYSTEMs** (Marimon, McGrattan and Sargent 1990, [GOLD89]), and Neural-networks (Fogel and Harrald 1994), though not necessarily with a GA. (Fogel 1993), and Fogel and Harrald 1994 use an Evolutionary Program).

Other methods of evolving a population can be found in Lindgren 1991, Glance and Huberman 1993 and elsewhere.

A GA for playing the game "Mastermind" has been produced. See **http://kal-el.ugr.es/mastermind**

References.

Axelrod, R. (1987) "The Evolution of Strategies in the Repeated Prisoner's Dilemma," in [DAVIS91]

Axelrod, R (?) "The Complexity of Cooperation" (See the web site, which includes code to implement tournaments: **http://pscs.physics.lsa.umich.edu/Software/ComplexCoop.html**)

Miller, J.H. (1989) "The Coevolution of Automata in the Repeated Prisoner's Dilemma" Santa Fe Institute Working Paper 89-003.

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.

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

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.

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

Fogel (1993) "Evolving Behavior 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.

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

Stanley, E.A., Ashlock, D. and Tesfatsion, L. (1994) "Iterated Prisoners Dilemma with Choice and Refusal of Partners in [ALIFEIII] 131-178

JOB-SHOP SCHEDULING

The Job-Shop Scheduling Problem (JSSP) 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 JSSP, (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 encodings. 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 JSSP 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)."

Other scheduling problems

In contrast to job shop scheduling some maintenance scheduling problems consider which activities to schedule within a planned maintenance period, rather than seeking to minimise 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).

References

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

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.

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

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.

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

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

MANAGEMENT SCIENCES

"Applications of **EA** in management science and closely related fields like organizational ecology is a domain that has been covered by some EA researchers - with considerable bias towards scheduling problems. Since I believe that EA have considerable potential for applications outside the rather narrow domain of scheduling and related combinatorial problems, I started collecting references about the status quo of EA-applications in management science. This report intends to make available my findings to other researchers in the field. It is a short overview and lists some 230 references to current as well as finished research projects. [..]

"At the end of the paper, a questionnaire has been incorporated that may be used for this purpose. Other comments are also appreciated."

——— from the Introduction of (Nissen 93)

References

Nissen, V. (1993) "Evolutionary Algorithms in Management Science: An Overview and List of References", Papers on Economics and Evolution, edited by the European Study Group for Evolutionary Economics. This report is also avail. via anon. **FTP** from **ftp.gwdg.de/pub/msdos/reports/wi/earef.eps**

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

NON-LINEAR FILTERING

New connections between **GENETIC ALGORITHMs** and Non Linear Filtering Theory have been established. **GAs** have already been successfully applied to a large class of non-linear filtering problems such as RADAR / SONAR/ **GPS** signal processing. This relatively new branch of GA application has also lead to new results on the convergence of GAs: large deviations, fluctuations...

Some preprints and references on this topic are available in the web page: **http://www-sv.cict.fr/lsp/Delmoral/index.html**

The new results also points out some natural connections between: genetic type algorithms, information theory, non-linear filtering theory, interacting and branching particle systems.

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 Tabu 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 (eg) 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, hybridisation 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.

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 Information Technology, C.S.I.R.O., c/o Dept. of Communication & Electronic Engineering, Royal Melbourne Institute of Technology, PO BOX 2476V, Melbourne 3001, Australia)

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

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

Colorni A., M. Dorigo & V. Maniezzo (1990). A Genetic Algorithm to Solve the Timetable Problem. Technical Report No. 90-060, Politecnico di Milano, Italy. **http://iridia.ulb.ac.be/dorigo/dorigo/tec.reps/TR.01-TTP.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.

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

Q3: Who is concerned with EAs?

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" [HILLIS92], 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 modelled more easily and esp. "faster" on a parallel machine, than on a serial "PC" one.

Biologists

Are finding **EAs** useful when it comes to protein folding and other such biocomputational problems (see Q2).

EAs can also be used to model the behaviour of *real* **POPULATIONs** of organisms. Some biologists are hostile to modeling, but an entire community of *Population Biologists* arose with the 'evolutionary synthesis' of the 1930's 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. Maynard Smith's text *Evolutionary Genetics.* Richard Dawkin'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...)

Chemists

And in particular biochemists and molecular chemists, are interested in problems such as the conformational analysis of molecular clusters and related problems in molecular sciences. The application of **GAs** to molecular systems has opened an interesting area of research and the number of chemists involved in it increases day-by-day.

Some typical research topics include:

- protein folding; • conformational analysis and energy minimization; • docking algorithms for drug-design; • solvent site prediction in macromolecules;

Several papers have been published in journals such as *Journal of Computational Chemistry* and *Journal of Computer-Aided Design.*

Some interesting WWW sites related to the applications of GAs to chemistry (or molecular science in general) include:

- **http://isl.msu.edu/GA/projects/biochem/biochem.html** about GAs in biochemistry (water site prediction, drug-design and protein folding); • **http://www.tc.cornell.edu/Edu/SPUR/SPUR94/Main/John.html** about the application of GAs to the search of conformational energy minima; • **http://cmp.ameslab.gov/cmp/CMP_Theory/gsa/gen2.html** By using a GA in combiation with a Tight-binding model, David Deaven and Kai-Ming Ho founded fullerene cages (including C60) starting from random coordinates.

See also Q2 for applications in biocomputing.

Philosophers

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

Q4: How many EAs exist? Which?

The All Stars

There are currently 3 main paradigms in **EA** research: **GENETIC ALGORITHMs, EVOLUTIONARY PROGRAMMING,** and **EVOLUTION STRATEGIEs. CLASSIFIER SYSTEMs** and **GENETIC PROGRAMMING** are **OFFSPRING** of the **GA** community. Besides this leading crop, there are numerous other different approaches, alongside hybrid experiments, i.e. there exist pieces of software residing in some researchers computers, that have been described in papers in conference proceedings, and may someday prove useful on certain tasks. To stay in EA slang, we should think of these evolving strands as

BUILDING BLOCKs, that when recombined someday, will produce new offspring and give birth to new EA paradigm(s).

Promising Rookies

As far as "solving complex function and **COMBINATORIAL OPTIMIZATION** tasks" is concerned, Davis' work on real-valued representations and adaptive operators should be mentioned (Davis 89). Moreover Whitley's Genitor system incorporating ranking and "steady state" mechanism (Whitley 89), Goldberg's "messy GAs", involves adaptive representations (Goldberg 91), and Eshelman's CHC algorithm (Eshelman 91). For real **FUNCTION OPTIMIZATION,** Differential **EVOLUTION** seems hard to beat in terms of convergence speed as well as simplicity: With just three control variables, tuning is particularly easy to do.

For "the design of robust learning systems", i.e. the field characterized by **CFS,** Holland's (1986) **CLASSIFIER SYSTEM,** with its state-of-the-art implementation CFS-C (Riolo 88), we should note recent developments in SAMUEL (Grefenstette 89), GABIL (De Jong & Spears 91), and GIL (Janikow 91).

References

Davis, L. (1989) "Adapting operator probabilities in genetic algorithms", [ICGA89], 60-69.

De Jong K.A. & Spears W. (1991) "Learning concept classification rules using genetic algorithms". Proc. 12th IJCAI, 651-656, Sydney, Australia: Morgan Kaufmann.

Dorigo M. & E. Sirtori (1991)."ALECSYS: A Parallel Laboratory for Learning Classifier Systems". Proceedings of the Fourth International Conference on Genetic Algorithms, San Diego, California, R.K.Belew and L.B.Booker (Eds.), Morgan Kaufmann, 296-302.

Dorigo M. (1995). "ALECSYS and the AutonoMouse: Learning to Control a Real Robot by Distributed Classifier Systems". Machine Learning, 19, 3, 209-240.

Eshelman, L.J. et al. (1991) "Preventing premature convergence in genetic algorithms by preventing incest", [ICGA91], 115-122.

Goldberg, D. et al. (1991) "Don't worry, be messy", [ICGA91], 24-30.

Grefenstette, J.J. (1989) "A system for learning control strategies with genetic algorithms", [ICGA89], 183-190.

Holland, J.H. (1986) "Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems". In R. Michalski, J. Carbonell, T. Mitchell (eds), Machine Learning: An Artificial Intelligence Approach. Los Altos: Morgan Kaufmann.

Janikow C. (1991) "Inductive learning of decision rules from attribute-based examples: A knowledge-intensive Genetic Algorithm approach". TR91-030, The University of North Carolina at Chapel Hill, Dept. of Computer Science, Chapel Hill, NC.

Riolo, R.L. (1988) "CFS-C: A package of domain independent subroutines for implementing classifier systems in arbitrary, user-defined environments". Logic of computers group, Division of computer science and engineering, University of Michigan.

Whitley, D. et al. (1989) "The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best", [ICGA89], 116-121.

Q4.1: What about Alife systems, like Tierra and VENUS?

None of these are **EVOLUTIONARY ALGORITHMs,** but all of them use the evolutionary metaphor as their "playing field".

Tierra

Synthetic organisms have been created based on a computer metaphor of organic life in which CPU time is the ‘‘energy’’ resource and memory is the ‘‘material’’ resource. Memory is organized into informational patterns that exploit CPU time for self-replication. **MUTATION** generates new forms, and **EVOLUTION** proceeds by natural **SELECTION** as different **GENOTYPEs** compete for CPU time and memory space.

Observation of nature shows that evolution by natural selection is capable of both **OPTIMIZATION** and creativity. Artificial models of evolution have demonstrated the optimizing ability of evolution, as exemplified by the field of **GENETIC ALGORITHMs.** The creative aspects of evolution have been more elusive to model. The difficulty derives in part from a tendency of models to specify the meaning of the ‘‘genome’’ of the evolving entities, precluding new meanings from emerging. I will present a natural model of evolution demonstrating both optimization and creativity, in which the **GENOME** consists of sequences of executable machine code.

From a single rudimentary ancestral ‘‘creature’’, very quickly there evolve parasites, which are not able to replicate in isolation because they lack a large portion of the genome. However, these parasites search for the missing information, and if they locate it in a nearby creature, parasitize the information from the neighboring genome, thereby effecting their own replication.

In some runs, hosts evolve immunity to attack by parasites. When immune hosts appear, they often increase in frequency, devastating the parasite **POPULATIONs.** In some runs where the community comes to be dominated by immune hosts, parasites evolve that are resistant to immunity.

Hosts sometimes evolve a response to parasites that goes beyond immunity, to actual (facultative) hyper-parasitism. The hyper-parasite deceives the parasite causing the parasite to devote its energetic resources to replication of the hyper-parastie genome. This drives the parasites to extinction. Evolving in the absence of parasites, hyper-parasites completely dominate the community, resulting in a relatively uniform community characterized by a high degree of relationship between **INDIVIDUALs.** Under these circumstances, sociality evolves, in the form of creatures which can only replicate in aggregations.

The cooperative behavior of the social hyper-parasites makes them vulnerable to a new class of parasites. These cheaters, hyper-hyper-parasites, insert themselves between cooperating social individuals, deceiving the social creatures, causing them to replicate the genomes of the cheaters.

The only genetic change imposed on the simulator is random bit flips in the machine code of the creatures. However, it turns out that parasites are very sloppy replicators. They cause significant **RECOMBINATION** and rearrangement of the genomes. This spontaneous sexuality is a powerful force for evolutionary change in the system.

One of the most interesting aspects of this instance of life is that the bulk of the evolution is based on adaptation to the biotic **ENVIRONMENT** rather than the physical environment. It is co-evolution that drives the system.

−−− "Tierra announcement" by Tom Ray (1991)

How to get Tierra?

The complete source code and documentation (but not executables) is available by anonymous **FTP** at: **tierra.slhs.udel.edu:/** and **life.slhs.udel.edu/** in the directories: almond/, beagle/, doc/, and tierra/.

If you do not have FTP access you may obtain everything on DOS disks. For details, write to: Virtual Life, 25631 Jorgensen Rd., Newman, CA 95360.

References

Ray, T. S. (1991) "Is it alive, or is it GA?" in [ICGA91], 527--534.

Ray, T. S. (1991) "An approach to the synthesis of life." in [ALIFEII], 371--408.

Ray, T. S. (1991) "Population dynamics of digital organisms." in [ALIFEII].

Ray, T. S. (1991) "Evolution and optimization of digital organisms." Scientific Excellence in Supercomputing: The IBM 1990 Contest Prize Papers, Eds. Keith R. Billingsley, Ed Derohanes, Hilton Brown, III. Athens, **GA,** 30602, The Baldwin Press, The University of Georgia.

Ray, T. S. (1992) "Evolution, ecology and optimization of digital organisms." Santa Fe Institute working paper 92-08-042.

Ray, T. S. "Evolution, complexity, entropy, and artificial reality." submitted Physica D. Avail. as **tierra.slhs.udel.edu/doc/PhysicaD.tex**

Ray, T. S. (1993) "An evolutionary approach to synthetic biology, Zen and the art of creating life. Artificial Life 1(1). Avail. as **tierra.slhs.udel.edu/doc/Zen.tex**

VENUS

Steen Rasmussen's (et al.) VENUS I+II "coreworlds" as described in [ALIFEII] and [LEVY92], are inspired by A.K. Dewdney's well-known article (Dewdney 1984). Dewdney proposed a game called "Core Wars", in which hackers create computer programs that battle for control of a computer's "core" memory (Strack 93). Since computer programs are just patterns of information, a successful program in core wars is one that replicates its pattern within the memory, so that eventually most of the memory contains its pattern rather than that of the competing program.

VENUS is a modification of Core Wars in which the Computer programs can mutate, thus the pseudo assembler code creatures of VENUS evolve steadily. Furthermore each memory location is endowed with "resources" which, like sunshine are added at a steady state. A program must have sufficient resources in the regions of memory it occupies in order to execute. The input of resources determines whether the VENUS ecosystem is a "jungle" or a "desert." In jungle **ENVIRONMENTs,** Rasmussen et al. observe the spontaneous emergence of primitive "copy/split" organisms starting from (structured) random initial conditions.

––– [ALIFEII], p.821

Dewdney, A.K. (1984) "Computer Recreations: In the Game called Core War Hostile Programs Engage in a Battle of Bits", Sci. Amer. 250(5), 14-22.

Farmer & Belin (1992) "Artificial Life: The Coming Evolution", [ALIFEII], 815-840.

Rasmussen, et al. (1990) "The Coreworld: Emergence and Evolution of Cooperative Structures in a Computational Chemistry", [FORREST90], 111-134.

Rasmussen, et al. (1992) "Dynamics of Programmable Matter", [ALIFEII], 211-254.

Strack (1993) "Core War Frequently Asked Questions (**rec.games.corewar** FAQ)" Avail. by anon. **FTP** from **rtfm.mit.edu/pub/usenet/news.answers/games/corewar-faq.Z**

PolyWorld

Larry Yaeger's PolyWorld as described in [ALIFEIII] and [LEVY92] is available via anonymous **FTP** from **ftp.apple.com/pub/larryy/polyworld/**

"The subdirectories in this "polyworld" area contain the source code for the PolyWorld ecological simulator, designed and written by Larry Yaeger, and Copyright 1990, 1991, 1992 by Apple Computer.

PostScript versions of my **ARTIFICIAL LIFE** III technical paper have now been added to the directory. These should be directly printable from most machines. Because some unix systems' "lpr" commands cannot handle very large files (ours at least), I have split the paper into Yaeger.ALife3.1.ps and Yaeger.ALife3.2.ps. These files can be ftp-ed in "ascii" mode. For unix users I have also included compressed versions of both these files (indicated by the .Z suffix), but have left the uncompressed versions around for people connecting from non-unix systems. I have not generated PostScript versions of the images, because they are color and the resulting files are much too large to store, retrieve, or print. Accordingly, though I have removed a Word-formatted version of the textual body of the paper that used to be here, I have left a Word-formatted version of the color images. If you wish to acquire it, you will need to use the binary transfer mode to move it to first your unix host and then to a Macintosh (unless Word on a PC can read it - I don't know), and you may need to do something nasty like use ResEdit to set the file type and creator to match those of a standard Word document (Type = WDBN, Creator = MSWD). [..]"

−−− from the README by Larry Yaeger <larryy@apple.com>

General Alife repositories?

Also, all of the following **FTP** sites carry **ALIFE** related info:

ftp.cognet.ucla.edu/pub/alife/ , **life.anu.edu.au/pub/complex_systems/alife/** , **ftp.cogs.susx.ac.uk/pub/reports/csrp/** , **xyz.lanl.gov/nlin-sys/** , **alife.santafe.edu/pub/** .

Q5: What about all this Optimization stuff?

Just think of an **OPTIMIZATION** problem as a black box. A large black box. As large as, for example, a Coca-Cola vending machine. Now, we don't know anything about the inner workings of this box, but see, that there are some regulators to play with, and of course we know, that we want to have a bottle of the real thing...

Putting this everyday problem into a mathematical model, we proceed as follows:

(1) we label all the regulators with *x* and a number starting from 1; the result is a vector *x*, i.e. *(x_1,...,x_n)*, where *n* is the number of visible regulators.

(2) we must find an objective function, in this case it's obvious, we want to get *k* bottles of the real thing, where *k* is equal to 1. [some might want a "greater or equal" here, but we restricted ourselves to the visible regulators (we all know that sometimes a "kick in the right place" gets use more than 1, but I have no idea how to put this mathematically...)]

(3) thus, in the language some mathematicians prefer to speak in: $f(x) = k = 1$. So, what we have here is a maximization problem presented in a form we know from some boring calculus lessons, and we also know that there at least a dozen utterly uninteresting techniques to solve problems presented this way...

What can we do in order to solve this problem?

We can either try to gain more knowledge or exploit what we already know about the interior of the black box. If the objective function turns out to be smooth and differentiable, analytical methods will produce the exact solution.

If this turns out to be impossible, we might resort to the brute force method of enumerating the entire **SEARCH SPACE.** But with the number of possibilities growing exponentially in *n*, the number of dimensions (inputs), this method becomes infeasible even for low-dimensional spaces.

Consequently, mathematicians have developed theories for certain kinds of problems leading to specialized **OPTIMIZATION** procedures. These algorithms perform well if the black box fulfils their respective prerequisites. For example, Dantzig's *simplex algorithm* (Dantzig 66) probably represents the best known multidimensional method capable of efficiently finding the global optimum of a linear, hence convex, objective function in a search space limited by linear constraints. (A **USENET FAQ** on linear programming is maintained by *John W. Gregory* of Cray Research, Inc. Try to get your hands on "linear-programming-faq" (and "nonlinear-programming-faq") that is posted monthly to **sci.op-research** and is mostly interesting to read.)

Gradient strategies are no longer tied to these linear worlds, but they smooth their world by exploiting the objective function's first partial derivatives one has to supply in advance. Therefore, these algorithms rely on a locally linear internal model of the black box.

Newton strategies additionally require the second partial derivatives, thus building a quadratic internal model. *Quasi-Newton*, conjugate gradient and variable metric strategies approximate this information during the search.

The deterministic strategies mentioned so far cannot cope with deteriorations, so the search will stop if anticipated improvements no longer occur. In a multimodal **ENVIRONMENT** these algorithms move "uphill" from their respective starting points. Hence, they can only converge to the next local optimum.

Newton-Raphson-methods might even diverge if a discrepancy between their internal assumptions and reality occurs. But of course, these methods turn out to be superior if a given task matches their requirements. Not relying on derivatives, *polyeder strategy, pattern search* and *rotating coordinate search* should also be mentioned here because they represent robust non-linear optimization algorithms (Schwefel 81).

Dealing with technical optimization problems, one will rarely be able to write down the objective function in a closed form. We often need a **SIMULATION** model in order to grasp reality. In general, one cannot even expect these models to behave smoothly. Consequently, derivatives do not exist. That is why optimization algorithms that can successfully deal with black box-type situations have been developed. The increasing applicability is of course paid for by a loss of "convergence velocity," compared to algorithms specially designed for the given problem. Furthermore, the *guarantee to find the global optimum no longer exists!*

But why turn to nature when looking for more powerful algorithms?

In the attempt to create tools for various purposes, mankind has copied, more often instinctively than geniously, solutions invented by nature. Nowadays, one can prove in some cases that certain forms or structures are not only well adapted to their **ENVIRONMENT** but have even reached the optimum (Rosen 67). This is due to the fact that the laws of nature have remained stable during the last 3.5 billion years. For instance, at branching points the measured ratio of the diameters in a system of blood-vessels comes close to the theoretical optimum provided by the laws of fluid dynamics (2^-1/3). This, of course, only represents a limited, engineering point of view on nature. In general, nature performs *adaptation,* not *optimization.*

The idea to imitate basic principles of natural processes for optimum seeking procedures emerged more than three decades ago (cf Q10.3). Although these algorithms have proven to be robust and direct **OPTIMIZATION** tools, it is only in the last five years that they have caught the researchers' attention. This is due to the fact that many people still look at organic **EVOLUTION** as a giantsized game of dice, thus ignoring the fact that this model of evolution cannot have worked: a human germ-cell comprises approximately 50,000 **GENEs,** each of which consists of about 300 triplets of nucleic bases. Although the four existing bases only encode 20 different amino acids, 20^15,000,000, ie circa 10^19,500,000 different **GENOTYPEs** had to be tested in only circa 10^17 seconds, the age of our planet. So, simply rolling the dice could not have produced the diversity of today's complex living systems.

Accordingly, taking random samples from the high-dimensional parameter space of an objective function in order to hit the global optimum must fail (*Monte-Carlo search*). But by looking at organic evolution as a cumulative, highly parallel sieving process, the results of which pass on slightly modified into the next sieve, the amazing diversity and efficiency on earth no longer appears miraculous. When building a model, the point is to isolate the main mechanisms which have led to today's world and which have been subjected to evolution themselves. Inevitably, nature has come up with a mechanism allowing **INDIVIDUALs** of one **SPECIES** to exchange parts of their genetic information **(RECOMBINATION** or **CROSSOVER),** thus being able to meet changing environmental conditions in a better way.

Dantzig, G.B. (1966) "Lineare Programmierung und Erweiterungen", Berlin: Springer. (Linear programming and extensions)

Kursawe, F. (1994) " Evolution strategies: Simple models of natural processes?", Revue Internationale de Systémique, France (to appear).

Rosen, R. (1967) "Optimality Principles in Biologie", London: Butterworth.

Schwefel, H.-P. (1981) "Numerical Optimization of Computer Models", Chichester: Wiley.

Q10: What introductory material on EAs is there?

There are many sources of introductory material on evolutionary algorithms: background books (see Q10.1), textbooks (see Q10.2), classical works (see Q10.3), journal articles (see Q10.4), technical reports (see Q10.5), more advanced literature (see Q10.6), biological background reading (see Q10.7), bibliography collections (see Q10.8), videos (see Q10.9) and CD-ROMs (Q10.10). Information on how to get dissertations is also given below (see Q10.11).

Conference proceedings (see Q12) are also a good source of up-to-date (and sometimes introductory) material.

Q10.1: Suitable background reading for beginners?

These books give a "flavor" of what the subject is about.

Dawkins, R. (1976, 1989 2nd ed) "The Selfish Gene", Oxford: Oxford University Press. [The 2nd edition includes two new chapters]

Dawkins, R. (1982) "The Extended Phenotype: The Gene as a Unit of Selection", Oxford: Oxford University Press.

Dawkins, R. (1986) "The Blind Watchmaker", New York: W.W. Norton.

Fogel, D. (1998) "Evolutionary Computation: The Fossil Record," IEEE Press. Chronicles the history of simulated evolution from the early 1950s. **http://www.natural-selection.com/people/dbf.html**

Gonick, L. (1983) "The Cartoon Guide to Computer Science", New York: Barnes & Noble. [eds note: features an interesting chapter on Charles Babbage in conjunction with "horse racing forecasting", if you want to use EAs to fullfill this task, better read this section first]

Gonick, L. (1983) "The Cartoon Guide to Genetics", New York: Barnes & Noble.

Regis, E. (1987) "Who got Einstein's Office? Eccentricity and Genius at the Institute for Advanced Study", Reading, MA: Addison Wesley [eds note: chapters 5, 10 and 12]

Levy, S. (1992) "Artificial Life: The Quest for a new Creation", New York, NY: Pantheon. [LEVY92]: [eds note: read this and you will have the *urge* to work in this field]

Sigmund, K. (1993) "Games of Life: Explorations in Ecology, Evolution and Behaviour", Oxford: Univ. Press. 252 pp. Hard/Softcover avail.

Q10.2: Textbooks on EC?

These books go into the "nuts and bolts" of EC.

Goldberg, D.E. (1989) "Genetic Algorithms in Search, Optimization, and Machine Learning",Addison-Wesley. [GOLD89]: (Probably the most widely referenced book in the field!)

Davis, L. (ed) (1991) "Handbook of Genetic Algorithms", Van Nostrand Reinhold, New York, NY. [DAVIS91]:

Michalewicz, Z. (1992) Genetic algorithms + Data Structures = Evolution Programs", Springer-Verlag, New York, NY. [MICHALE92]: Also second, extended edition (1994) with index. [MICHALE94]:

Koza, J.R. (1992), Genetic Programming: On the Programming of Computers by means of Natural Selection", Cambridge, MA: MIT Press. [KOZA92]:

Langdon, W.B. (1998), Genetic Programming and Data Structures Hingham, MA: Kluwer. [LANG98]: **http://www.wkap.nl/book.htm/0-7923-8135-1**

Q10.3: The Classics?

Mostly older works which have helped to shape the field.

Charles Darwin (1859), "The Origin of Species", London: John Murray. (Penguin Classics, London, 1985; New American Library, Mentor Paperback)

Box, G.E.P. (1957) "Evolutionary operation: a method of increasing industrial productivity", Applied Statistics, 6, 81-101.

Fraser, A.S. (1957) "Simulation of genetic systems by automatic digital computers", Australian Journal of Biological Sciences, 10, 484-491.

Friedman, G.J. (1959) "Digital simulation of an evolutionary process", General Systems Yearbook, 4:171-184.

Bremermann, H.J. (1962) "Optimization through evolution and recombination". In M.C. Yovits, et al, (eds) Self-Organizing Systems. Washington, DC: Spartan Books.

Holland, J.H. (1962) "Outline for a logical theory of adaptive systems", JACM, 3, 297-314.

Samuel, A.L. (1963) "Some Studies in Machine Learning using the Game of Checkers", in Computers and Thought, E.A. Feigenbaum and J. Feldman (eds), New York: McGraw-Hill.

Walter, W.G. (1963) "The Living Brain", New York: W.W. Norton.

Fogel, L.J., Owens, A.J. & Walsh, M.J. (1966) "Artificial Intelligence through Simulated Evolution", New York: Wiley. [Fogel66]:

Rosen, R. (1967) "Optimality Principles in Biology", London: Butterworths.

Rechenberg, I. (1973, 1993 2nd edn) "Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution", Stuttgart: Fromman-Holzboog. (Evolution Strategy: Optimization of technical systems by means of biological evolution)

Holland, J.H. (1975) "Adaptation in natural and artificial systems", Ann Arbor, MI: The University of Michigan Press. [HOLLAND75]: 2nd edn. (1992) [HOLLAND92]:

De Jong, K.A. (1975) "An analysis of the behavior of a class of genetic adaptive systems", Doctoral thesis, Dept. of Computer and Communication Sciences, University of Michigan, Ann Arbor.

Schwefel, H.-P. (1977) "Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie", Basel: Birkhäuser.

Schwefel, H.-P. (1981) "Numerical Optimization of Computer Models", Chichester: Wiley. [eds note: English translation of the previous entry; a reworked edition is currently in preparation for 1994]

Axelrod, R. (1984) "The evolution of cooperation", NY: Basic Books.

Cramer, N.L. (1985) "A Representation for the Adaptive Generation of Simple Sequential Programs" [ICGA85], 183-187.

Bäck, T., Hoffmeister, F. & Schwefel, H.-P. (1991) "A Survey of Evolution Strategies" [ICGA91], 2-9.

Q10.4: Introductory Journal Articles?

Bäck, T. & Schwefel, H.-P. (1993) "An Overview of Evolutionary Algorithms for Parameter Optimization", Evolutionary Computation, 1(1), 1-23.

Bäck, T., Rudolph, G. & Schwefel, H.-P. (1993) "Evolutionary Programming and Evolution Strategies: Similarities and Differences", [EP93], 11-22.

Bäck, T., Hammel, U. and Schwefel, H.-P. (1997) "Evolutionary computation: Comments on the history and current state," IEEE Trans. Evolutionary Computation, Vol. 1:1, pp. 3-17

Beasley, D., Bull, D.R., & Martin, R.R. (1993) "An Overview of Genetic Algortihms: Part 1, Fundamentals", University Computing, 15(2) 58-69. Available by ftp from **ENCORE** (See Q15.3) in file: **GA/papers/over93.ps.gz** or from **ralph.cs.cf.ac.uk/pub/papers/GAs/ga_overview1.ps**

Beasley, D., Bull, D.R., & Martin, R.R. (1993) "An Overview of Genetic Algortihms: Part 2, Research Topics", University Computing, 15(4) 170-181. Available by ftp from **ENCORE** (See Q15.3) in file: **GA/papers/over93-2.ps.gz** or from **ralph.cs.cf.ac.uk/pub/papers/GAs/ga_overview2.ps**

Brooks, R.A. (1991) "Intelligence without Reason", MIT AI Memo No. 1293. Appeared in "Computer's and Thought", IJCAI-91.

Dawkins, R. (1987) "The Evolution of Evolvability", [ALIFEI], 201-220.

Fogel, D.B. (1994) "An introduction to simulated evolutionary optimization," IEEE Trans. Neural Networks, Vol. 5:1, pp. 3-14.

Goldberg, D.E. (1986) "The Genetic Algorithm: Who, How, and What Next?". In Kumpati S. Narenda, ed., Adaptive and Learning Systems, Plenum, New York, NY.

Goldberg, D. (1994), "Genetic and Evolutionary Algorithms Come of Age", Communications of the ACM, 37(3), 113--119.

Hillis, W.D. (1987) "The Connection Machine", Scientific American, 255(6).

Hillis, W.D. (1992) "Massively Parallel Computing" Daedalus, winter, 121(1), 1-29. [HILLIS92]:

Holland, J.H. (1989) "Using Classifier Systems to Study Adaptive Nonlinear Networks". In: Lectures in the Science of Complexity, SFI Studies in the Science of Complexity, D. Stein, (ed), Addison Wesley.

Holland, J.H. (1992) "Genetic Algorithms", Scientific American, 267(1), 66-72.

Holland, J.H. (1992) "Complex Adaptive Systems" Daedalus, winter, 121(1), 17-30.

Mitchell, M. & Forrest S. (1993) "Genetic Algorithms and Artificial Life", Artificial Life, 1(1). Also avail. as SFI Working Paper 31-11-072.

Sims, K. (1991) "Artificial Evolution for Computer Graphics", Computer Graphics, 25(4), 319-328

Sipper, M (1996) "A Brief Introduction to Genetic Algorithms", unpublished guide, available from **http://lslwww.epfl.ch/˜moshes/ga.html**

Spears, W.M., DeJong, K.A., Bäck, T., Fogel, D. & de Garis, H. (1993) "An Overview of Evolutionary Computation", [ECML93], 442-459.

Peter Wayner (1991), "Genetic Algorithms: Programming takes a valuable tip from nature", BYTE, January, 361--368.

Q10.5: Introductory Technical Reports?

See also Q14 for other technical

Ficek, Rhona (1990) "Genetic Algorithms", Dept. of Computer Science and Operations Research, North Dakota State University. An introductory report, available from: **http://www.atm.cs.ndsu.nodak.edu/Dienst/UI/2.0/Describe/ncstrl.ndsu_cs%2fNDSU-CS-TR-90-51**

Hoffmeister, F. & Bäck, T. (1990, 1992) "Genetic Algorithms and Evolution Strategies: Similarities and Differences", University of Dortmund, Dept. of CS, SyS-1/92. Available by ftp from **lumpi.informatik.uni-dortmund.de:**

Serrada, Anselmo Perez (1996) "Una introducci'on a la Computaci'on Evolutiva". An introduction to EC in Spanish. Available from **ENCORE** (see Q15.3) in file **EA/papers/intro-spanish.ps.gz** with an overview in **EA/papers/intro-spanish.leeme** .

Whitley, D. (1993) "A Genetic Algorithm Tutorial", Colorado State University, Dept. of CS, TR CS-93-103. Available by ftp from **ftp.cs.colostate.edu/pub/public_html/TechReports/1993/tr-103.ps.Z** or from **http://www.cs.colostate.edu**
- follow the link to Technical Reports.

Q10.6: Not-quite-so-introductory Literature?

Bock, P. (1993) "The Emergence of Artificial Cognition: An Introduction to Collective Learning", Singapore: World Scientific.

Davis, L. (ed) (1987) "Genetic Algorithms and Simulated Annealing", available from Morgan Kaufmann Publishers Inc., 340 Pine St, San Francisco, CA 94104, (415-392-2665).

Davidor, Y. (1991) "Genetic Algorithms and Robotics", Singapore: World Scientific. ISBN 9-810202172.

Forrest, S. (ed) (1990) "Emergent Computation. Self-Organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Networks", [FORREST90]:, Cambridge, MA: MIT Press. (Special issue of Physica D.)

Hillis, W.D. (1990) "Co-Evolving Parasites Improve Simulated Evolution as an Optimization procedure", [ALIFEII], 313-324.

Holland, J.H., Holyoak, K.J., Nisbett, R.E. & Thagard, P.R. (1986) "Induction: Processes of Inference, Learning, and Discovery", Cambridge, MA: MIT Press.

Holland, J.H. (1992) "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, Cambridge, MA: MIT Press/Bradford Books, (2nd edn). Hard: ISBN 0-262-08213-6. Soft: ISBN 0-262-58111-6.

Serra, R. & Zanarini, G. (1990) "Complex Systems and Cognitive Processes", New York, NY: Springer-Verlag.

Stender, J. (ed.). (1993) "Parallel Genetic Algorithms", IOS Publishing. [Cites just about everything in the parallel GA field. –– John Koza]

Rujan, P. (1988) "Searching for optimal configurations by simulated tunneling", Zeitschrift der Physik B", Vol.73, 391-416.

Rudolph, G. (1994) "Convergence Analysis of Canonical Genetic Algorithms", IEEE Trans. on Neural Networks, Special issue on EP. Available by ftp from **ENCORE** (See Q15.3) in file: **GA/papers/canon94.ps.gz**

Fogel, D. (1995), "Evolutionary Computation: Toward a New Philosophy of Machine Intelligence", Piscataway, NJ: IEEE Press. ISBN 0-7803-1048-0.

Schwefel, H-P. (1995) "Evolution and Optimum Seeking", New York: Wiley. ISBN 0-471-57148-2

Q10.7: Biological Background Readings?

Adams, D. with Carwardine M. (1990) "Last Chance to see...", London: Heinemann. [David Corne: I strongly suggest you read this. Its a report on visits to various parts of the world to see endangered species. It is remarkably and wonderfully funny and illuminating. It would actually be a good reference to have in any bit of the FAQ to do with genetic diversity and/or the lack of it, or the remarkable kinds of adaptations that can occur for the strangest reasons.]

Cairns-Smith, A.G. (1985) "Seven Clues to the Origin of Life", Cambridge: Cambridge Univ. Press.

Fisher, R.A. (1958) "The Genetic Theory of Natural Selection", New York: Dover.

Futuyma, D.J. (1986) "Evolutionary Biology", Sunderland, MA: Sinauer Assoc. [eds note: the bibliography of this book is truly a treasure chest]

Lewin, B. (1993) "Genes IV".

Lewontin, R.C. (1974) "The Genetic Basis of Evolutionary Change", New York: Columbia Univ. Press.

Maynard Smith, J. (1972) "On Evolution", Edinburgh: Edinburgh Univ. Press.

Maynard Smith, J. (1978) "Optimization Theory in Evolution", Annual Review of Ecology and Systematics 9:31-56.

Maynard Smith, J. (1982) "Evolution and the Theory of Games", Cambridge: Cambridge Univ. Press.

Maynard Smith, J. (1989) "The Problems of Biology", Oxford: Oxford Univ. Press.

Maynard Smith, J. (1989) "Evolutionary Genetics", Oxford: Oxford Univ. Press.

Mayr, E. (1963) "Animal Species and Evolution", Cambridge, MA: Harvard Univ. Press.

Mayr, E. (1982) "The Groth of Biological Thought", Cambridge, MA: The Belknap Press of Harvard Univ. Press.

Ridley, M. (1985) "The Problems of Evolution", Oxford: Oxford Univ. Press.

Tort, P. Ed. (1996) "Dictionary of Darwinism and of Evolution", Paris, France: Presses Universitaires de France. Produced by a team of 150 international experts over a period of 10 years. Contains a vast amount of information about what Darwinism is and (perhaps more importantly) is *not.* Further information from **http://www.planete.net/˜ptort/darwin/evolengl.html** (in various languages).

Watson, J.D. (1966) "Molecular Biology of the Gene", Menlo Park: Benjamin.

Watson, J.D., Hopkins, N.H., Roberts, J.W., Steitz, J.A. & Weiner, A.M. (1987) "Molecular Biology of the Gene (4th edn)", Menlo Park: Benjamin.

Williams, G.C. (1966) "Adaptation and Natural Selection", Princeton, NJ: Princeton Univ. Press.

Wright, S. (1932) "The roles of mutation, inbreeding, crossbreeding and selection in evolution", in: Proc. of the 6th Int'l Congress on Genetics I, 356.

There is a *lot* of interesting material on biology and evolution in the **talk.origins** newsgroup repository, available by FTP. The index of files, available from **ics.uci.edu/pub/origins/Index** , lists what's there, and includes files on Darwinism, definition of evolution, introduction to evolutionary biology, a list of important FAQ files, speciation, and genetic drift.

Q10.8: On-line bibliography collections?

The Big One

Jarmo Alander has compiled probably the biggest EC bibliography around. It has 2500 entries, and is available in postscript form by ftp from: **garbo.uwasa.fi/pc/research/2500GArefs.ps.gz** and also from **ENCORE** (see Q15.3) in file **refs/2500GArefs.ps.gz** Please send any additions or corrections to <ja@cs.hut.fi>

The same directory on ENCORE also contains some other bibliography collections.

Combinations of GAs and NNs

Dave Schaffer <ds1@philabs.Philips.Com> has compiled a bibliograpy on combinations of **GAs** and neural networks. About 150 entries, available in Bib format from **ENCORE** (See Q15.3) in file **refs/cogann.bib.gz**

Jochen Ruhland <jochenr@neuro.informatik.uni-kassel.de> has also compiled a bibliography on this topic. Some papers deal only with neural networks, some only with genetic algorithms. About 300 references altogether. Some include an abstract. Available from: **ftp.neuro.informatik.uni-kassel.de/pub/NeuralNets/** in **We_and_our_work/papers/diplom.1.bib.gz** There are plans to expand this bibliography from time to time; the sequels will have names diplom.2.bib.gz, etc.

Bibliography at IlliGAL

A bibliography on Genetic Algorithms compiled by David E. Goldberg, Kelsey Milman, and Christina Tidd is available as IlliGAL Report No 92008 (see Q14), via ftp from: **gal4.ge.uiuc.edu/pub/papers/IlliGALs/92008part1.ps.Z** and 92008part2.ps.Z

GAPHD Bibliography Collection

Martyn Amos <Martyn.Amos@dcs.warwick.ac.uk> has assembled a collection of bibliographies from various sources, tidied up the entries and removed duplicates. The collections are as follows:

Alife.bib.gz - General Artificial Life
ICGA-93.bib.gz - Proc. International Conference on GAs (1993)
chaos.bib.gz - Chaos theory
ga+nn.bib.gz - GAs and neural networks
ga.bib.gz - General GA references
ga2.bib.gz - General GA references
parallelGA.bib.gz - Parallel GAs
theory.bib.gz - Theoretical computer science (bias towards graph
theory, stochasic modelling and pobability theory)
misc.bib.gz - Miscellaneous topics (eg. Internet)

There are about 6200 references in total, although the biggest file by far is theory.bib, which is not directly related to EC. The references are in BibTeX format. The files are available by FTP from **ftp.dcs.warwick.ac.uk/pub/gaphd/Bibliographies/** or by WWW from **http://www.dcs.warwick.ac.uk/~martyn/ga.html**

Genetic Programming Bibliography

A collection of Genetic Programming references (and other tools) is maintained by Bill Langdon <W.Langdon@cs.ucl.ac.uk> and is available via anonymous ftp from **cs.ucl.ac.uk/genetic/biblio/**

Evolutionary Models in the Social Sciences

Edmund Chattoe <E.Chattoe@surrey.ac.uk> has set up a bibliography on Evolutionary Models In Economics and the Social Sciences. The latest copy of the EMSS bibliography and some accompanying notes can be found at **http://www.soc.surrey.ac.uk/~scs1ec/emssbib.html**

GAs and Economics

Bernard Manderick <manderic@cs.few.eur.nl> has compiled a bibliography on the use of GAs in economics, and this was published in GA-Digest, v7n4 (with some followup comments in v7n5 & v7n7). This can be retrieved by FTP from **ftp.aic.nrl.navy.mil/pub/galist/digests/v7n4** (see Q15.1).

GAs in Control

Carlos Fonseca <fonseca@acse.sheffield.ac.uk> has compiled a bibliography of about 50 references on GAs in Control, and it was published in GA-Digest, v7n18. This can be retrieved by FTP from **ftp.aic.nrl.navy.mil/pub/galist/digests/v7n18** (see Q15.1).

Parallel GAs

A parallel GA bibliography is available via ftp from: **unix.hensa.ac.uk/pub/parallel/faqs/parallel-genetic-algorithms**

Andreas Uhl <uhl@wst.wst.edvz.sbg.ac.at> has also compiled a parallel GA bibliography with about 80 entries. It is available by WWW in: **http://www.mat.sbg.ac.at/~uhl/GA.html**

Genetic Programming

John Koza <koza@CS.Stanford.EDU> has compiled an annotated bibliography on GP, and about 60 references were published in GA-Digest, v7n30. This can be retrieved by FTP from **ftp.aic.nrl.navy.mil/pub/galist/digests/v7n30** or from **ENCORE** (See Q15.3) in file **refs/gp-ref.gz**

GAs and protein folding

Melanie Mitchell <mm@santafe.edu > has compiled a bibliography of about 40 references on this topic, and it was published in GA-Digest, v7n33. This can be retrieved by FTP from **ftp.aic.nrl.navy.mil/pub/galist/digests/v7n33** (see Q15.1).

GAs in Image Processing and Computer Vision

Kyeongmo Park <kpark@cs.gmu.edu> has compiled a bibliography of about 20 references on this topic, and it was published in GA-Digest, v8n10. This can be retrieved by FTP from **ftp.aic.nrl.navy.mil/pub/galist/digests/v8n10** (see Q15.1).

GAs in telecommunications and data networks

Bhaskar Krishnamachari <bhaskar@ee.cornell.edu> has compiled the following bibliographies:

The application of genetic algorithms to telecommunication systems: a bibliography **http://www.ee.cornell.edu/~bhaskar/gacomm-bib.html**

The application of genetic algorithms to the design and optimization of data networks: a bibliography **http://www.ee.cornell.edu/~bhaskar/ganet-bib.html**

Masters and PhD theses

Richard K. Belew has collected information on approximately 2600 Masters and Ph.D. theses, nominally in the area of AI. The entire list (about 170KB) is available for anonymous FTP at: **cs.ucsd.edu/pub/rik/aigen.rpt** Questions, suggestions, additions etc. to <rik@cs.ucsd.edu>.

Q10.9: Videos?

Fogel, D.B. (1997) "An Introduction to Evolutionary Computation," for ordering contact <customer.service@ieee.org>

Sims, K. (1990) "Panspermia", ACM SIGGRAPH Video Review. Ordering information from **http://www.siggraph.org/publications/video-review/SVR.html**

Langton, C.G. (ed) (1992) "Artificial Life II Video Proceedings" The Advanced Book Program of the Santa Fe Institute: Studies in the Sciences of Complexity, Addison Wesley, ISBN 0-201-55492-5. [ALIFEII-V]:

Koza, J.R. & Rice, J.P. (1992) "Genetic Programming: The Movie", Cambridge, MA: MIT Press. See GP-faq for an order form. (see Q15)

The Santa Fe Institute has produced a thirteen minute promotional video, which includes a five minute segment discussing the Tierra research project, illustrated with a very high quality animation produced by the Anti Gravity Workshop in Santa Monica, CA. To obtain the video, contact the Santa Fe Institute at: 1660 Old Pecos Trail, Suite A, Santa Fe, New Mexico 87501 (Tel: 505-984-8800, Fax: 505-982-0565, Net: <email@santafe.edu>) or contact Linda Feferman: <fef@santafe.edu> or <0005851689@mcimail.com>

Q10.10: CD-ROMs?

PTF for AI by CMU

Carnegie Mellon University is establishing an Artificial Intelligence Repository to contain public domain and freely distributable software, publications, and other materials of interest to AI researchers, educators, and students. The AI Repository will be accessible by anonymous FTP and Andrew File System (AFS) without charge (See Q15.3). The contents of the repository will also be published by Prime Time Freeware as an inexpensive mixed-media (Book/CD-ROM) publication.

For your information, here is a precis of the CD-ROM:

PTF for AI is a periodic collection of AI-related source code and documentation. PTF for AI in no way modifies the legal restrictions on any package it includes. The first issue (1-1; Summer, 1993) consisted of an ISO-9660 CD-ROM bound into a ˜100 page book. It contained ˜600 MB of gzipped archives (2+ GB uncompressed and unpacked). Cost: $60 US.

For more information contact: *Mark Kantrowitz*, Archivist, CMU AI Repository, Editor, PTF for AI. Net: <mkant+repository@cs.cmu.edu>, Tel: +1 412-268-2582, Fax: +1 412-681-5739.

AI CD-ROM by NCC

Network Cybernetics Corporation has released a new CD-ROM title, the AI CD-ROM Revision 3 (ISBN 1-886376-01-8). This is the newest version of an annually updated collection of artificial intelligence programming and research tools. This ISO-9660 format CD-ROM contains thousands of programs, source code collections, tutorials, research papers, Internet journals, and other resources. Previous versions of the AI CD-ROM are currently in use as teaching aids for AI-related University courses, as research aids to computer scientists, and as a source of advanced computer programming tools for application program developers around the world.

The AI CD-ROM contains thousands of up to date files covering a wide range of topics including: Fuzzy Logic, Genetic Algorithms, Neural Networks, Expert Systems, Robotics, Machine Vision, Natural Language, Prolog, Lisp, Embedded AI, Virtual Reality, Cellular Automata, Chaos, Fractals, and more. The disc is divided into topical subdirectories and each directory contains an index file with descriptive listings of the contents. The AI CD-ROM has received good reviews in many magazines including Byte (Jerry Pournelle, March '93) and IEEE Computer (J. Zalewski, July '93), CD-ROM Professional and others. The CD-ROM has a list price of $89.00.

For people wanting to see a complete listing of the CD's contents, FTP to **ftp.ncc.com:/** and get the file AICD3.ZIP. The file is also available from the Compuserve AIEXPERT forum, and the NCC dial-up BBS at 214-258-1832. Also check out the WWW site at: **http://www.ncc.com/cdroms/ai/index.html**

Enquiries to: Network Cybernetics Corporation, 4201 Wingren Road, Suite 202, Irving, TX 75062-2763, USA <ai-info@ncc.com>

Q10.11: How do I get a copy of a dissertation?

All US American dissertations are available from: UMI Dissertation Information Service, University Microfilms International, A Bell & Howell Information Company, 300 N. Zeeb Road, Ann Arbor, Michigan 48106, USA. Tel.: 800-521-0600, or +1 (313) 761-4700

Q11: What EC related journals and magazines are there?

1. Dedicated EC Journals:

Evolutionary Computation

Published quarterly by: MIT Press Journals, 55 Hayward Street, Cambridge, MA 02142-1399, USA. Tel: (617) 253-2889, Fax: (617) 258-6779, <journals-orders@mit.edu>

Along with the explosive growth of the computing industry has come the need to design systems capable of functioning in complex, changing **ENVIRONMENTs.** Considerable effort is underway to explore alternative approaches to designing more robust computer systems capable of learning from and adapting to the environment in which they operate.

One broad class of such techniques takes its inspiration from natural systems with particular emphasis on evolutionary models of computation such as **GAs, ESs. CFS,** and **EP.** Until now, information on these techniques has been widely spread over numerous disciplines, conferences, and journals. [eds note: The editorial board reads like a who-is-who in **EC.]** For paper e-mail submission, use one of the following addresses:

- America: *John Grefenstette* <gref@aic.nrl.navy.mil>
- Europe: *Heinz Muehlenbein* <heinz.muehlenbein@gmd.de>
- Asia: *Hiroaki Kitano* <kitano@csl.sony.co.jp>
- Ed-in-chief: *Ken De Jong* <kdejong@aic.gmu.edu>

Please note, that submissions should be sent to one of the sub-editors. Grefenstette and Kitano accept LaTeX or PostScript submissions.

BioSystems

Journal of Biological and Information Processing Sciences, Elsevier Science Publishers, P.O. Box 1527, 1000 BM Amsterdam, The Netherlands.

BioSystems encourages experimental, computational, and theoretical articles that link biology, evolutionary thinking, and the information processing sciences. The link areas form a circle that encompasses the fundamental nature of biological information processing, computational modeling of complex biological systems, evolutionary models of computation, the application of biological principles to the design of novel computing systems, and the use of biomolecular materials to synthesize artificial systems that capture essential principles of natural biological information processing.

Topics: Molecular **EVOLUTION:** Self-organizing and self-replicating systems, Origin and evolution of the genetic mechanism; Biological Information Processing: Molecular recognition, Cellular control, Neuromuscular computing, Biological adaptability, Molecular computing technologies; **EVOLUTIONARY SYSTEMS:** Stochastic **EVOLUTIONARY ALGORITHMs,** Evolutionary **OPTIMIZATION, SIMULATION** of genetic and ecological systems, Applications (neural nets, machine learning, robotics))

IEEE Transactions on Evolutionary Computation

The IEEE Transactions on Evolutionary Computation will publish archival journal quality original papers in **EVOLUTIONARY COMPUTATION** and related areas, with particular emphasis on the practical application of the techniques to solving real problems in industry, medicine, and other disciplines. Specific techniques include but are not limited to **EVOLUTION STRATEGIEs, EVOLUTIONARY PROGRAMMING, GENETIC ALGORITHMs,** and associated methods of **GENETIC PROGRAMMING** and **CLASSIFIER SYSTEMs.** Papers emphasizing mathematical results should ideally seek to put these results in the context of algorithm design, however purely theoretical papers will be considered. Other papers in the areas of cultural algorithms, **ARTIFICIAL LIFE,** molecular computing, evolvable hardware, and the use of simulated evolution to gain a better understanding of naturally evolved systems are also encouraged.

Papers must conform to IEEE standard submission guidelines which are available in IEEE transactions (for example, see the IEEE Transactions on Neural Networks or the IEEE Transactions on Fuzzy Systems). Those wanting to receive an author's information booklet from the IEEE can request this at <trans@ieee.org>.

Six (6) hard copies of the manuscript should be sent to: David B. Fogel, Editor-in-Chief, IEEE Transactions on Evolutionary Computation, c/o Natural Selection, Inc., 3333 N. Torrey Pines Ct., Suite 200, La Jolla, CA 92037, USA.

The editor-in-chief will be pleased to comment on the suitability of other submissions at the request of the authors. Further questions can be directed to <d.fogel@ieee.org>. The transactions will appear quarterly.

2. Related Journals:

Complex Systems

Published by: Complex Systems Publications, Inc., P.O. Box 6149, Champaign, IL 61821-8149, USA.

Complex Systems devotes to the rapid publication of research on the science, mathematics, and engineering of systems with simple components but complex overall behavior. Try **finger**(1) on <jcs@wri.com> for additional info.

Machine Learning

Published by: Kluwer Academic Publishers, P.O. Box 358, Accord Station, Hingham, MA 02018-0358 USA.

Machine Learning is an international forum for research on computational approaches to learning. The journal publishes articles reporting substantive research results on a wide range of learning methods applied to a variety of task domains. The ideal paper will make a theoretical contribution supported by a computer implementation.

The journal has published many key papers in learning theory, reinforcement learning, and decision tree methods. The journal regularly publishes special issues devoted to **GAs** and **CFS** as well.

Adaptive Behavior

Published quarterly by: MIT Press Journals, details above.

Broadly, behavior is adaptive if it deals successfully with changes circumstances. For example, when surprised, a hungry --but environmentally informed-- mouse may dart for cover rather than another piece of cheese. Similarly, a tripped-up **ROBOT** [eds note: not necessarily built by Sirius Cybernetics Corp.] could get back on its feet and accomplish a moonrock-finding mission if it had learned to cope with unanticipated lunar potholes.

Adaptive Behavior thus takes an approach complementary to traditional **AI.** Now basic abilities that allow animals to survive, or robots to perform their mission in unpredictable **ENVIRONMENTs,** will be studied in preference to more elaborate and human-specific abilities.

The journal also aims to investigate which new insights into intelligence and cognition can be achieved by explicitly taking into account the environment feedback --mediated by behavior-- that an animal or a robot receives, instead of studying components of intelligence in isolation.

Topics: **INDIVIDUAL** and Collective Behavior. Neural Correlates of Behavior. Perception and Motor Control. Motivation and Emotion. Action **SELECTION** and Behavioral Sequences. Internal World Models. Ontogeny, Learning, and **EVOLUTION.** Characterization of environments.

Artificial Life

Published quarterly by: MIT Press Journals, details above.

Artificial Life is intended to be the primary forum for the dissemination of scientific and engineering research in the field of **ARTIFICIAL LIFE.** It will report on synthetic biological work being carried out in any and all media, from the familiar "wetware" of organic chemistry, through the inorganic "hardware" of mobile robots, all the way to the virtual "software" residing inside computers.

Research topics ranging from the fabrication of self-replicating molecules to the study of evolving **POPULATIONs** of computer programs will be included.

There will also be occasional issues devoted to special topics, such as L-Systems, **GENETIC ALGORITHMs,** in-vitro evolution of molecules, artificial cells, computer viruses, and many social and philosophical issues arising from the attempt to synthesize life artificially.

[eds note: The editorial board reads like a who-is-who in **ALIFE]**

Evolutionary Economics

Published quarterly by: Springer-Verlag New York, Inc., Service Center Secaucus, 44 Hartz Way, Secaucus, NJ 07094, USA. Tel: (201) 348-4033, Fax: (201) 348-4505.

Evolutionary Economics aims to provide an international forum for a new approach to economics. Following the tradition of Joseph A. Schlumpeter, it is designed to focus on original research with an evolutionary conception of the economy. The journal will publish articles with strong emphasis on dynamics, changing structures (including technologies, institutions, beliefs, imitation, etc.). It favors interdisciplinary analysis and is devoted to theoretical, methodological and applied work.

Research areas include: industrial dynamics; multi-sectoral and cross-country studies of productivity; innovations and new technologies; dynamic competition and structural change in a national and international context; causes and effects of technological, political and social changes; cyclic processes in economic evolution; the role of governments in a dynamic world; modeling complex dynamic economic systems; application of concepts, such as self-organization, bifurcation, and chaos theory to economics; evolutionary games.

Q12: What are the important conferences/proceedings on EC?

1. Dedicated EC Conferences:

ICGA: International Conference on Genetic Algorithms

Major international conference held in North America in odd-numbered years. Covers all aspects of **EVOLUTIONARY COMPUTATION.** The 1999 conference will be held on July 14--17 in Orlando, Florida, in conjunction with the annual Genetic Programming conference. It will be titled GECCO (Genetic and Evolutionary Computation Conference). Details from **http://www-illigal.ge.uiuc.edu/gecco/**

The 1997 conference was at Michigan State University, East Lansing, USA. Details from **http://GARAGe.cps.msu.edu/icga97/index.html**

Proceedings of the 1st International Conference on Genetic Algorithms (1985) J.J. Grefenstette (ed) [ICGA85]: and Proc. of the 2nd Int'l Conf. on Genetic Algorithms (1987) J.J. Grefenstette (ed) [ICGA87]: available from Lawrence Erlbaum Associates, Inc., 365 Broadway, Hillsdale, New Jersey, 07642, (800) 926-6579.

Proc. of the 3rd Int'l Conf. on Genetic Algorithms (1989) J.D. Schaffer (ed) [ICGA89]: and Proc. of the 4th Int'l Conf. on Genetic Algorithms (1991) R.K. Belew and L.B. Booker (eds) [ICGA91]: and Proc. of the 5th Int'l Conf. on Genetic Algorithms (1993) S. Forrest (ed) [ICGA93]: and Proc. of the 6th Int'l Conf. on Genetic Algorithms (1995) [ICGA95]: available from Morgan Kaufmann Publishers, Inc., San Francisco (415-392-2665). <morgan@unix.sri.com>

FOGA: Foundations of Genetic Algorithms

Major international workshop focusing on theoretical aspects of **EC,** that's usually limited to some 50 participants and is usually held somewhere in North America. FOGA 5, however, was held in Leiden, The Netherlands on 24–26 September 1998. Details from: **http://www.wi.leidenuniv.nl/CS/ALP/foga98.html**

Foundations of Genetic Algorithms (1991) G.J.E. Rawlins (ed) [FOGA91]: and Foundations of Genetic Algorithms 2 (1993) L.D. Whitley [FOGA93]: available from Morgan Kaufmann Publishers, Inc., San Francisco (415-392-2665). <morgan@unix.sri.com>

FOGA 3 took place in 1994. Enquires to: Darrell Whitley, <whitley@cs.colostate.edu>. FOGA 4 took place from August 3-5 1996 in San Diego, California. Details from **http://www.aic.nrl.navy.mil/galist/foga/**

PPSN: Parallel Problem Solving from Nature

Major international conference held in Europe in even-numbered years. Covers all aspects of problem solving inspired by natural processes. The 1998 conference was held in Amsterdam, The Netherlands, September 27 – October 1. Information from: **http://www.wi.leidenuniv.nl/CS/ALP/ppsn98.html** Further information on all PPSN conferences is available from: **http://LS11-www.informatik.uni-dortmund.de/PPSN/**

Parallel Problem Solving from Nature, (1990) H.-P. Schwefel and R. Männer (eds) [PPSN90]: published by Springer-Verlag, 175 5th Avenue, New York, NY, 10010, (212) 460-1500. Parallel Problem Solving from Nature 2, (1992) R. Männer and B. Manderick (eds) [PPSN92]: published by North-Holland, Elsevier Science Publishers, Sara Burgerhartstraat 25, P.O. Box 211, 1000 AE Amsterdam, The Netherlands. Parallel Problem Solving from Nature 3, (1994) Y. Davidor (ed.), [PPSN94]: PPSN96 was held in Berlin, September 1996.

EP: Annual Conference on Evolutionary Programming

Major international annual conference held in USA. Covers all aspects of EC with emphasis on **EP** related research. The 1999 conference will be held in conjunction with the ICEC (See below).

The 1997 conference was held on April 13−16 in Indianapolis. Details from Pete Angeline <pja@lfs.loral.com>. The 1996 conference was held on Feb 29−March 3. The 1995 conference was held on March 1–4. Details from David Fogel <fogel@sunshine.ucsd.edu>.

Proceedings of the 1st Annual Conference on Evolutionary Programming, (1992) D.B. Fogel and W. Atmar (eds), [EP92]:, and Proc. of the 2nd Annual Conf. on Evolutionary Programming, (1993) D.B. Fogel and W. Atmar (eds), [EP93]: published by the Evolutionary Programming Society, 9363 Towne Centre Dr., San Diego, CA 92121, Attn: Bill Porto, Treasurer (cf Q13). Proceedings of the Third Annual Conference on Evolutionary Programming, (1994) A.V. Sebald and L.J. Fogel (eds), [EP94]:, World Scientific Publishers, River Edge, NJ.

ICEC: IEEE Conference on Evolutionary Computation

Major international conference covering all aspects of EC. The sixth conference will be held in Washington DC, from 6−9 July 1999. It is titled the *Congress on Evolutionary Computation,* (CEC) and it will be held in conjunction with the *Evolutionary Programming Conference* (EP) and *Genetic Algorithms in Engineering Systems: Innovations and Applications* (GALESIA). Details from **http://garage.cps.msu.edu/cec99/**

The fifth conference was held in Anchorage, Alaska, USA, from May 4−9 1998. Details from **http://www.arc.unm.edu/wcci-98/icec.html** . The fourth was on April 14−17 1997 in Indianapolis (in conjunction with EP97). The third was on May 20−22 1996 in Nagoya, Japan, details from **http://www.bioele.nuee.nagoya-u.ac.jp/ICEC96/** . The second was on 29 Nov--1 Dec 1995 in Perth, Australia. Details from <ec95@ee.uwa.edu.au> . The first took place in June 1994 at the World Congress on Computational Intelligence, Florida.

Proceedings of the 1st IEEE Conference on Evolutionary Computation, (1994) D.B. Fogel (ed.) (2 Volumes). Published by IEEE, 445 Hoes Lane, PO Box 1331, Piscataway, NJ 08855-1331. Also, talks from invited speakers are published in "Computational Intelligence Imitating Life" (1994) J.M. Zurada, R.J. Marks, C.J. Robinson (eds), IEEE.

Genetic Programming

The 1998 conference dedicated to **GP** was held on July 22−25 at the University of Winconsin. The 1999 conference will be held on July 14−17 in Orlando, Florida, in conjunction with the ICGA (see above).

Details of the GP conferences can be obtained from: **http://www.genetic-programming.org** or from <gp@aaai.org>.

The first conference was held on July 28--31 1996 at Stanford University, California. Details from: **http://www.cs.brandeis.edu/˜zippy/gp-96.html**

2. Related Conferences:

Alife: International Conference on Artificial Life

Proceedings of the 1st International Conference on **ARTIFICIAL LIFE,** (1989) C.G. Langton (ed), Santa Fe Institute Studies in the Sciences of Complexity, Proc. Vol. VI, [ALIFEI]: and Proc. of the 2nd Int'l Conf. on Artificial Life II, (1992) C.G. Langton, C. Taylor, J. Doyne Farmer and S. Rasmussen (eds), Santa Fe Institute Studies in the Sciences of Complexity, Proc. Vol. X, [ALIFEII]: and Proc. of the 3rd Int'l Conf. on Artificial Life III, (1993) C.G. Langton (ed), [ALIFEIII]: published by Addison Wesley, Redwood City, CA, USA.

Artificial life IV, was organized by Rodney Brooks, MIT AI Lab, <alife@ai.mit.edu> and held on July 6-8, 1994. Proceedings edited by R. Brooks and P. Maes. [ALIFEIV]:

ECAL: European Conference on Artificial Life

Proceedings of the 1st European Conference on Artificial life, (1991) F.J. Varela and P. Bourgine (eds), [ECAL91]: and Proc. of the 2nd European Conf. on **ALIFE:** Self-organization and life, from simple rules to global complexity, (1993), (? eds) (? pub) [ECAL93]: published by MIT Press, Cambridge, MA, USA.

ECML: European Conference on Machine Learning

Machine Learning: ECML-93, Proc. European Conf. on Machine Learning, (1993) P.B. Brazil (ed), [ECML93]: published by Springer, New York, NY, USA.

ICANNGA: International Conference on Artificial Neural Networks and Genetic Algorithms

Held every 2 years since 1993. The 1997 conference is on April 1–4 in Norwich, England. Details from **http://www.sys.uea.ac.uk/Research/ResGroups/MAG/ICANNGA97/**

SAB: International Conference on Simulation of Adaptive Behavior

From Animals to Animats. Proceedings of the 1st International Conference on **SIMULATION** of Adaptive Behavior, (1991) [SAB90]: J.-A. Meyer and S.W. Wilson, ISBN 0-262-63138-5, and Proc. of the 2nd Int'l Conf. on Simulation of Adaptive Behavior, (1993) [SAB92]:, J.-A. Meyer, H. Roitblat and S.W. Wilson (eds) and Proc. of the 3rd Int'l Conf. on Simulation of Adaptive Behavior, [SAB94]:, P. Husbands, J.-A. Meyer and S.W. Wilson (eds) published by MIT Press, Cambridge, MA, USA.

SAB96 took place on September 9–13, 1996 in Cape Cod, MA USA. Details from **http://www.cs.brandeis.edu/conferences/sab96**

3. Pointers to upcoming Conferences:

The Genetic Algorithm Digest

Aka "GA-Digest" always starts with a "Calendar of GA-related Events," i.e. a list of upcoming conferences, covering the complete field of **EAs** (see Q15.1), available from **http://www.aic.nrl.navy.mil/galist/**

The Artificial Life Digest

Aka "Alife digest" always starts with a "Calendar of Alife-related Events," that lists conferences, workshops, etc. (cf Q15)

The Evolutionary Programming Digest

Aka "EP-digest" doesn't list conferences explicitly, like the previously mentioned ones, but carries most CFP's; that can be looked at in the backissues folder: **amazon.eng.fau.edu/pub/ep-list/digest/** (cf Q15)

Q13: What Evolutionary Computation Associations exist?

ISGA: International Society on Genetic Algorithms

The ISGA is a mostly fascinating society: it neither has a membership fee (which makes it even more fascinating), nor an address. However, ISGA meetings usually take place during ICGA conferences, in so-called business meetings (BMs). [eds note: So during a conference, ask for BMs, if you want to join; or be ready to dart out of the room if you don't...]

EPS: Evolutionary Programming Society

Membership is $40/year ($10/year for students with id) and also gives you a discounted registration at the annual conference. You can also order **EP** proceedings ($30/members, $45/other) from EPS.

Address: Evolutionary Programming Society, 9363 Towne Centre Dr., San Diego, CA 92121, Attn: Bill Porto, Treasurer.

Q14: What Technical Reports are available?

Technical reports are informally published, unrefereed papers giving up-to-date information on what is going on at research institutes. Many later go on to be formally published in journals or at conferences.

TCGA Reports

The Clearing House for Genetic Algorithms (TCGA) at the Univ. of Alabama (Tuscaloosa) distributes TCGA technical reports. A number of these are now available in compressed Postscript form via **FTP** from: **aramis.cs.ua.edu/pub/tech-reports/** Read the file README first.

Contact: Robert Elliott Smith, Department of Engineering of Mechanics, Room 210 Hardaway Hall, The University of Alabama, P.O. Box 870278, Tuscaloosa, AL 35487, USA. Tel: (205) 348-1618, <rob@comec4.mh.ua.edu>, or Dr. Ron Sun <rsun@athos.cs.ua.edu>

IlliGAL Reports

The Illinois Genetic Algorithms Laboratory distributes IlliGAL technical reports, as well as reprints of other publications; they are available in hardcopy and can be ordered from: IlliGAL Librarian, Department of General Engineering, 117 Transportation Building, 104 South Mathews Avenue, Urbana, IL 61801-2996, USA. <library@gal1.ge.uiuc.edu>

NOTE: When ordering, please include your surface mail address!

IlliGAL also have an anonymous-FTP server, holding most of the existing IlliGAL reports, at: **gal4.ge.uiuc.edu/pub/papers/IlliGALs/** There is also a WWW home page with a complete list, order form, and other information at: **ftp://gal4.ge.uiuc.edu/illigal.home.html**

SyS Reports

The Systems Analysis Research Group (SyS) at the University of Dortmund, maintains an experimental anonymous **FTP** server: **lumpi.informatik.uni-dortmund.de/pub/** On lumpi you can find SyS-Reports from 1992 on. (Get "/pub/ls-Ral.Z" and look for "papers" folders, the server is sorted by **EA** paradigms, i.e. "/pub/GA/papers" contains papers related to **GAs,** etc.). A strongly recommended, and quarterly updated, report is a list of current applications of GAs, **EP** and **ESs;** get "/pub/EA/papers/ea-app.ps.gz" (SyS-2/92).

Bionics Reports

The Bionics and **EVOLUTION** Techniques Laboratory at the Technical University of Berlin maintains an anonymous **FTP** server: **ftp-bionik.fb10.tu-berlin.de/pub/** On ftp-bionik you find reports and software, related to **EVOLUTIONARY ALGORITHMs** and Artificial Neural Networks.

University College London Reports

A number of **GENETIC ALGORITHM** reports produced by UCL are available via anonymous **FTP** at **cs.ucl.ac.uk/genetic/papers/** Abstracts of others can be obtained via WWW at **http://www.cs.ucl.ac.uk/rns/**

Other Sources of Reports

Reports are also available from some of the sources listed in Q15.1, Q15.2 and Q15.3.

Q15: What information is available over the net?

A whole lot of information is available "electronically" via the internet, accessible using e-mail or (more easily) **FTP.** There are electronic digests (see Q15.1), electronic mailing lists (see Q15.2), online FTP repositories (see Q15.3), and various **USENET** news groups (see Q15.4).

Q15.1: What digests are there?

Digests are regulated, moderated, information sources in which many contributions are combined together before being posted out to subscribers, usually on a regular basis (eg. weekly). Mailing lists are listed in Q15.2.

Genetic Algorithm Digest

The **GA** research community exchanges news, CFP's, etc. through this (approximately weekly) digest, currently moderated by Bill Spears (formerly by Connie Ramsey and by Alan C. Schultz, Naval Research Laboratory, Washington, DC).

A statistic published in v7,i3 stated that GA-digest is sent out world-wide to 1800 addresses in 28 countries. The digest is also posted to the **comp.ai.genetic** newsgroup.

- Send administrative requests to <ga-list-REQUEST@aic.nrl.navy.mil>
- The anonymous **FTP** archive: **ftp.aic.nrl.navy.mil/pub/galist/** contains back issues, GA-code, conference announcements (in "/pub/galist/information/conferences") and many other things. Info in "/pub/galist/FTP".
- The archive may also be accessed via the World Wide Web at **http://www.aic.nrl.navy.mil/galist** Also, links are given to many interesting sites around the World with material related to **EVOLUTIONARY COMPUTATION.**

Artificial Life Digest

The **ALIFE** research community exchanges news, CFP's, etc. through this digest, edited by Liane Gabora and Rob Collins of the **ARTIFICIAL LIFE** Research Group at UCLA.

- Send administrative requests to <alife-REQUEST@cognet.ucla.edu>
- Anonymous **FTP** archive: **ftp.cognet.ucla.edu/pub/alife/**

Evolutionary Programming Digest

The digest is intended to promote discussions on a wide range of technical issues in evolutionary **OPTIMIZATION,** as well as provide information on upcoming conferences, events, journals, special issues, and other items of interest to the **EP** community. Discussions on all areas of **EVOLUTIONARY COMPUTATION** are welcomed, including **ARTIFICIAL LIFE, EVOLUTION STRATEGIEs,** and **GENETIC ALGORITHMs.** The digest is meant to encourage interdisciplinary communications. Your suggestions and comments regarding the digest are always welcome.

To subscribe to the digest, send mail to <ep-list-REQUEST@magenta.me.fau.edu> and include the line "subscribe ep-list" in the body of the text. Further instructions will follow your subscription. The digest is moderated by N. Saravan of Florida Atlantic University.

Q15.2: What mailing lists are there?

Mailing lists are unregulated, unmoderated, information sources in which messages sent in by subscribers are posted out immediately and individually to all other subscribers. Digests are listed in Q15.1.

Genetic Programming Mailing List

The **GP** community uses this list as a discussion forum, news exchange and **FAQ** distribution channel, originally set up by John Koza and James Rice at Stanford.

- Admin requests: <genetic-programming-REQUEST@cs.stanford.edu>
- The anonymous **FTP** archive includes a lengthy, but "mostly interesting" FAQ by James Rice on GP related subjects. The archive used to be held at **ftp.io.com/pub/genetic-programming/** and a HTML version used to be accessible at: **http://www.salford.ac.uk/docs/depts/eee/genetic.html** but both these sites have now gone (does anyone know where they are now?)

Tierra Mailing List

Thomas Ray's Tierra is discussed elsewhere (see Q4.1); here's how to obtain Tierra electronically and get in contact with other users.

- Admin requests: <tierra-REQUEST@life.slhs.udel.edu>
- Anonymous **FTP** archive: **tierra.slhs.udel.edu/pub/** (tierra, almond, beagle, etc.)

UK's Evolutionary-Computation mailing list

- Admin details: <evolutionary-computing-request@mailbase.ac.uk>

GEnetic Algorithm Research Student mailing list

Provides a forum for research students interested in **GENETIC ALGORITHMs.**

- Admin requests: <gears-request@research.de.uu.net>

GEARS articles can be viewed at: **http://research.de.uu.net:8080/~gears/** while the old GEARS archive is kept at: **http://research.de.uu.net:8080/~gears/archive/**

GANN: Genetic Algorithms and Neural Networks

This list will focus on the use of **EVOLUTIONARY ALGORITHMs (GENETIC ALGORITHMs, GENETIC PROGRAMMING** and their variants) in the **EXPLORATION** of the design space of (artificial) neural network architectures and algorithms. The list will be semi-moderated to keep the signal to noise ratio as high as possible. (This list was formerly known as the neuro-evolution e-mail list.)

- Admin requests/enquiries: gann-request@cs.iastate.edu
- Subscription requests to the admin address with Subject: subscribe

gattbl: Timetabling mailing list

This group is for people using **GAs** and other techniques for exam or course scheduling for academic institutions. To subscribe, send email to <ttp-request@cs.nott.ac.uk>.

Evolutionary Models in the Social Sciences

See Q10.8 for details.

Genetic Algorithms in Production Scheduling

The GASched list is for discussion of the use of **GENETIC ALGORITHMs** on Production Scheduling Problems (only). Possible subjects for the list include: **GAs** for job-shop scheduling theory, GAs for practical problem solving in industry, problem representation within the GA, combinatorial optimisation techniques for scheduling problems, results & effects of GA-based systems working in industry, techniques for improving performance, problem data, or any other burning issues which come into GAs for production scheduling.

A full introduction can be obtained by mailing <listproc@sheffield.ac.uk> with no subject line and 'info gascheduling' in the body of the message.

To subscribe to the list, email <listproc@sheffield.ac.uk> with the body of the message containing 'subscribe gascheduling YOUR NAME'. Please dont include anything else in the message, and leave the subject empty. For help on how to use the automated software, and some other commands which may be available in future, mail <listproc@sheffield.ac.uk> with 'HELP' in the body of your message, and no subject line.

For non-standard administration requests, or if you are having problems with the automated address, please email: <gascheduling-request@sheffield.ac.uk> These messages will be dealt with manually, and so may take a couple of days for a response.

There is also a related Web site at: **http://www.shef.ac.uk/~gaipp/index.html**

Autopoiesis

There is an Autopoiesis Email List for the discussion of the theory of Autopoiesis of H. Maturana and F. Varela. Autopoiesis means self-production and concerns self-organizing systems.

To join send a message containing the text: SUB AUTOPOIESIS to <listserv@think.net>

To see what other systems and philosophy lists exist at this site send the message: HELP instead.

Q15.3: What online information repositories are there?

Many research institutes have online repositories of information which my be retrieved using **FTP** or HTTP (World Wide Web).

NOTE: See also Q14 above.

ENCORE

ENCORE (The EvolutioNary COmputation REpository network) is a collection of **FTP** servers/World Wide Web sites providing a wealth of information in the area of **EC,** from technical reports, copies of journal articles, down to source code for various **EAs.** ENCORE acts as a distributor of much material generated at research institutes (and other places) which don't necessarily have their own FTP servers.

Each node of Encore is referred to as an "EClair". There are numerous nodes around the world, all carrying copies of the same information. The sites may be accessed using FTP or WWW browsers. Sites offering HTTP access are the best to use if using a WWW browser. They include:

- UUNET Deutschland GmbH (Germany): **http://research.de.uu.net:8080/encore/**
- The University of Girona (Spain) **http://gnomics.udg.es/~encore/**
- The University of Granada (Spain): **http://krypton.ugr.es/~encore/**
- The University of Birmingham (UK) **http://www.cs.bham.ac.uk/Mirrors/ftp.de.uu.net/EC/clife/**
- The Santa Fe Institute (USA): **http://alife.santafe.edu/~joke/encore/**
- Purdue University, West Lafayette, IN (USA): **http://www.cs.purdue.edu/coast/archive/clife/Welcome.html**
- The Chinese University of Hong Kong: **http://www.cs.cuhk.hk/pub/EC/Welcome.html**

Other sites offer FTP access (slow if using WWW). If using FTP, omit the initial "ftp://" and the final "Welcome.html" in the file specification in order to access the top-level directory. The FTP sites include:

- UUNET Deutschland GmbH (Germany): **ftp://ftp.de.uu.net/pub/research/softcomp/EC/Welcome.html**

• Technical University of Berlin (Germany): **ftp://ftp-bionik.fb10.tu-berlin.de/pub/EC/Welcome.html**

• Ecole Polytechnique, Palaiseau (France): **ftp://blanche.polytechnique.fr/pub/eark/EC/Welcome.html**

• The University of Oviedo (Spain): **ftp://zeus.etsimo.uniovi.es/pub/EC/Welcome.html**

• The Santa Fe Institute (USA): **ftp://alife.santafe.edu/pub/USER-AREA/EC/Welcome.html**

• The California Institute of Technology (USA): **ftp://ftp.krl.caltech.edu/pub/EC/Welcome.html**

• Wayne State University, Detroit (USA): **ftp://ftp.cs.wayne.edu/pub/EC/Welcome.html**

• The Michigan State University, East Lansing (USA): **ftp://ftp.egr.msu.edu/pub/EC/Welcome.html**

• Purdue University, West Lafayette, IN (USA): **ftp://coast.cs.purdue.edu/pub/EC/Welcome.html**

• The Chinese University of Hong Kong: **ftp://ftp.cs.cuhk.hk/pub/EC/Welcome.html**

• University of Cape Town (South Africa): **ftp://ftp.uct.ac.za/pub/mirrors/EC/Welcome.html**

• Center of Technological Education of Parana, Curitiba (Brazil): **ftp://ftp.cefetpr.br/pub/EC/Welcome.html**

Well worth getting is "The Navigator's Guide to ENCORE", a handbook to this service, in file:

- **handbook/encore.ps.gz** (A4 paper) or
- **handbook/encore-US.ps.gz** (US letter size paper).

Encore is administered by Jörg Heitkötter <joke@de.uu.net>.

The Santa Fe Institute

The Santa Fe Institute Studies in the Sciences of Complexity (SFI) issues a recommended series: SFI Studies in the Science of Complexity, published by Addison Wesley and maintains a well-sorted **FTP** server with **EC** related material.

- Admin requests: <ftp@santafe.edu>
- Anonymous FTP archive: **ftp.santafe.edu/pub/**

Information on **SUMMERSCHOOLs** held by the SFI can be obtained from: Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, USA.

The Australian National University (ANU)

The Bioinformatics facility at Australian National University has set up an anonymous **FTP** server, that contains **EC** related material, maintained by David G. Green.

- Admin requests: <david.green@anu.edu.au>
- Anonymous FTP archive: **life.anu.edu.au/pub/complex_systems/alife/**
- World Wide Web: The hypermedia server offers introductory tutorials, preprints and papers online. The URL for this service is **http://complex.csu.edu.au/complex** or link via the servers home page **http://life.anu.edu.au/**

LGI laboratory, Grenoble, France

Research into Parallel **GENETIC ALGORITHMs:** papers (technical reports, conference and journal articles, theses, monographies, etc...) written by members of the SYMPA team are available by **FTP** from

- **imag.fr/pub/SYMPA/**

Their adress is: SYMPA/LGI - Institut IMAG, BP 53 38041 Grenoble Cedex, FRANCE <muntean@imag.fr>

The University of Alabama, Department of Computer Science

A number of papers and preprints are available in compressed Postscript form by **FTP** from the Univ. of Alabama (Tuscaloosa) from **aramis.cs.ua.edu/pub/tech-reports/** The naming convention for files is: (author's last name).(journal name).ps . Maintained by Dr. Ron Sun <rsun@athos.cs.ua.edu>

CMU Artificial Intelligence Repository

Holds more than a gigabyte of software, publications, and other materials of interest to **AI** researchers, educators, students, and practitioners. The AI Programming Languages and the AI Software Packages sections of the repository can be accessed in the **lang/** and **areas/** subdirectories. Other directories, which are in varying states of completion, are **events/** and **pubs/** (Publications, including technical reports, books, mail/news archives).

The AI Programming Languages section includes directories for Common Lisp, Prolog, Scheme, Smalltalk, and other AI-related programming languages. The AI Software Packages section includes subdirectories for: alife/ **(ARTIFICIAL LIFE),** anneal/ (Simulated Annealing), genetic/ **(GENETIC ALGORITHMs** etc., including benchmarks and test problems) and many more.

The AI Repository is accessible by **FTP** at: **ftp.cs.cmu.edu/user/ai/** (Be sure to read the files 0.doc and readme.txt in this directory) and by WWW at: **http://www.cs.cmu.edu/Groups/AI/html/repository.html** It is also available on CD-ROM (See Q10.10).

The MSU Genetic Algorithms Research and Applications Group (GARAGe)

GARAGe has a number of interesting projects, both in terms of **GA** and **GP** fundamental research and in GA/GP applications including: parallelization of GAs/GPs; multiple **POPULATION** topologies and interchange methodologies; scheduling applications, including sponsored research on job-floor scheduling; design applications, including sponsored research on composite material design; configuration applications, particularly physics applications of optimal molecule configurations for particular systems like C60 (buckyballs) and others.

Information on GARAGe research projects is available by WWW at the URL: **http://GARAGe.cps.msu.edu**

School of Cognitive and Computing Sciences, University of Sussex

The Evolutionary and Adaptive Systems Group in COGS does a significant amount of research in the area of **GAs** and Neural Networks and modeling the process of biological development. For purposes of artificial **EVOLUTION,** many at COGS see this as the major issue to be tackled. For general info about the group, consult the WWW server at: **http://www.cogs.susx.ac.uk/lab/adapt/index.html**

The Navy Center for Applied Research in Artificial Intelligence

The Navy Center for Applied Research in **ARTIFICIAL INTELLIGENCE** (NCARAI) is conducting basic research in the analysis of **GAs** and other **EVOLUTIONARY ALGORITHMs.** GAs are being applied to the learning of strategies and behaviors for autonomous vehicles, and for adaptively testing complex systems such as vehicle controllers. You will find description of projects, researchers, and downloadable papers at URL **http://www.aic.nrl.navy.mil/** in addition to other information. The GA-digest and the **GENETIC ALGORITHMs** Archive are maintained at NCARAI. See Q15.1, "Genetic Algorithms Digest", for more information.

Case Western Reserve University

A WWW home page is available for the CWRU Autonomous Agents Research Group at: **http://yuggoth.ces.cwru.edu/**

The group, led by Randall Beer, conducts interdisciplinary research in the departments of Computer Engineering and Science, Biology, Mechanical Engineering, and Systems Engineering. This research includes work in **EVOLUTIONARY ALGORITHMs,** mobile robotics, and computational biology. The aim is to study the mechanisms that can produce adaptive behavior in animals and **ROBOTs.**

Currently available are Postscript versions of a number of our research papers (in particular, those related to mobile robotics, evolving recurrent neural networks, and computational models of development), an HTML version of a paper on computational development which appeared in **ALIFE** IV, and images of the robots used in our research.

Comments to <yamauchi@alpha.ces.cwru.edu>

Genetic Algorithms Group, George Mason University

Members of the research group are working on a variety of projects including **GA** theory, coevolutionary algorithms, decentralized GAs, representation issues, evolutionary microeconomics, the application of GAs to molecular biology, and GA-based machine learning. There is an online publications list that contains links to PostScript copies of many of their published papers. A WWW home page is available at: **http://www.cs.gmu.edu/research/gag/**

The Complexity and Artificial-Life Research Concept

Includes a whole load of information on the topics of complexity, artificial-life, GAs, NNs, cellular automata, nonlinear science, fractals, self-organisation, evolution, and more. Visit: **http://www.calresco.force9.co.uk**

Q15.4: What relevant newsgroups and FAQs are there?

Besides the obvious **comp.ai.genetic** there exist some other newsgroups that sometimes carry **EC** related topics:

- **comp.ai (FAQ** in **news.answers** , **comp.answers**)
- **comp.ai.digest**
- **comp.ai.fuzzy** (FAQ in **news.answers** , **comp.answers**)
- **comp.ai.jair.announce** (FAQ in **news.answers** , **comp.answers**)
- **comp.ai.jair.papers** (PostScript papers of the *Journal of AI Research,* published by Morgan Kaufmann <morgan@unix.sri.com>) [eds note: this is the first journal that's completely published on **USENET** first, and later in paper form; read the jair-faq, that's posted to the announcement group to find out how to submit your papers, get JAIR papers by **FTP,** Gopher or e-mail, etc.]
- **comp.ai.neural-nets** (FAQ in **news.answers** , **comp.answers**)
- **comp.robotics** (FAQ in **news.answers** , **comp.answers**)
- **comp.theory.cell-automata** (FAQ in **http://alife.santafe.edu/alife/topics/cas/ca-faq/ca-faq.html**)
- **comp.theory.dynamic-sys** (no FAQ)

- **comp.theory.self-org-sys** (no FAQ)
- **sci.bio.evolution** (no FAQ as such, but there is an archive of interesting material, accessible via WWW at **http://www.cqs.washington.edu/˜evolution**)
- **sci.math.num-analysis** (some FAQs in **news.answers** , **comp.answers**)
- **sci.op-research** (some FAQs in **news.answers** , **comp.answers**)
- **talk.origins** (discusses origins of life, **EVOLUTION,** etc. FTP repository index at **ics.uci.edu/pub/origins/Index** –– see Q10.7 for more details.)

Q15.5: What about all these Internet Services?

The Internet supports a variety of on-line services, and a number of tools are available to enable people to make good use of these, including: telnet, **FTP,** gopher, veronica, archie, Wide Area Information Servers (WAIS), and the World-Wide Web (WWW).

Information about using Internet is available from a number of sources, many accesible on-line, via email or FTP. For example, the EFF (Electronic Frontier Foundation) publishes two guides for novices on all the Internet has to offer, by Adam Gaffin and Jörg Heitkötter (see below). These are avaiable over the net.

To receive a short guide to using anonymous FTP, send e-mail with the text "help" to <info@sunsite.unc.edu>.

If you dont have FTP access, you can retrieve documents using the FTP-by-email service. The "ftpmail" service is installed on several sites to allow transmission of FTPable files from almost anywhere. To get the PostScript version of this **FAQ** from **ENCORE,** (See Q15.3) for example, send a message to (for example) <ftpmail@decwrl.dec.com> containing the lines:

```
reply <your-own-e-mail-address-here>
connect alife.santafe.edu
get pub/USER-AREA/EC/FAQ/hhgtec.ps.gz
quit
```

where <your-e-mail-address> is e.g. foo@bar.edu

FTPmail sites available are listed below. Use one that is near you for best performance.

(USA) <ftpmail@decwrl.dec.com>
<ftpmail@sunsite.unc.edu>
<bitftp@pucc.princeton.edu>

(Europe) <bitftp@dearn> or to <bitftp@vm.gmd.de>
<ftpmail@ftp.uni-stuttgart.de>
<ftpmail@ftp.inf.tu-dresden.de>
<ftpmail@grasp.insa-lyon.fr>
<bitftp@plearn.edu.pl>
<ftpmail@doc.ic.ak.uk>

Documents from the archive at <rtfm.mit.edu> can be retrieved similarly by sending email to <mail-server@rtfm.mit.edu>, containing a message such as:

```
send usenet/news.answers/index
send usenet/news.answers/ai-faq/genetic/part1
quit
```

References

Kehoe, B.P. (1992) "Zen and the Art of the Internet: A Beginner's Guide to the Internet", 2nd Edition (July). Prentice Hall, Englewood Cliffs, NJ. 112 pages. The 1st Edition, (February) is available in PostScript format via anonymous FTP from **ftp.cs.widener.edu:** and many other Internet archives.

Krol, E. (1992) "The Whole Internet: Catalog & User's Guide". O'Reilly & Associates, Inc., Sebastopol, CA. 376 pages.

LaQuey, T. and J.C. Ryer (1992) "The Internet Companion: A Beginner's Guide to Global Networking". Addison-Wesley Publishing Co., Reading, MA. 208 pages.

Smith, Una R. (1993) "A Biologist's Guide to Internet Resources." **USENET sci.answers** FTP and e-mail from many archives, eg. **rtfm.mit.edu/pub/usenet/sci.answers/biology/guide/part?**

Gaffin, A. (1994) "Everybody's Guide to the Internet." Published by the EFF and MIT Press. $14.95. ISBN 9-780262-67105-7. This book is available in ASCII by sending e-mail to <netguide@eff.org>; you'll receive the book split into several pieces; for a more elaborate version of the guide see the following entry.

Gaffin, A. with Heitkötter, J. (1994) "EFF's (Extended) Guide to the Internet: A round trip through Global Networks, Life in Cyberspace, and Everything...", aka 'eegtti.texi'. This is available from **ftp.eff.org/pub/Net_info/Net_Guide/Other_versions/** (Texinfo, ASCII, HTML, DVI and PostScript). The European edition is kept on **ftp.de.uu.net/pub/books/eff-guide/** ~300 pages. A README file gives more information. The hypertext (HTML) version can be browsed at: **http://www.de.uu.net/books/eegtti/eegtti.html**

The EARN Association (May 1993) "A Guide to Network Resource Tools", available via e-mail from <listserv@EARNCC.bitnet>, by sending the message "get nettools ps" (PostScript) or "get nettools memo" (plain text).

Q20: What EA software packages are available?

This gives a list of all known **EA** software packages available to the public. The list was originally maintained by Nici Schraudolph. In June '93 it was agreed that it would be incorporated into this **FAQ** and the responsibility for maintenance taken over by the FAQ editor.

A copy of most of the packages described below are kept at **ENCORE,** (See Q15.3), available by anonymous **FTP.**

Most **GENETIC PROGRAMMING** software is available by FTP in: **ftp.io.com/pub/genetic-programming/** There are subdirectories containing papers related to **GP,** archives of the mailing list, as well as a suite of programs for implementing GP. These programs include the Lisp code from Koza's "Genetic Programming" [KOZA92], as well as implementations in C and C++, as for example SGPC: Simple Genetic Programming in C by Walter Alden Tackett and Aviram Carmi <gpc@ipld01.hac.com>.

A survey paper entitled "Genetic Algorithm Programming Environments" was published in IEEE Computer in the June 1994 issue. Written by Filho, Alippi and Treleaven of University College, London, UK. It's available by FTP as **bells.cs.ucl.ac.uk/papagena/game/docs/gasurvey.ps** (file size: 421k).

PLEASE NOTE

For many of these software packages, specific ordering instructions are given in the descriptions below (see Q20.1 - Free Software packages, Q20.2 - Commercial Software Packages, Q20.3 - Research Projects). Please read and follow them before unnecessarily bothering the listed author or contact! Also note that these programs haven't been independently tested, so there are no guarantees of their quality.

A major revision was undertaken in August 1994, when all authors were contacted, and asked to confirm the accuracy of the information contained here. A few authors did not respond to the request for information. These are noted below by: (Unverified 8/94). In these cases, **FTP** address were checked by the **FAQ** editor, to confirm that this information (at least) is correct. In two cases, email to the author bounced back as "undeliverable" -- these are noted below.

Legend

Type (this is a very ad-hoc classification)

- GE: generational GA
- SS: steady-state GA
- PA: (pseudo) parallel GA
- ES: evolution strategy
- OO: object-oriented
- XP: expert system
- ED: educational/demo
- CF: classifier system

OS Operating System; X11 implies Unix; "Win" means Microsoft Windows 3.x/NT (PC); "DOS" means MS-DOS or compatibles.

Lang Programming Language; in parentheses: source code not included; "OPas" = MPW Object Pascal

Price (1) free to government contractors, $221 otherwise, (2) educational discount available, (3) available as addendum to a book, (4) single 1850 DM, site license 5200 DM, (5) single 200 DM, site license 500 DM, (6) free for academic and educational use.

Author or Contact
given as Internet e-mail address if possible

ES/GA System Packages:

Name	Type	OS	Lang	Price	Author/Contact
BUGS	GE, ED	X11, Suntools	C	free	*Joshua Smith <jrs@media.mit.edu>*
ComputerAnts	ED, GA	Win	?	free	*Scott Kennedy, Axcelis Inc. <staff@axcelis.com>*
DGenesis	GE, PA,ED	Unix	C	free	*Erick Cantu-Paz <ecantu@lamport.rhon.itam.mx>*
DOUGAL	SS, GE	DOS	Turbo Pascal	free	*Brett Parker <b.s.parker@durham.ac.uk>*
ESCaPaDE	ES	Unix	C	free	*Frank Hoffmeister <hoffmeister@ls11.informatik.uni-dortmund.de>*
Evolution Machine	GE, ES	DOS	C	free	*Hans-Michael Voigt and Joachim Born <voigt@max.fb10.tu-berlin.de>*
Evolutionary Objects	GE OO	Unix	C++	free	*JJ Merelo <jmerelo@kal-el.ugr.es>*
GAC, GAL	GE "	Unix "	C Lisp	free "	*Bill Spears <spears@aic.nrl.navy.mil>*
GAlib	GA	Unix, Mac,DOS	C++	free	*Matthew Wall <mbwall@mit.edu>*
GAGA	GE	Unix	C	free	*Jon Crowcroft <jon@cs.ucl.ac.uk>*
GAGS	GE, SS,OO	Unix, DOS	C++	free	*JJ Merelo <jmerelo@kal-el.ugr.es>*
GALOPPS	GE, PA	Unix, DOS	C	free	*Erik Goodman <goodman@egr.msu.edu>*
GAMusic	ED	Win	(VB)	$10	*Jason H. Moore <jhm@superh.hg.med.umich.edu>*
GANNET	GA, NN	Unix	C	free	*Darrell Duane <dduane@fame.gmu.edu>*
GAucsd	GE	Unix	C	free	*Nici Schraudolph <GAucsd-request@cs.ucsd.edu>*
GA Workbench	GE, ED	DOS	(C++)	free	*Mark Hughes <mrh@i2ltd.demon.co.uk>*
GECO	GE, OO,ED	Unix, MacOS	Lisp	free	*George P. W. Williams, Jr. <george@hsvaic.hv.boeing.com>*
Genesis	GE, ED	Unix, DOS	C	free	*John Grefenstette <gref@aic.nrl.navy.mil>*
GENEsYs	GE	Unix	C	free	*Thomas Bäck <baeck@ls11.informatik.uni-dortmund.de>*
GenET	SS, ES,ED	Unix, X, etc.	C	free	*Cezary Z. Janikow <janikow@radom.umsl.edu>*

Continued over . . .

ES/GA System Packages continued:

Name	Type	OS	Lang	Price	Author/Contact
Genie	GE	Mac	Think Pascal	free	*Lance Chambers <pstamp@yarrow.wt.uwa.edu.au>*
Genitor	SS	Unix	C	free	*Darrell Whitley <whitley@cs.colostate.edu>*
GENlib	SS	Unix, DOS	C	(6)	*Jochen Ruhland <jochenr@neuro.informatik.uni-kassel.de>*
GENOCOP	GE	Unix	C	free	*Zbigniew Michalewicz <zbyszek@uncc.edu>*
GIGA	SS	Unix	C	free	*Joe Culberson <joe@cs.ualberta.ca>*
GPEIST	GP	Win, OS/2	Smalltalk	free	*Tony White <arpw@bnr.ca>*
Imogene	GP	Win	C++	free	*Harley Davis <davis@ilog.fr>*
JAG	GA	–	Java	free	*Stephen Hartley <shartley@mcs.drexel.edu>*
LibGA	GE, SS,ED	Unix/DOS NeXT/Amiga	C	free	*Art Corcoran <corcoran@penguin.mcs.utulsa.edu>*
LICE	ES	Unix, DOS	C	free	*Joachim Sprave <joe@ls11.informatik.uni-dortmund.de>*
Matlab-GA	GE	?	Matlab	free	*Andy Potvin <potvin@mathworks.com>*
mGA	GE	Unix	C, Lisp	free	*Dave Goldberg <goldberg@vmd.cso.uiuc.edu>*
PARAGenesis	PA, GE	CM	C*	free	*Michael van Lent <vanlent@eecs.umich.edu>*
PGA	PA, SS,GE	Unix, etc.	C	free	*Peter Ross <peter@aisb.ed.ac.uk>*
PGAPack	GA, PA	any	C	free	*David Levine <levine@mcs.anl.gov>*
REGAL	GA		C	free	*Filippo Neri <neri@di.unito.it>*
SGA-C, SGA-Cube	GE	Unix nCube	C	free	*Robert E. Smith <rob@comec4.mh.ua.edu>*
Splicer	GE	Mac, X11	C	(1)	*Steve Bayer*
TOLKIEN	OO, GE	Unix, DOS	C++	free	*Anthony Yiu-Cheung Tang <tang028@cs.cuhk.hk>*
Trans-Dimensional Learning	NN	Win	?	free	*<upso@prodigy.com>*
WOLF	SS	Unix	C	free	*David Rogers <drogers@msi.com>*

Classifier System Implementations:

Name	Type	OS	Lang	Price	Author/Contact
CFS-C	CF, ED	Unix/DOS	C	free	*Rick Riolo* *<rlr@merit.edu>*
SCS-C	CF, ED	Unix/DOS Atari TOS	C	free	*Jörg Heitkötter* *<joke@Germany.EU.net>*

Commercial Packages:

Name	Type	OS	Lang	Price	Author/Contact
ActiveGA	GA	Win	(ActiveX)	$99	*Brightwater Software* *<support@brightsoft.com>*
EnGENEer	OO, GA	X11	C	?	*George Robbins, Logica Cambridge Ltd.*
EvoFrame/ REALizer	OO, ES	Mac, DOS	C++/ OPas	(4,2) (5,2)	*Optimum Software* *<optimum@applelink.apple.com>*
Evolver	GE	DOS, Mac	(C, Pascal)	UKP350	*Palisade* *<sales@palisade-europe.com>*
FlexTool	GA	Win	Matlab	?	*Flexible Intelligence Group* *<info@flextool.com>*
GAME	OO, GA	X11	C++	(3)	*Jose R. Filho* *<zeluiz@cs.ucl.ac.uk>*
GeneHunter	GA	Win, Excel	(VB)	$369	*Ward Systems* *<wardsystems@msn.com>*
Generator	GE,SS ES,OO,ED	Win, Excel	(C++)	$379	*Steve McGrew, New Light Industries* *<nli@comtch.iea.com>*
MicroGA/ Galapagos	OO, SS	Mac, Win	C++	$249 (2)	*Emergent Behavior, Inc.* *<emergent@aol.com>*
Omega	?	DOS	?	?	*David Barrow, KiQ Ltd.*
OOGA	OO, GE	Mac, DOS	Lisp	$60	*Lawrence Davis*
PC/Beagle	XP	DOS	?	69UKP	*Richard Forsyth*
XpertRule/ GenAsys	XP	DOS	(Think Pascal)	995UKP	*Attar Software* *<100116.1547@compuserve.com>*
XYpe	SS	Mac	(C)	$725	*Ed Swartz, Virtual Image Inc.*

Q20.1: Free software packages?

BUGS:

BUGS (Better to Use Genetic Systems) is an interactive program for demonstrating the **GENETIC ALGORITHM** and is written in the spirit of Richard Dawkins' celebrated Blind Watchmaker software. The user can play god (or 'GA **FITNESS** function,' more accurately) and try to evolve lifelike organisms (curves). Playing with BUGS is an easy way to get an understanding of how and why the **GA** works. In addition to demonstrating the basic **GENETIC OPERATORs (SELECTION, CROSSOVER,** and **MUTATION),** it allows users to easily see and understand phenomena such as **GENETIC DRIFT** and premature convergence. BUGS is written in C and runs under Suntools and X Windows.

BUGS was written by Joshua Smith <jrs@media.mit.edu> at Williams College and is available from **www.aic.nrl.navy.mil/pub/galist/src/BUGS.tar.Z** Note that it is unsupported software, copyrighted but freely distributable. Address: Room E15-492, MIT Media Lab, 20 Ames Street, Cambridge, MA 02139. (Unverified 8/94).

ComputerAnts:

ComputerAnts is a free Windows program that teaches principles of **GENETIC ALGORITHMs** by breeding a colony of ants on your computer screen. Users create ants, food, poison, and set **CROSSOVER** and **MUTATION** rates. Then they watch the colony slowly evolve. Includes extensive on-line help and tutorials on genetic algorithms. For further information or to download, see **http://www.axcelis.com**

DGenesis:

DGenesis is a distributed implementation of a Parallel **GA.** It is based on Genesis 5.0. It runs on a network of UNIX workstations. It has been tested with DECstations, microVAXes, Sun Workstations and PCs running 386BSD 0.1. Each subpopulation is handled by a UNIX process and the communication between them is accomplished using Berkeley sockets. The system is programmed in C and is available free of charge by anonymous **FTP** from **lamport.rhon.itam.mx:/** and from **ftp.aic.nrl.navy.mil/pub/galist/src/ga/dgenesis-1.0.tar.Z**

DGenesis allows the user to set the **MIGRATION** interval, the migration rate and the topology between the **SUB-POPULATIONs.** There has not been much work investigating the effect of the topology on the **PERFORMANCE** of the GA, DGenesis was written specifically to encourage experimentation in this area. It still needs many refinements, but some may find it useful.

Contact Erick Cantu-Paz <ecantu@lamport.rhon.itam.mx> at the Instituto Tecnologico Autonomo de Mexico (ITAM)

Dougal:

DOUGAL is a demonstration program for solving the **TRAVELLING SALESMAN PROBLEM** using **GAs.** The system guides the user through the GA, allowing them to see the results of altering parameters relating to **CROSSOVER, MUTATION** etc. The system demonstrates graphicaly the **OPTIMIZATION** of the route. The options open to the user to experiment with include percentage CROSSOVER and MUTATION, **POPULATION** size, steady state or generational replacement, **FITNESS** technique (linear normalised, is evaluation, etc).

DOUGAL requires an IBM compatible PC with a VGA monitor. The software is free, however I would appreciate feedback on what you think of the software.

Dougal is available by **FTP** from **ENCORE** (see Q15.3) in file **EC/GA/src/dougal.zip** It's pkzipped and contains executable, vga driver, source code and full documentation. It is important to place the vga driver (egavga.bgi) in the same directory as DOUGAL. Author: Brett Parker, 7 Glencourse, East Boldon, Tyne + Wear, NE36 0LW, England. <b.s.parker@durham.ac.uk>

ESCaPaDE:

ESCaPaDE is a sophisticated software environment to run experiments with **EVOLUTIONARY ALGORITHMs,** such as e.g. an **EVOLUTION STRATEGY.** The main support for experimental work is provided by two internal tables: (1) a table of objective functions and (2) a table of so-called data monitors, which allow easy implementation of functions for monitoring all types of information inside the Evolutionary Algorithm under experiment.

ESCaPaDE 1.2 comes with the KORR implementation of the evolution strategy by H.-P. Schwefel which offers simple and correlated **MUTATIONs.** KORR is provided as a FORTRAN 77 subroutine, and its cross-compiled C version is used internally by ESCaPaDE.

An extended version of the package was used for several investigations so far and has proven to be very reliable. The software and its documentation is fully copyrighted although it may be freely used for scientific work; it requires 5-6 MB of disk space.

In order to obtain ESCaPaDE, please send a message to the e-mail address below. The SUBJECT line should contain 'help' or 'get ESCaPaDE'. (If the subject lines is invalid, your mail will be ignored!). For more information contact: Frank Hoffmeister, Systems Analysis Research Group, LSXI, Department of Computer Science, University of Dortmund, D-44221 Dortmund, Germany. Net: <hoffmeister@ls11.informatik.uni-dortmund.de>

Evolution Machine:

The Evolution Machine (EM) is universally applicable to continuous (real-coded) **OPTIMIZATION** problems. In the EM we have coded fundamental **EVOLUTIONARY ALGORITHMs (GENETIC ALGORITHMs** and **EVOLUTION STRATEGIEs),** and added some of our approaches to evolutionary search.

The EM includes extensive menu techniques with:

- Default parameter setting for unexperienced users.
- Well-defined entries for EM-control by freaks of the EM, who want to leave the standard process control.
- Data processing for repeated runs (with or without change of the strategy parameters).
- Graphical presentation of results: online presentation of the **EVOLUTION** progress, one-, two- and three-dimensional graphic output to analyse the **FITNESS** function and the evolution process.
- Integration of calling MS−DOS utilities (Turbo C).

We provide the EM-software in object code, which can be run on PC's with MS−DOS and Turbo C, v2.0, resp. Turbo C++,v1.01. The Manual to the EM is included in the distribution kit.

The EM software is available by **FTP** from **ftp-bionik.fb10.tu-berlin.de/pub/software/Evolution-Machine/** This directory contains the compressed files em_tc.exe (Turbo C), em_tcp.exe (Turbo C++) and em_man.exe (the manual). There is also em-man.ps.Z, a compressed PostScript file of the manual. If you do not have FTP access, please send us either 5 1/4 or 3 1/2 MS-DOS compatible disks. We will return them with the compressed files (834 kB).

Official contact information: Hans-Michael Voigt or Joachim Born, Technical University Berlin, Bionics and evolution Techniques Laboratory, Bio- and Neuroinformatics Research Group, Ackerstrasse 71-76 (ACK1), D-13355 Berlin, Germany. Net: <voigt@fb10.tu-berlin.de>, <born@fb10.tu-berlin.de> (Unverified 8/94).

EVOLUTIONARY OBJECTS:

EO (Evolutionary Objects) is a C++ library written and designed to allow a variety of evolutionary algorithms to be constructed easily. It is intended to be an "Open source" effort to create the definitive EC library. It has: a mailing list, anon-CVS access, frequent snapshots and other features. For details, see **http://fast.to/EO**

Maintained by J.J. Merelo, Grupo Geneura, Univ. Granada <jmerelo@kal-el.ugr.es>

GA Workbench:

A mouse-driven interactive **GA** demonstration program aimed at people wishing to show GAs in action on simple **FUNCTION OPTIMIZATIONs** and to help newcomers understand how GAs operate. Features: problem functions drawn on screen using mouse, run-time plots of GA **POPULATION** distribution, peak and average **FITNESS.** Useful population **STATISTICS** displayed numerically, GA configuration (population size, **GENERATION** gap etc.) performed interactively with mouse. Requirements: MS-DOS PC, mouse, EGA/VGA display.

Available by **FTP** from the simtel20 archive mirrors, e.g. **wsmr-simtel20.army.mil/pub/msdos/neurlnet/gaw110.zip** or **wuarchive.wustl.edu:** or **oak.oakland.edu:** Produced by Mark Hughes <mrh@i2ltd.demon.co.uk>. A windows version is in preparation.

GAC, GAL:

Bill Spears <spears@aic.nrl.navy.mil> writes: These are packages I've been using for a few years. GAC is a **GA** written in C. GAL is my Common Lisp version. They are similar in spirit to John Grefenstette's Genesis, but they don't have all the nice bells and whistles. Both versions currently run on Sun workstations. If you have something else, you might need to do a little modification.

Both versions are free: All I ask is that I be credited when it is appropriate. Also, I would appreciate hearing about improvements! This software is the property of the US Department of the Navy.

The code will be in a "shar" format that will be easy to install. This code is "as is", however. There is a README and some documentation in the code. There is NO user's guide, though (nor am I planning on writing one at this time). I am interested in hearing about bugs, but I may not get around to fixing them for a while. Also, I will be unable to answer many questions about the code, or about GAs in general. This is not due to a lack of interest, but due to a lack of free time!

Available by **FTP** from **ftp.aic.nrl.navy.mil/pub/galist/src/ga/GAC.shar.Z** and GAL.shar.Z . PostScript versions of some papers are under "/pub/spears". Feel free to browse.

GAGA:

GAGA (**GA** for General Application) is a self-contained, re-entrant procedure which is suitable for the minimization of many "difficult" cost functions. Originally written in Pascal by Ian Poole, it was rewritten in C by Jon Crowcroft. GAGA can be obtained by request from the author: Jon Crowcroft <jon@cs.ucl.ac.uk>, Univeristy College London, Gower Street, London WCIE 6BT, UK, or by **FTP** from **ftp://cs.ucl.ac.uk/darpa/gaga.shar**

GAGS:

GAGS (Genetic Algorithms from Granada, Spain) is a library and companion programs written and designed to take the heat out of designing a **GENETIC ALGORITHM.** It features a class library for genetic algorithm programming, but, from the user point of view, is a genetic algorithm application generator. Just write the function you want to optimize, and GAGS surrounds it with enough code to have a genetic algorithm up and running, compiles it, and runs it. GAGS Is written in C++, so that it can be compiled in any platform running this GNU utility. It has been tested on various machines. Documentation is available.

GAGS includes:

- Steady-state, roulette-wheel, tournament and elitist **SELECTION.**
- **FITNESS** evaluation using training files.
- Graphics output through gnuplot.
- Uniform and 2-point **CROSSOVER,** and bit-flip and gene-transposition **MUTATION.**
- Variable length **CHROMOSOMEs** and related operators.

The application generator gags.pl is written in perl, so this language must also be installed before GAGS. Available from: **http://kal-el.ugr.es/GAGS** The programmer's manual is in the file **gagsprogs.ps.gz.** GAGS is also available from **ENCORE** (see Q15.3) in file **EC/GA/src/gags-0.92.tar.gz** (there may be a more recent version) with documentation in **EC/GA/docs/gagsprog.ps.gz**

Maintained by J.J. Merelo, Grupo Geneura, Univ. Granada <jmerelo@kal-el.ugr.es>

GAlib:

GAlib is a C++ library that provides the application programmer with a set of **GENETIC ALGORITHM** objects. With GAlib you can add **GA OPTIMIZATION** to your program using any data representation and standard or custom **SELECTION, CROSSOVER, MUTATION,** scaling, and replacement, and termination methods. View the documentation on-line at **http://lancet.mit.edu/ga/** There you will find a complete description of the programming interface, features, and examples.

The canonical source for this library is the **FTP** site: **lancet.mit.edu/pub/ga/** This directory contains UNIX (.tar.gz), MacOS (.sea.hqx), and DOS (.zip) versions of the GA library. Once you have downloaded the file, uncompress and extract it. It will expand to its own directory. If you extract the DOS version be sure to use the -d option to keep everything in one directory.

GAlib requires a cfront 3.0 compatible C++ compiler. It has been used on the following systems: SGI IRIX 4.0.x (Cfront); SGI IRIX 5.x (DCC 1.0, g++ 2.6.8, 2.7.0); IBM RSAIX 3.2 (g++ 2.6.8, 2.7.0); DEC MIPS ultrix 4.2 (g++ 2.6.8, 2.7.0); SUN SOLARIS 5.3 (g++ 2.6.8, 2.7.0); HP-UX (g++); MacOS (MetroWerks CodeWarrior 5); MacOS (Symantec THINK C++ 7.0); DOS/Windows (Borland Turbo C++ 3.0).

Maintained by: Matthew Wall <mbwall@mit.edu>

GALOPPS:

GALOPPS (Genetic Algorithm Optimized for Portability and Parallelism) is a general-purpose parallel **GENETIC ALGORITHM** system, written in 'C', organized like Goldberg's "Simple Genetic Algorithm". User defines objective function (in template furnished) and any callback functions desired (again, filling in template); can run one or many subpopulations, on one or many PC's, workstations, Mac's, MPP. Runs interactively (GUI or answering questions) or from files, makes file and/or graphical output. Runs easily interrupted and restarted, and a PVM version for Unix networks even moves processes automatically when workstations become busy. (Note: optional GUI requires Tcl/Tk.) 14 example problems included (De Jong Functions, Royal Road, BTSP, etc.)

User may choose:

- problem type (permutation or value-type)
- field sizes (arbitrary, possibly unequal, heeded by **CROSSOVER, MUTATION)**
- among 7 crossover types and 4 mutation types (or define own)
- among 6 **SELECTION** types, including "automatic" option based on Boltzmann scaling and Shapiro and Pruegel-Bennett statist. Mechanics stuff
- operator probabilities, **FITNESS** scaling, amount of output, **MIGRATION** frequency and patterns,
- stopping criteria (using "standard" convergence **STATISTICS,** etc.)
- the GGA (Grouping Genetic Algorithm) **REPRODUCTION** and operators of Falkenauer

GALOPPS allows and supports:

- use of a different representation in each subpopulation, with transformation of migrants
- **INVERSION** on level of subpopulations, with automatic handling of differing field sizes, migrants
- control over replacement by **OFFSPRING,** including DeJong crowding or random replacement or SGA-like replacement of **PARENTs**
- mate selection, using incest reduction
- migrant selection, using incest reduction, and/or DeJong crowding into receiving subpopulation
- optional **ELITISM**

Generic (Unix) GALOPPS 3.2 (includes 80-pp. manual) is available on **ENCORE.** For PVM GALOPPS, PC version (different line endings, makefiles), Threaded GALOPPS, and GALOPPS-based 2-level adaptive system, see the MSU GARAGe web site: **http://GARAGe.cps.msu.edu/** .

Contact: Erik D. Goodman, <goodman@egr.msu.edu>, MSU GARAGe, Case Center, 112 Engineering Building, MSU, East Lansing, MI 48824 USA.

GAMusic:

GAMusic 1.0 is a user-friendly interactive demonstration of a simple **GA** that evolves musical melodies. Here, the user is the **FITNESS** function. Melodies from the **POPULATION** can be played and then assigned a fitness. Iteration, **RECOMBINATION** frequency and **MUTATION** frequency are all controlled by the user. This program is intended to provide an introduction to GAs and may not be of interest to the experienced GA programmer.

GAMusic was programmed with Microsoft Visual Basic 3.0 for Windows 3.1x. No special sound card is required. GAMusic is distributed as shareware (cost $10) and can be obtained by **FTP** from **wuarchive.wustl.edu/pub/MSDOS_UPLOADS/GenAlgs/gamusic.zip** or from **fly.bio.indiana.edu/science/ibmpc/gamusic.zip** The program is also available from the America Online archive.

Contact: Jason H. Moore <jhm@superh.hg.med.umich.edu> or <jasonUMICH@aol.com>

GANNET:

GANNET (Genetic Algorithm / Neural NETwork) is a software package written by Jason Spofford in 1990 which allows one to evolve binary valued neural networks. It offers a variety of configuration options related to rates of the **GENETIC OPERATORs.** GANNET evolves nets based upon three **FITNESS** functions: Input/Output Accuracy, Output 'Stability', and Network Size.

The evolved neural network presently has a binary input and binary output format, with neurodes that have either 2 or 4 inputs and weights ranging from -3 to +4. GANNET allows for up to 250 neurons in a net. Research using GANNET is continuing.

GANNET 2.0 is available at **http://fame.gmu.edu/~dduane/thesis**
. As well as the software, the masters thesis that utilized this program as well as a paper is available in this directory. See also **fame.gmu.edu/gannet/source/**

The major enhancement of version 2.0 is the ability to recognize variable length binary strings, such as those that would be generated by a finite automaton. Included is code for calculating the Effective Measure Complexity (EMC) of finite automata as well as code for generating test data.

A mailing list has been established for discussing uses and problems with the GANNET software. To subscribe, send a message to: <listproc@gmu.edu> On the first line of the message (not the subject) type: *SUB GANNET Your-First-Name Your-Last-Name*

Contact: Darrell Duane or Dr. Kenneth Hintz, George Mason University, Dept. of Electrical & Computer Engineering, Mail Stop 1G5, 4400 University Drive, Fairfax, VA 22033-4444 USA. Net: <dduane@fame.gmu.edu> or <khintz@fame.gmu.edu>

GAucsd:

GAucsd is a Genesis-based **GA** package incorporating numerous bug fixes and user interface improvements. Major additions include a wrapper that simplifies the writing of evaluation functions, a facility to distribute experiments over networks of machines, and Dynamic Parameter Encoding, a technique that improves GA **PERFORMANCE** in continuous **SEARCH SPACEs** by adaptively refining the genomic representation of real-valued parameters.

GAucsd was written in C for Unix systems, but the central GA engine is easily ported to other platforms. The entire package can be ported to systems where implementations of the Unix utilities "make", "awk" and "sh" are available.

GAucsd is available by **FTP** from **cs.ucsd.edu/pub/GAucsd/GAucsd14.sh.Z** or from **ftp.aic.nrl.navy.mil/pub/galist/src/ga/GAucsd14.sh.Z** To be added to a mailing list for bug reports, patches and updates, send "add GAucsd" to <listserv@cs.ucsd.edu>.

Cognitive Computer Science Research Group, CSE Department, UCSD 0114, La Jolla, CA 92093-0114, USA. Net: <GAucsd-request@cs.ucsd.edu>

GECO:

GECO (Genetic Evolution through Combination of Objects) is an extensible, object-oriented framework for prototyping **GENETIC ALGORITHMs** in Common Lisp. GECO makes extensive use of CLOS, the Common Lisp Object System, to implement its functionality. The abstractions provided by the classes have been chosen with the intent both of being easily understandable to anyone familiar with the paradigm of genetic algorithms, and of providing the algorithm developer with the ability to customize all aspects of its operation. It comes with extensive documentation, in the form of a PostScript file, and some simple examples are also provided to illustrate its intended use.

GECO Version 2.0 is available by **FTP.** See the file **ftp.aic.nrl.navy.mil/pub/galist/src/ga/GECO-v2.0.README** for more information.

George P. W. Williams, Jr., 1334 Columbus City Rd., Scottsboro, AL 35768. Net: <george@hsvaic.hv.boeing.com>.

Genesis:

Genesis is a generational **GA** system written in C by John Grefenstette. As the first widely available GA program Genesis has been very influential in stimulating the use of GAs, and several other GA packages are based on it. Genesis is available together with OOGA (see below), or by **FTP** from **ftp.aic.nrl.navy.mil/pub/galist/src/genesis.tar.Z** (Unverified 8/94).

GENEsYs:

GENEsYs is a Genesis-based **GA** implementation which includes extensions and new features for experimental purposes, such as **SELECTION** schemes like linear ranking, Boltzmann, (mu, lambda)-selection, and general extinctive selection variants, **CROSSOVER** operators like n-point and uniform crossover as well as discrete and intermediate **RECOMBINATION. SELF-ADAPTATION** of **MUTATION** rates is also possible.

A set of objective functions is provided, including De Jong's functions, complicated continuous functions, a TSP-problem, binary functions, and a fractal function. There are also additional data-monitoring facilities such as recording average, variance and skew of **OBJECT VARIABLES** and mutation rates, or creating bitmap-dumps of the **POPULATION.**

GENEsYs 1.0 is available via **FTP** from **lumpi.informatik.uni-dortmund.de/pub/GA/src/GENEsYs-1.0.tar.Z** The documentation alone is available as **/pub/GA/docs/GENEsYs-1.0-doc.tar.Z**

For more information contact: Thomas Bäck, Systems Analysis Research Group, LSXI, Department of Computer Science, University of Dortmund, D-44221 Dortmund, Germany. Net: <baeck@ls11.informatik.uni-dortmund.de> (Unverified 8/94).

GenET:

GenET is a "generic" **GA** package. It is generic in the sense that all problem independent mechanisms have been implemented and can be used regardless of application domain. Using the package forces (or allows, however you look at it) concentration on the problem: you have to suggest the best representation, and the best operators for such space that utilize your problem-specific knowledge. You do not have to think about possible GA models or their implementation.

The package, in addition to allowing for fast implementation of applications and being a natural tool for comparing different models and strategies, is intended to become a depository of representations and operators. Currently, only floating point representation is implemented in the library with few operators.

The algorithm provides a wide selection of models and choices. For example, **POPULATION** models range from generational GA, through steady-state, to (n,m)-EP and (n,n+m)-EP models (for arbitrary problems, not just parameter **OPTIMIZATION).** (Some are not finished at the moment). Choices include automatic adaptation of operator probabilities and a dynamic ranking mechanism, etc.

Even though the implementation is far from optimal, it is quite efficient - implemented in ATT's C++ (3.0) (functional design) and also tested on gcc. Along with the package you will get two examples. They illustrate how to implement problems with heterogeneous and homogeneous structures, with explicit rep/opers and how to use the existing library (FP). Very soon I will place there another example - our GENOCOP operators for linearly constrained optimization. One more example soon to appear illustrates how to deal with complex structures and non-stationary problems - this is a fuzzy rule-based controller optimized using the package and some specific rep/operators.

If you start using the package, please send evaluations (especially bugs) and suggestions for future versions to the author.

GenET Version 1.00 is available by **FTP** from **radom.umsl.edu/var/ftp/GenET.tar.Z** To learn more, you may get the User's Manual, available in compressed postscript in "/var/ftp/userMan.ps.Z". It also comes bundled with the complete package.

Cezary Z. Janikow, Department of Math and CS, CCB319, St. Louis, MO 63121, USA. Net: <janikow@radom.umsl.edu>

Genie:

Genie is a GA-based modeling/forecasting system that is used for long-term planning. One can construct a model of an **ENVIRONMENT** and then view the forecasts of how that environment will evolve into the future. It is then possible to alter the future picture of the environment so as to construct a picture of a desired future (I will not enter into arguments of who is or should be responsible for designing a desired or better future). The **GA** is then employed to suggest changes to the existing environment so as to cause the desired future to come about.

Genie is available free of charge via e-mail or on 3.5'' disk from: Lance Chambers, Department of Transport, 136 Stirling Hwy, Nedlands, West Australia 6007. Net: <pstamp@yarrow.wt.uwa.edu.au> It is also available by **FTP** from **hiplab.newcastle.edu.au/pub/Genie&Code.sea.Hqx**

Genitor:

"Genitor is a modular **GA** package containing examples for floating-point, integer, and binary representations. Its features include many sequencing operators as well as subpopulation modeling.

The Genitor Package has code for several order based **CROSSOVER** operators, as well as example code for doing some small **TSPs** to optimality.

We are planning to release a new and improved Genitor Package this summer (1993), but it will mainly be additions to the current package that will include parallel island models, cellular GAs, delta coding, perhaps CHC (depending on the legal issues) and some other things we have found useful."

Genitor is available from Colorado State University Computer Science Department by **FTP** from **ftp.cs.colostate.edu/pub/GENITOR.tar**

Please direct all comments and questions to <mathiask@cs.colostate.edu>. If these fail to work, contact: L. Darrell Whitley, Dept. of Computer Science, Colorado State University, Fort Collins, CO 80523, USA. Net: <whitley@cs.colostate.edu> (Unverified 8/94).

GENlib:

GENlib is a library of functions for **GENETIC ALGORITHMs.** Included are two applications of this library to the field of neural networks. The first one called "cosine" uses a genetic algorithm to train a simple three layer feed-Forward network to work as a cosine-function. This task is very difficult to train for a backprop algorithm while the genetic algorithm produces good results. The second one called "vartop" is developing a Neural Network to perform the XOR-function. This is done with two genetic algorithms, the first one develops the topology of the network, the second one adjusts the weights.

GENlib may be obtained by **FTP** from **ftp.neuro.informatik.uni-kassel.de/pub/NeuralNets/GA-and-NN/**

Author: Jochen Ruhland, FG Neuronale Netzwerke / Uni Kassel, Heinrich-Plett-Str. 40, D-34132 Kassel, Germany. <jochenr@neuro.informatik.uni-kassel.de>

GENOCOP:

This is a GA-based **OPTIMIZATION** package that has been developed by Zbigniew Michalewicz and is described in detail in his book *Genetic Algorithms + Data Structures = Evolution Programs* [MICHALE94].

GENOCOP (Genetic Algorithm for Numerical Optimization for COnstrained Problems) optimizes a function with any number of linear constraints (equalities and inequalities).

The second version of the system is available by **FTP** from **ftp.uncc.edu/coe/evol/genocop2.tar.Z**

Zbigniew Michalewicz, Dept. of Computer Science, University of North Carolina, Chappel-Hill, NC, USA. Net: <zbyszek@uncc.edu>

GIGA:

GIGA is designed to propogate information through a **POPULATION,** using **CROSSOVER** as its operator. A discussion of how it propogates **BUILDING BLOCKs,** similar to those found in Royal Road functions by John Holland, is given in the **DECEPTION** section of: "Genetic Invariance: A New Paradigm for Genetic Algorithm Design." University of Alberta Technical Report TR92-02, June 1992. See also: "GIGA Program Description and Operation" University of Alberta Computing Science Technical Report TR92-06, June 1992

These can be obtained, along with the program, by **FTP** from **ftp.cs.ualberta.ca/pub/TechReports/** in the subdirectories TR92-02/ and TR92-06/ .

Also, the paper "Mutation-Crossover Isomorphisms and the Construction of Discriminating Functions" gives a more in-depth look at the behavior of GIGA. Its is available from **ftp.cs.ualberta.ca/pub/joe/Preprints/xoveriso.ps.Z**

Joe Culberson, Department of Computer Science, University of Alberta, CA. Net: <joe@cs.ualberta.ca>

GPEIST:

The **GENETIC PROGRAMMING ENVIRONMENT** in Smalltalk (GPEIST) provides a framework for the investigation of Genetic Programming within a ParcPlace VisualWorks 2.0 development system. GPEIST provides program, **POPULATION,** chart and report browsers and can be run on HP/Sun/PC (OS/2 and Windows) machines. It is possible to distribute the experiment across several workstations - with subpopulation exchange at intervals - in this release 4.0a. Experiments, populations and **INDIVIDUAL** genetic programs can be saved to disk for subsequent analysis and experimental statistical measures exchanged with spreadsheets. Postscript printing of charts, programs and animations is supported. An implementation of the Ant Trail problem is provided as an example of the use of the GPEIST environment.

GPEIST is available from **ENCORE** (see Q15.3) in file: **EC/GP/src/GPEIST4.tar.gz**

Contact: Tony White, Bell-Northern Research Ltd., Computer Research Lab - Gateway, 320 March Road, Suite 400, Kanata, Ontario, Canada, K2K 2E3. tel: (613) 765-4279 <arpw@bnr.ca>

Imogene:

Imogene is a Windows 3.1 shareware program which generates pretty images using **GENETIC PROGRAMMING.** The program displays **GENERATIONs** of 9 images, each generated using a formula applied to each pixel. (The formulae are initially randomly computed). You can then select those images you prefer. In the next generation, the nine images are generated by combining and mutating the formulae for the most-preferred images in the previous generation. The result is a **SIMULATION** of natural **SELECTION** in which images evolve toward your aesthetic preferences.

Imogene supports different color maps, palette animation, saving images to .BMP files, changing the wallpaper to nice images, printing images, and several other features. Imogene works only in 256 color mode and requires a floating point coprocessor and a 386 or better CPU.

Imogene is based on work originally done by Karl Sims at (ex-)Thinking Machines for the CM-2 massively parallel computer - but you can use it on your PC. You can get Imogene from: **http://www.aracnet.com/˜wwir/software.html**

Contact: Harley Davis, ILOG S.A., 2 Avenue Gallini, BP 85, 94253 Gentilly Cedex, France. tel: +33 1 46 63 66 66 <davis@ilog.fr>

JAG:

This Java program implements a simple **GENETIC ALGORITHM** where the **FITNESS** function takes non-negative values only. It employs **ELITISM.** The Java code was derived from the C code in the Appendix of *Genetic Algorithms + Data Structures = Evolution Programs,* [MICHALE94]. Other ideas and code were drawn from GAC by Bill Spears.

Four sample problems are contained in the code: three with bit **GENEs** and one with double genes. To use this program, modify the class MyChromosome to include your problem, which you have coded in some class, say YourChromosome. All changes to the sGA.java file to run your problem are confined to your class YourChromosome. This is what object-oriented programming is all about! The sGA.java source code file has a big comment at the end containing some sample runs.

Available by **FTP** from **ftp.mcs.drexel.edu/pub/shartley/simpleGA.tar.gz** . Further information from Stephen J. Hartley <shartley@mcs.drexel.edu>, **http://www.mcs.drexel.edu/˜shartley** . Drexel University, Math and Computer Science Department Philadelphia, PA 19104 USA. +1-215-895-2678

LibGA:

LibGA is a library of routines written in C for developing **GENETIC ALGORITHMs.** It is fairly simple to use, with many knobs to turn. Most **GA** parameters can be set or changed via a configuration file, with no need to recompile. (E.g., operators, pool size and even the data type used in the **CHROMOSOME** can be changed in the configuration file.) Function pointers are used for the **GENETIC OPERATORs,** so they can easily be manipulated on the fly. Several genetic operators are supplied and it is easy to add more. LibGA runs on many systems/architectures. These include Unix, DOS, NeXT, and Amiga.

LibGA Version 1.00 is available by **FTP** from **ftp.aic.nrl.navy.mil/pub/galist/src/ga/libga100.tar.Z** or by email request to its author, Art Corcoran <corcoran@penguin.mcs.utulsa.edu> (Unverified 8/94).

LICE:

LICE is a parameter **OPTIMIZATION** program based on **EVOLUTION STRATEGIEs (ES).** In contrast to classic ES, LICE has a local **SELECTION** scheme to prevent premature stagnation. Details and results were presented at the EP'94 conference in San Diego. LICE is written in ANSI-C (more or less), and has been tested on Sparc-stations and Linux-PCs. If you want plots and graphics, you need X11 and gnuplot. If you want a nice user interface to create parameter files, you also need Tk/Tcl.

LICE-1.0 is available as source code by **FTP** from **lumpi.informatik.uni-dortmund.de/pub/ES/src/LICE-1.0.tar.gz**

Author: Joachim Sprave <joe@ls11.informatik.uni-dortmund.de>

Matlab-GA:

The MathWorks **FTP** site has some Matlab **GA** code in the directory **ftp.mathworks.com/pub/contrib/v4/optim/genetic** It's a bunch of .m files that implement a basic GA. Contact: Andrew Potvin, <potvin@mathworks.com> for information.

mGA:

mGA is an implementation of a messy **GA** as described in TCGA report No. 90004. Messy GAs overcome the linkage problem of simple **GENETIC ALGORITHMs** by combining variable-length strings, **GENE** expression, messy operators, and a nonhomogeneous phasing of evolutionary processing. Results on a number of difficult deceptive test functions have been encouraging with the messy GA always finding global optima in a polynomial number of function evaluations.

See TCGA reports 89003, 90005, 90006, and 91004, and IlliGAL report 91008 for more information on messy GAs (See Q14). The C language version is available by **FTP** from IlliGAL in the directory **gal4.ge.uiuc.edu/pub/src/messyGA/C/**

PARAGenesis:

PARAGenesis is the result of a project implementing Genesis on the CM-200 in C*. It is an attempt to improve **PERFORMANCE** as much as possible without changing the behavior of the **GENETIC ALGORITHM.** Unlike the punctuated equilibria and local **SELECTION** models, PARAGenesis doesn't modify the genetic algorithm to be more parallelizable as these modifications can drastically alter the behavior of the algorithm. Instead each member is placed on a separate processor allowing initialization, evaluation and **MUTATION** to be completely parallel. The costs of global control and communication in selection and **CROSSOVER** are present but minimized as much as possible. In general PARAGenesis on an 8k CM-200 seems to run 10-100 times faster than Genesis on a Sparc 2 and finds equivalent solutions.

PARAGenesis includes all the features of serial Genesis plus some additions. The additions include the ability to collect timing **STATISTICS,** probabilistic selection (as opposed to Baker's stochastic universal sampling), uniform crossover and local or neighborhood selection. Anyone familiar with the serial implementation of Genesis and C* should have little problem using PARAGenesis.

PARAGenesis is available by **FTP** from **ftp.aic.nrl.navy.mil/pub/galist/src/ga/paragenesis.tar.Z**

DISCLAIMER: PARAGenesis is fairly untested at this point and may contain some bugs.

Michael van Lent, Advanced Technology Lab, University of Michigan, 1101 Beal Av., Ann Arbor, MI 48109, USA. Net: <vanlent@eecs.umich.edu>.

PGA:

PGA is a simple testbed for basic explorations in **GENETIC ALGORITHMs.** Command line arguments control a range of parameters, there are a number of built-in problems for the **GA** to solve. The current set includes:

- maximize the number of bits set in a **CHROMOSOME**
- De Jong's functions DJ1, DJ2, DJ3, DJ5
- binary F6, used by Schaffer et al
- a crude 1-d knapsack problem; you specify a target and a set of numbers in an external file, GA tries to find a subset that sums as closely as possible to the target
- the 'royal road' function(s); a chromosome is regarded as a set of consecutive blocks of size K, and scores K for each block entirely filled with 1s, etc; a range of parameters.
- max contiguous bits, you choose the **ALLELE** range.
- timetabling, with various smart **MUTATION** options; capable of solving a good many real-world timetabling problems (has done so)

Lots of GA options: rank, roulette, tournament, marriage-tournament, spatially-structured **SELECTION;** one-point, two-point, uniform or no **CROSSOVER;** fixed or adaptive mutation; one child or two; etc.

Default output is curses-based, with optional output to file; can be run non-interactively too for batched series of experiments.

It's easy to add your own problems. Chromosomes are represented as character arrays, so you are not (quite) stuck with bit-string problem encodings.

PGA has been used for teaching for a couple of years now, and has been used as a starting point by a fair number of people for their own projects. So it's reasonably reliable. However, if you find bugs, or have useful contributions to make, Tell Me! It is available by **FTP** from **ftp.dai.ed.ac.uk/pub/pga/pga-3.1.tar.gz** (see the file pga.README in the same directory for more information)

Peter Ross, Department of AI, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, UK. Net: <peter@aisb.ed.ac.uk>

PGAPack:

PGAPack is a general-purpose, data-structure-neutral parallel **GENETIC ALGORITHM** library. It is intended to provide most capabilities desired in a genetic algorithm library, in an integrated, seamless, and portable manner.

Features include:

- Callable from Fortran or C.
- Runs on uniprocessors, parallel computers, and workstation networks.
- Binary-, integer-, and real- and character-valued native data types

- Full extensibility to support custom operators and new data types.
- Easy-to-use interface for novice and application users.
- Multiple levels of access for expert users.
- Extensive debugging facilities.
- Large set of example problems.
- Detailed users guide
- Parameterized **POPULATION** replacement.
- Multiple choices for **SELECTION, CROSSOVER,** and **MUTATION** operators
- Easy integration of hill-climbing heuristics.

Availability: PGAPack is freely available and may be obtained by **FTP** from **info.mcs.anl.gov/pub/pgapack/pgapack.tar.Z** or from **http://www.mcs.anl.gov/pgapack.html**

Further Information from David Levine, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois 60439, (708)-252-6735 <levine@mcs.anl.gov> **http://www.mcs.anl.gov/home/levine**

REGAL:

REGAL (RElational Genetic Algorithm Learner) is a distributed GA-based system, designed for learning multi-modal First Order Logic concept descriptions from examples. REGAL is based on a **SELECTION** operator, called Universal Suffrage operator, provably allowing the **POPULATION** to asymptotically converge, on average, to an equilibrium state, in which several **SPECIES** coexist. REGAL makes use of PVM 3.3 and Tcl/Tk. This version of REGAL is provided with a graphical user interface developed in Tcl/Tk language.

REGAL has been jointly developed by: Attilio Giordana <attilio@di.unito.it> **http://www.di.unito.it/~attilio/** and Filippo Neri <neri@di.unito.it> **http://www.di.unito.it/~neri/** at the University of Torino, Dipartimento di Informatica, Italy.

See also:

Neri F. and Giordana A. (1995). "A Distributed Genetic Algorithm for Concept Learning", Proc. Int. Conf. on Genetic Algorithms (Pittsburgh, PA), Morgan Kaufmann, pp. 436-443.

Neri F. and Saitta L. (1995). "A Formal Analysis of Selection Schemes". Proc. Int. Conf. on Genetic Algorithms (Pittsburgh,PA), Morgan Kaufmann, pp. 32-39 .

Giordana A. and Neri F. (1996). "Search-Intensive Concept Induction". Evolutionary Computation Journal, MIT Press, vol. 3, n. 4, pp. 375 - 416.

Neri F. and Saitta L. (1997). "An Analysis of the Universal Suffrage Selection Operator". Evolutionary Computation Journal, MIT Press, vol. 4, n. 1, pp. 89-109.

SGA-C, SGA-Cube:

SGA-C is a C-language translation and extension of the original Pascal SGA code presented in Goldberg's book [GOLD89]. It has some additional features, but its operation is essentially the same as that of the Pascal version. SGA-C is described in TCGA report No. 91002.

SGA-Cube is a C-language translation of Goldberg's SGA code with modifications to allow execution on the nCUBE 2 Hypercube Parallel Computer. When run on the nCUBE 2, SGA-Cube can take advantage of the hypercube architecture, and is scalable to any hypercube dimension. The hypercube implementation is modular, so that the algorithm for exploiting parallel processors can be easily modified.

In addition to its parallel capabilities, SGA-Cube can be compiled on various serial computers via compile-time options. In fact, when compiled on a serial computer, SGA-Cube is essentially identical to SGA-C. SGA-Cube is described in TCGA report No. 91005.

Each of these programs is distributed in the form of a Unix shar file, available via e-mail or on various formatted media by request from: Robert Elliott Smith, Department of Engineering of Mechanics, Room 210 Hardaway Hall,, The University of Alabama P.O. Box 870278, Tuscaloosa, Alabama 35487, USA. Net: <rob@comec4.mh.ua.edu>

SGA-C and SGA-Cube are also available in compressed tar form by **FTP** from **ftp.aic.nrl.navy.mil/pub/galist/src/ga/sga-c.tar.Z** and sga-cube.tar.Z .

Splicer:

Splicer is a **GENETIC ALGORITHM** tool created by the Software Technology Branch (STB) of the Information Systems Directorate at NASA/Johnson Space Center with support from the MITRE Corporation. Splicer has well-defined interfaces between a **GA** kernel, representation libraries, **FITNESS** modules, and user interface libraries.

The representation libraries contain functions for defining, creating, and decoding genetic strings, as well as multiple **CROSSOVER** and **MUTATION** operators. Libraries supporting binary strings and permutations are provided, others can be created by the user.

Fitness modules are typically written by the user, although some sample applications are provided. The modules may contain a fitness function, initial values for various control parameters, and a function which graphically displays the best solutions.

Splicer provides event-driven graphic user interface libraries for the Macintosh and the X11 window system (using the HP widget set); a menu-driven ASCII interface is also available though not fully supported. The extensive documentation includes a reference manual and a user's manual; an architecture manual and the advanced programmer's manual are currently being written.

An electronic bulletin board (300/1200/2400 baud, 8N1) with information regarding Splicer can be reached at (713) 280-3896 or (713) 280-3892. Splicer is available free to NASA and its contractors for use on government projects by calling the STB Help Desk weekdays 9am-4pm CST at (713) 280-2233. Government contractors should have their contract monitor call the STB Help Desk; others may purchase Splicer for $221 (incl. documentation) from: COSMIC, 382 E. Broad St., Athens, GA 30602, USA. (Unverified 8/94). Last known address <bayer@galileo.jsc.nasa.gov> (Steve Bayer). This now bounces back with "user unknown".

TOLKIEN:

TOLKIEN (TOoLKIt for gENetics-based applications) is a C++ class library, intended for those involved in **GAs** and **CLASSIFIER SYSTEM** research with a working knowledge of C++. It is designed to reduce effort in developing genetics-based applications by providing a collection of reusable objects. For portability, no compiler specific or class library specific features are used. The current version has been compiled successfully using Borland C++ Version 3.1 and GNU C++.

TOLKIEN contains a lot of useful extensions to the generic **GENETIC ALGORITHM** and classifier system architecture. Examples include: (i) **CHROMOSOMEs** of user-definable types; binary, character, integer and floating point; (ii) Gray code encoding and decoding; (iii) multi-point and uniform **CROSSOVER;** (iv) diploidy and dominance; (v) various **SELECTION** schemes such as tournament selection and linear ranking; (vi) linear **FITNESS** scaling and sigma truncation; (vii) the simplest one-taxon-one-action classifiers and the general two-taxa-one-action classifiers.

TOLKIEN is available from **ENCORE** (See Q15.3) in file: **GA/src/TOLKIEN.tar.gz** The documentation and two primers on how to build GA and **CFS** applications alone are available as: **GA/docs/tolkien-doc.tar.gz**

Author: Anthony Yiu-Cheung Tang <tang028@cs.cuhk.hk>, Department of Computer Science (Rm 913), The Chinese University of Hong Kong. Tel: 609-8403, 609-8404.

Trans-Dimensional Learning:

This is a Windows 3.1 artificial neural netwrk and **GA** program (shareware). TDL allows users to perform pattern recognition by utilizing software that allows for fast, automatic construction of Neural Networks, mostly alleviating the need for parameter tuning. Evolutionary processes combined with semi-weighted networks (hybrid cross between standard weighted neurons and weightless n-level threshold units) generally yield very compact networks (i.e., reduced connections and hidden units). By supporting multi-shot learning over standard one-shot learning, multiple data sets (characterized by varying input and output dimensions) can be learned incrementally, resulting in a single coherent network. This can also lead to significant improvements in predictive accuracy (Trans-dimensional generalization). Graphical support and several data files are also provided.

Available on the WWW from: **http://pages.prodigy.com/upso**

For further details contact: <upso@prodigy.com>

WOLF:

This is a simulator for the G/SPLINES (genetic spline models) algorithm which builds spline-based functional models of experimental data, using **CROSSOVER** and **MUTATION** to evolve a **POPULATION** towards a better fit. It is derived from Friedman's MARS models. The original work was presented at ICGA-4, and further results including additional basis function types such as B-splines have been presented at the NIPS-91 meeting.

Available free by **FTP** by contacting the author; runs on SUN (and possibly any SYSV) UNIX box. Can be redistributed for noncommercial use. Simulator includes executable and C source code; a technical report (RIACS tech report 91.10) is also available.

David Rogers, MS Ellis, NASA Ames Research Center, Moffett Field, CA 94035, USA. Net: <drogers@msi.com>

CLASSIFIER SYSTEMS

CFS-C:

CFS-C 1.0 is a domain independent collection of **CLASSIFIER SYSTEM** routines written by Rick L. Riolo as part of his PhD dissertation. A completely rewritten CFS-C is planned for 1994/95; this may include the features of CFS-C 2.0 mentioned in [SAB90] (e.g. "latent learning") or they may be included in a separate package released in 1995. An ANSIfied version of CFS-C 1.0 (CFS-C 1.98j) is available by **FTP.**

CFS-C is available from **ENCORE** (See Q15.3) in file: **CFS/src/cfsc-1.98j.tar.gz** and includes the original 1.02 CFS-C in its "cfsc/orig" folder after unpacking. On the "SyS" FTP server its: **lumpi.informatik.uni-dortmund.de/pub/LCS/src/cfsc-1.98j.tar.gz** with documentation in **/pub/LCS/docs/cfsc.ps.gz**

Another version of CFS-C (version XV 0.1) by Jens Engel <engel@asterix.irb.uni-hannover.de> is also available. This includes bug fixes of earlier versions, allowing it to run on a wider range of machines (e.g. Linux and nCUBE). It also has an XView front end that makes it easier to control, and some extensions to the algorithms. It is available from Encore in file: **CFS/src/cfscxv-0.1.tar.gz** with documentation in **CFS/docs/cfscxv-0.1.readme.gz**

References

Rick L. Riolo (1988) "CFS-C: A package of domain independent subroutines for implementing classifier systems in arbitrary, user-defined environments", Logic of computers group, Division of computer science and engineering, University of Michigan.

Rick L. Riolo (1988) "LETSEQ: An implementation of the CFS-C classifier-system in a task-domain that involves learning to predict letter sequences", Logic of computers group, Division of computer science and engineering, University of Michigan.

Rick L. Riolo (1988) "CFS-C/FSW1: An implementation of the CFS-C classifier system in a task domain that involves learning to traverse a finite state world", Logic of computers group, Division of computer science and engineering, University of Michigan.

SCS-C:

SCS-C is a (‘mostly ANSI’) C language translation and extension of Goldberg’s Simple **CLASSIFIER SYSTEM,** as presented in Appendix D in his seminal book [GOLD89].

SCS-C has been developed in parallel on a Sun 10/40 and an ATARI ST, and thus should be quite portable; it’s distributed free of charge under the terms of the GNU General Public License. Included are some additional goodies, e.g. the VAX/VMS version of SCS, rewritten in C by Erik Mayer <emayer@uoft02.utoledo.edu>.

SCS-C v1.0j is available from **ENCORE** (See Q15.3), by **FTP in file EC/CFS/src/scsc-1.0j.tar.gz**

For more information contact: Jörg Heitkötter, UUNET Deutschland GmbH, Techo-Park, Emil-Figge-Str. 80, D-44227 Dortmund, Germany. Net: <joke@de.uu.net>.

Q20.2: Commercial software packages?

ActiveGA:

ActiveGA is an activeX (OLE) control that uses a **GENETIC ALGORITHM** to find a solution for a given problem. For example, you can insert an ActiveGA control into Microsoft Excel 97 and have it optimize your worksheet.

Features include:

- **OPTIMIZATION** Mode: Minimize, Maximize or Closest To
- **SELECTION** Mode: Tournament, Roulette Wheel
- User defined **POPULATION** size, **MUTATION** rate and other parameters
- Event driven, cancelable iteration

- Invisible at run time
- Excel 97, Visual Basic, Visual C++ samples

Various samples are available for free download. For these and further information, see **http://www.brightsoft.com/products/activega.htm** or contact Brightwater Software <support@brightsoft.com>. For a limited time the ActiveGA costs $99 per developer. ActiveGA has no run time royalties.

EnGENEer:

Logica Cambridge Ltd. developed EnGENEer as an in-house **GENETIC ALGORITHM** environment to assist the development of **GA** applications on a wide range of domains. The software was written in C and runs under Unix as part of a consultancy and systems package. It supports both interactive (X-Windows) and batch (command-line) modes of operation.

EnGENEer provides a number of flexible mechanisms which allow the developer to rapidly bring the power of GAs to bear on new problem domains. Starting with the Genetic Description Language, the developer can describe, at high level, the structure of the ‘‘genetic material’’ used. The language supports discrete **GENEs** with user defined cardinality and includes features such as multiple **CHROMOSOMEs** models, multiple **SPECIES** models and non-evolvable parsing symbols which can be used for decoding complex genetic material.

The user also has available a descriptive high level language, the Evolutionary Model Language. It allows the description of the GA type used in terms of configurable options including: **POPULATION** size, population structure and source, **SELECTION** method, **CROSSOVER** and **MUTATION** type and probability, **INVERSION,** dispersal method, and number of **OFFSPRING** per **GENERATION.**

Both the Genetic Description Language and the Evolutionary Model Language are fully supported within the interactive interface (including online help system) and can be defined either "on the fly" or loaded from audit files which are automatically created during a GA run.

Monitoring of GA progress is provided via both graphical tools and automatic storage of results (at user defined intervals). This allows the user to restart EnGENEer from any point in a run, by loading both the population at that time and the evolutionary model that was being used.

Connecting EnGENEer to different problem domains is achieved by specifying the name of the program used to evaluate the problem specific **FITNESS** function and constructing a simple parsing routine to interpret the genetic material. A library of standard interpretation routines are also provided for commonly used representation schemes such as gray-coding, permutations, etc. The fitness evaluation can then be run as either a slave process to the GA or via a standard handshaking routines. Better still, it can be run on either the machine hosting the EnGENEer or on any sequential or parallel hardware capable of connecting to a Unix machine.

For more information, contact: George Robbins, Systems Intelligence Division, Logica Cambridge Ltd., Betjeman House, 104 Hills Road, Cambridge CB2 1LQ, UK. Tel: +44 1716 379111, Fax: +44 1223 322315 (Unverified 8/94).

EvoFrame:

EvoFrame is to **EVOLUTION STRATEGIEs** what MicroGA is to **GENETIC ALGORITHMs,** a toolkit for application development incorporating **ESs** as the **OPTIMIZATION** engine.

EvoFrame is an object oriented implemented programming tool for evolution strategies (Rechenberg/Schwefel, Germany) for easy implementation and solution of numerical and combinatorical problems. EvoFrame gives you freedom of implementing every byte of the optimization principle and its user interface. You can focus on the optimization problem and forget about all the rest.

EvoFrame is available as Version 2.0 in Borland-Pascal 7.0 and Turbo-Vision for PC's and as Version 1.0 in C++ for Apple Macintosh using MPW and MacApp. Both implementations allow full typed implementation, i.e. *no more* translation from problem specific format to an optimization specific one. A prototyping tool (cf REALizer) exists for both platforms too.

EvoFrame allows pseudoparallel optimization of many problems at once and you can switch optimization parameters and internal methods (i.e. quality function etc.) during runtime and during optimization cycle. Both tools can be modified or extended by overloading existing methods for experimental use. They are developed continously in correlation to new research results.

The PC version is prepared for experimental use due to a comprehensive protocolling mechanism of optimzation cycles and user data. It also allows compilation of executable files with different complexity by setting conditional compilation flags. It can be used with 3 levels of stacked **POPULATIONs.**

The Mac version is the more complex (recursive) implementation. It allows stacking of any number of populations for modelling of complex systems. Theory stops at multipopulation level at the time. EvoFrame for Mac is ready for the future, allowing any number of population levels.

Ask for porting the Mac version (C++) to any other platform, i.e. X Windows.

REALizer is a tool for rapid prototyping of EvoFrame applications. It's an override of the corresponding framework which is prepared to optimize using a vector of real numbers. All methods for standard **EVOLUTION** and file handling, etc. are ready implemented. The remaining work for the user is to define a constant for the problem size, fill in the quality function and start the optimization process.

For further information, current prices and orders, contact: Wolfram Stebel, Optimum Software, Braunfelser Str. 26, 35578 Wetzlar, Germany. Net: <optimum@applelink.apple.com>

Evolver:

Evolver is a **GENETIC ALGORITHM** package for Windows. Beginners can use the Excel add-in to model and solve problems from within Excel. Advanced users can use the included Evolver API to build custom applications that access any of the six different genetic algorithms. Evolver can be customized and users can monitor progress in real-time graphs, or change parameters through the included EvolverWatcher program. The package costs $349 (or UKP350), comes on two 3.5" disks, and includes support for Visual Basic. For further information or to order, contact: Palisade Corp, (607) 277-8000 **http://www.palisade.com** or Palisade Europe <sales@palisade-europe.com>, Tel +44 1752 204310 **http://www.palisade-europe.com**

FlexTool:

FlexTool(GA) is a modular software tool which provides an **ENVIRONMENT** for applying **GA** to diverse domains with minimum user interaction and design iteration.

Version M2.2 is the MATLAB version which provides a total GA based design and development environment in MATLAB. MATLAB provides us with an interactive computation intensive environment. The high level, user friendly programming language combined with built-in functions to handle matrix algebra, Fourier series, and complex valued functions provides the power for large scale number crunching.

The GA objects are provided as .m files. FlexTool(GA) Version M2.2 is designed with emphasis on modularity, flexibility, user friendliness, environment transparency, upgradability, and reliability. The design is engineered to evolve complex, robust models by drawing on the power of MATLAB.

FlexTool(GA) Version M2.2 Features:
BUILDING BLOCK : Upgrade to EFM or ENM or CI within one year
Niching module : to identify multiple solutions
Clustering module : Use separately or with Niching module
Optimization : Single and Multiple Objectives
Flex-GA : Very fast proprietary learning algorithm

GA : Modular, User Friendly, and System Transparent
GUI : Easy to use, user friendly
Help : Online
Tutorial : Hands-on tutorial, application guidelines
Parameter Settings : Default parameter settings for the novice
General : Statistics, figures, and data collection
Compatibility : FlexTool product suite

GA options : generational, steady state, micro, Flex-GA
Coding schemes : include binary, logarithmic, real
Selection : tournament, roulette wheel, ranking
Crossover : include 1, 2, multiple point crossover
Compatible to : FlexTool(GA) M1.1 Genetic Algorithms Toolbox

The FlexTool product suite includes various soft computing **BUILDING BLOCKs:**
CI: Computational Intelligence **http://www.flextool.com/ftci.html**
EFM: Evolutionary Fuzzy Modeling **http://www.flextool.com/ftefm.html**
ENM: Evolutionary Neuro Modeling **http://www.flextool.com/ftenm.html**
FS : Fuzzy Systems **http://www.flextool.com/ftfs.html**
EA : **EVOLUTIONARY ALGORITHMs http://www.flextool.com/ftga.html**
NN : Neural Networks **http://www.flextool.com/ftnn.html**

For information contact <info@flextool.com> **http://www.flextool.com**

GAME:

GAME (GA Manipulation Environment) aims to demonstrate GA applications and build a suitable programming **ENVIRONMENT.**

GAME is being developed as part of the PAPAGENA project of the European Community's Esprit III initiative.

GAME is available as an addendum to a book on PGAs (cf PAPAGENA, Q20.3). And from the project's **FTP** server **bells.cs.ucl.ac.uk/papagena/** e.g. "papagena/game/docs" contains all the papers that have been produced over the course of the GAME project. The sources can also be obtained by FTP see **papagena/game/version2.01/**

GAME is now in version 2.01. This version is still able to run only sequential **GAs,** but version 3.0 will handle parallel GAs as well.

Unfortunately, The project yet only produced a Borland C++ 3.x version, so far. It is intended to distribute a version for UNIX/GNU C++ as well, when some compatibility issues concerning C++ "standards" have been resolved. Afterward a UNIX version will be released, but this will be only happen after the release of PC version 3.0.

For more information contact: Jose Luiz Ribeiro Filho, Department of Computer Science, University College London, Gower Street, London WC1E 6BT, UK. Net: <zeluiz@cs.ucl.ac.uk> (Unverified 8/94).

GeneHunter:

GeneHunter from Ward Systems runs on a PC under Windows. It is callable from Microsoft Excel 5 spreadsheets, and accessible via function calls in a dynamic link library. The DLL is designed especially for Visual Basic, but runs with other languages which call DLLs under Windows 3.1 such as Visual C++. 16- and 32-bit versions are available. GeneHunter can also integrate with Ward's neural network software. Cost $369.

For full details, see **http://www.wardsystems.com/** or contact: Ward Systems Group Inc, Executive Park West, 5 Hillcrest Drive, Frederick, MD 21703, USA. 301-662-7950 <wardsystems@msn.com>

Generator:

GENERATOR is a **GENETIC ALGORITHM** package designed to interact with Microsoft Excel for Windows. Users are able to define and solve problems using Excel formulas, tables and functions. **FITNESS** is easily defined as an Excel formula or optionally a macro. Progress can be monitored using GENERATOR's real-time fitness graph and status window as well as user-defined Excel graphs. GENERATOR can be paused at any time to allow adjustment of any of the parameters and then resumed.

GENERATOR Features:

- Multiple **GENE** types: integer, real and permutation.
- Combined roulette-wheel and elitist **SELECTION** method.
- **ELITISM** is optional and adjustable.
- None, two-point, and a proprietary permutation **CROSSOVER.**
- Random, Random Hillclimb and Directional Hillclimb **MUTATION** methods.
- Special hillclimbing features to find solutions faster.
- fitness goal: maximize, minimize or seek value.
- Convergence: duplicates not allowed.
- Real-Time alteration of parameters relating to crossover, mutation, **POPULATION,** etc.
- Real-Time progress graph of Best, Worst and Median fitness.
- fitness defined using an Excel formula or macro.

The parameters available to the user include mutation probability for population and genes, control of mutation limit per gene, control of hillclimbing, population size, elite group size, **RECOMBINATION** method, and mutation technique.

Connecting generator to problems defined on the Excel spreadsheet is achieved by first specifying the spreadsheet locations of the gene group cells and their type, and lastly, the location of the formula used to evaluate the problem-specific fitness function.

GENERATOR requires at least a 386 IBM compatible PC with 2 MB of RAM, Windows 3.0 (or later) and Microsoft Excel 4.0 (or later). A comprehensive manual includes an explanation of genetic algorithms and several tutorial example problems. The $379 package.includes GENERATOR on a 3.5" diskette, the manual, and free customer support.

For further information or to order, contact: New Light Industries, Ltd.; 9713 W. Sunset Hwy; Spokane, WA USA 99204 Tel: (509) 456-8321; Fax (509) 456-8351; E-mail: <nli@comtch.iea.com> WWW page: **http://www.iea.com/~nli**

MicroGA:

MicroGA is a powerful and flexible new tool which allows programmers to integrate **GAs** into their software quickly and easily. It is an object-oriented C++ framework that comes with full source code and documentation as well as three sample applications. Also included is the Galapagos code generator which allows users to create complete applications interactively without writing any C++ code, and a sample MacApp interface.

MicroGA is available for Macintosh II or higher with MPW and a C++ compiler, and also in a Microsoft Windows version for PC compatibles. Compiled applications made with MicroGA can be sold without license fee. MicroGA is priced at $249.

Galapagos is a tool for use with Emergent Behavior's MicroGA Toolkit. It allows a user to define a function and set of constraints for a problem that the user wants to solve using the GA. Galapagos then generates a complete C++ program using the information supplied. Then all the user has to do is to compile these files, using either Turbo/Borland C++ (PC, MS Windows), or MPW and C++ compiler (Macintosh), and link the resulting code to the MicroGA library. Then just run the program. Galapagos comes free with every copy of MicroGA.

For further information and orders, contact: Steve Wilson, Emergent Behavior, 635 Wellsbury Way, Palo Alto, CA 94306, USA. Net: <emergent@aol.com>

MicroGA is distributed in Germany by Optimum Software (cf EvoFrame & REALizer entries).

Omega:

The Omega Predictive Modeling System, marketed by KiQ Limited, is a powerful approach to developing predictive models. It exploits advanced **GA** techniques to create a tool which is "flexible, powerful, informative and straightforward to use". Omega is geared to the financial domain, with applications in Direct Marketing, Insurance, Investigations and Credit Management. The **ENVIRONMENT** offers facilities for automatic handling of data; business, statistical or custom measures of **PERFORMANCE,** simple and complex profit modeling, validation sample tests, advanced confidence tests, real time graphics, and optional control over the internal GA.

For further information, contact: KiQ, Business Modeling Systems Ltd., Easton Hall, Great Easton, Essex CM6 2HD, UK. Tel: +44 1371 870254 (Unverified 8/94).

OOGA:

OOGA (Object-Oriented **GA)** is a **GENETIC ALGORITHM** designed for industrial use. It includes examples accompanying the tutorial in the companion "Handbook of Genetic Algorithms". OOGA is designed such that each of the techniques employed by a GA is an object that may be modified, displayed or replaced in object-oriented fashion. OOGA is especially well-suited for individuals wishing to modify the basic GA techniques or tailor them to new domains.

The buyer of OOGA also receives Genesis (see above). This release sports an improved user interface. OOGA and Genesis are available together on 3.5" or 5.25" disk for $60 ($52.50 inside North America) by order from: The Software Partnership (T.S.P.), P.O. Box 991, Melrose, MA 02176, USA. Tel: +1 617 662 8991 (Unverified 8/94).

PC-Beagle:

PC-Beagle is a rule-finder program for PCs which examines a database of examples and uses machine-learning techniques to create a set of decision rules for classifying those examples, thus turning data into knowledge. The system contains six major components, one of which (HERB - the "Heuristic Evolutionary Rule Breeder") uses **GA** techniques to generate rules by natural **SELECTION.**

PC-Beagle is available to educational users for 69 pounds sterling. Orders, payment or requests for information should be addressed to: Richard Forsyth, Pathway Research Ltd., 59 Cranbrook Rd., Bristol BS6 7BS, UK. Tel: +44 117 942 8692 (Unverified 8/94).

XpertRule GenAsys:

XpertRule GenAsys is an expert system shell with embedded **GENETIC ALGORITHM** marketed by Attar Software. Targeted to solve scheduling and design applications, this system combines the power of genetic algorithms in evolving solutions with the power of rule-based programming in analyzing the effectiveness of solutions. Rule-based programming can also be used to generate the initial **POPULATION** for the genetic algorithm and for post-optimization planning. Some examples of design and scheduling problems which can be solved by this system include: **OPTIMIZATION** of design parameters in electronic and avionic industries, route optimization in the distribution sector, production scheduling in manufacturing, etc.

For further information, contact: Attar Software, Newlands Road, Leigh, Lancashire, UK. Tel: +44 1942 608844. <100116.1547@CompuServe.com> **http://www.attar.com** (confirmed 3/96).

XYpe:

XYpe (The **GA** Engine) is a commercial GA application and development package for the Apple Macintosh. Its standard user interface allows you to design **CHROMOSOMEs,** set attributes of the genetic engine and graphically display its progress. The development package provides a set of Think C libraries and include files for the design of new GA applications. XYpe supports adaptive operator weights and mixtures of alpha, binary, gray, ordering and real number codings.

The price of $725 (in Massachusetts add 5% sales tax) plus $15 shipping and handling includes technical support and three documentation manuals. XYpe requires a Macintosh SE or newer with 2MB RAM running OS V6.0.4 or greater, and Think C if using the development package.

Currently the GA engine is working; the user interface will be completed on demand. Interested parties should contact: Ed Swartz, Virtual Image, Inc., 75 Sandy Pond Road #11, Ayer, MA 01432, USA. Tel: +1 (508) 772-4225 (Unverified 8/94).

Q20.3: Current research projects?

PAPAGENA:

The European ESPRIT III project PAPAGENA is pleased to announce the availability of the following book and software:

Parallel Genetic Algorithms: Theory and Applications was recently published by IOS press. The book, edited by Joachim Stender, provides an overview of the theoretical, as well as practical, aspects involved in the study and implementation of parallel **GENETIC ALGORITHMs** (PGAs).

The book comes with a floppy disk version of GAME (Genetic Algorithm Manipulation Environment). For more information see the section on GAME in Q20.2.

PeGAsuS:

PeGAsuS is a general programming environment for evolutionary algorithms. developed at the German National Research Center for Computer Science. Written in ANSI-C, it runs on MIMD parallel machines, such as transputers, and distributed systems, as well as serial machines.

The Library contains **GENETIC OPERATORs,** a collection of **FITNESS** functions, and input/output and control procedures. It provides the user with a number of validated modules. Currently, PeGAsuS can be compiled with the GNU C, RS/6000 C, ACE-C, and Alliant's FX/2800 C compilers. It runs on SUNs and RS/6000 workstations, as well as on the Alliant FX/28. PeGAsuS is not available to the public.

For more information contact: Dirk Schlierkamp-Voosen, Research Group for Adative Systems, German National Research Center for Computer Science, 53731 Sankt Augustin, Germany. Net: <dirk.schlierkamp-voosen@gmd.de>

Q21: What are Gray codes, and why are they used?

The correct spelling is "Gray"−−−not "gray", "Grey", or "grey"−−− since Gray codes are named after the Frank Gray who patented their use for shaft encoders in 1953 [1]. Gray codes actually have a longer history, and the inquisitive reader may want to look up the August, 1972, issue of Scientific American, which contains two articles of interest: one on the origin of binary codes [2], and another by Martin Gardner on some entertaining aspects of Gray codes [3]. Other references containing descriptions of Gray codes and more modern, non-GA, applications include the second edition of Numerical Recipes [4], Horowitz and Hill [5], Kozen [6], and Reingold [7].

A Gray code represents each number in the sequence of integers {0...2^N-1} as a binary string of length N in an order such that adjacent integers have Gray code representations that differ in only one bit position. Marching through the integer sequence therefore requires flipping just one bit at a time. Some call this defining property of Gray codes the "adjacency property" [8].

Example (N=3): The binary coding of {0...7} is {000, 001, 010, 011, 100, 101, 110, 111}, while one Gray coding is {000, 001, 011, 010, 110, 111, 101, 100}. In essence, a Gray code takes a binary sequence and shuffles it to form some new sequence with the adjacency property. There exist, therefore, multiple Gray codings for any given N. The example shown here belongs to a class of Gray codes that goes by the fancy name "binary-reflected Gray codes". These are the most commonly seen Gray codes, and one simple scheme for generationg such a Gray code sequence says, "start with all bits zero and successively flip the right-most bit that produces a new string."

Hollstien [9] investigated the use of **GAs** for optimizing functions of two variables and claimed that a Gray code representation worked slightly better than the binary representation. He attributed this difference to the adjacency property of Gray codes. Notice in the above example that the step from three to four requires the flipping of all the bits in the binary representation. In general, adjacent integers in the binary representaion often lie many bit flips apart. This fact makes it less likely that a **MUTATION** operator can effect small changes for a binary-coded **INDIVIDUAL.**

A Gray code representation seems to improve a mutation operator's chances of making incremental improvements, and a close examination suggests why. In a binary-coded string of length N, a single mutation in the most significant bit (MSB) alters the number by 2^(N-1). In a Gray-coded string, fewer mutations lead to a change this large. The user of Gray codes does, however, pay a price for this feature: those "fewer mutations" lead to much larger changes. In the Gray code illustrated above, for example, a single mutation of the left-most bit changes a zero to a seven and vice-versa, while the largest change a single mutation can make to a corresponding binary-coded individual is always four. One might still view this aspect of Gray codes with some favor: most mutations will make only small changes, while the occasional mutation that effects a truly big change may initiate **EXPLORATION** of an entirely new region in the space of **CHROMOSOMEs.**

The algorithm for converting between the binary-reflected Gray code described above and the standard binary code turns out to be surprisingly simple to state. First label the bits of a binary-coded string B[i], where larger i's represent more significant bits, and similarly label the corresponding Gray-coded string G[i]. We convert one to the other as follows: Copy the most significant bit. Then for each smaller i do either G[i] = XOR(B[i+1], B[i])−−−to convert binary to Gray−−−or B[i] = XOR(B[i+1], G[i])−−−to convert Gray to binary.

One may easily implement the above algorithm in C. Imagine you do something like

typedef unsigned short ALLELE;

and then use type "allele" for each bit in your chromosome, then the following two functions will convert between binary and Gray code representations. You must pass them the address of the high-order bits for each of the two strings as well as the length of each string. (See the comment statements for examples.) NB: These functions assume a chromosome arranged as shown in the following illustration.

```
index:     C[9]                                C[0]
           *-----------------------------------------*
Char C:    | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
           *-----------------------------------------*
           ^^^^^                              ^^^^^
           high-order bit                     low-order bit
```

C CODE

```
/* Gray <==> binary conversion routines */
/* written by Dan T. Abell, 7 October 1993 */
/* please send any comments or suggestions */
/* to dabell@quark.umd.edu */

void gray_to_binary (Cg, Cb, n)
/* convert chromosome of length n+1 */
/*     from   Gray code Cg[0...n] */
/*       to  binary code Cb[0...n] */

allele      *Cg,*Cb;
int         n;
{
            int j;

            *Cb = *Cg;                      /* copy the high-order bit */
            for (j = 0; j < n; j++) {
                        Cb--; Cg--;         /* for the remaining bits */
                        *Cb= *(Cb+1)^*Cg;   /* do the appropriate XOR */
            }
}

void binary_to_gray(Cb, Cg, n)
/* convert chromosome of length n+1 */
/*     from  binary code Cb[0...n] */
/*       to    Gray code Cg[0...n] */

allele      *Cb, *Cg;
int         n;
{
            int j;

            *Cg = *Cb;                      /* copy the high-order bit */
            for (j = 0; j < n; j++) {
                        Cg--; Cb--;         /* for the remaining bits */
                        *Cg= *(Cb+1)^*Cb;   /* do the appropriate XOR */
            }
}
```

References

[1] F. Gray, "Pulse Code Communication", U. S. Patent 2 632 058, March 17, 1953.

[2] F. G. Heath, "Origins of the Binary Code", Scientific American v.227,n.2 (August, 1972) p.76.

[3] Martin Gardner, "Mathematical Games", Scientific American v.227,n.2 (August, 1972) p.106.

[4] William H. Press, et al., Numerical Recipes in C, Second Edition (Cambridge University Press, 1992).

[5] Paul Horowitz and Winfield Hill, The Art of Electronics, Second Edition (Cambridge University Press, 1989).

[6] Dexter Kozen, The Design and Analysis of Algorithms (Springer-Verlag, New York, NY, 1992).

[7] Edward M. Reingold, et al., Combinatorial Algorithms (Prentice Hall, Englewood Cliffs, NJ, 1977).

[8] David E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning (Addison-Wesley, Reading, MA, 1989).

[9] R. B. Hollstien, Artificial Genetic Adaptation in Computer Control Systems (PhD thesis, University of Michigan, 1971).

Q22: What test data is available?

TSP DATA

There is a **TSP** library (TSPLIB) available which has many solved and semi-solved TSPs and different variants. The library is maintained by Gerhard Reinelt <reinelt@ares.iwr.Uni-Heidelberg.de>. It is available from various **FTP** sites, including: **softlib.cs.rice.edu/pub/tsplib/tsblib.tar**

OPERATIONAL RESERACH DATA

Information about Operational Research test problems in a wide variety of areas can be obtained by emailing <o.rlibrary@ic.ac.uk> with the body of the email message being just the word "info". The files in OR-Library are also available via anonymous **FTP** from **mscmga.ms.ic.ac.uk/pub/** A WWW page is also available at URL: **http://mscmga.ms.ic.ac.uk/** Instructions on how to use OR-Library can be found in the file "paper.txt", or in the article: J.E.Beasley, "OR-Library: distributing test problems by electronic mail", Journal of the Operational Research Society 41(11) (1990) pp1069-1072.

The following is a list of some of the topics covered.

File	Problem area
assigninfo.txt	Assignment problem
deainfo.txt	Data envelopment analysis
gapinfo.txt	Generalised assignment problem
mipinfo.txt	Integer programming
lpinfo.txt	Linear programming
scpinfo.txt	Set covering
sppinfo.txt	Set partitioning
tspinfo.txt	Travelling salesman problem
periodtspinfo.txt	Period travelling salesman problem
netflowinfo.txt	Network flow problem

Location:

capmstinfo.txt	capacitated minimal spanning tree
capinfo.txt	capacitated warehouse location
pmedinfo.txt	p-median
uncapinfo.txt	uncapacitated warehouse location
mknapinfo.txt	Multiple knapsack problem
qapinfo.txt	Quadratic assignment problem
rcspinfo.txt	Resource constrained shortest path
phubinfo.txt	p-hub location problem

Scheduling:

airlandinfo.txt	Aircraft Landing Problem
cspinfo.txt	Crew scheduling
flowshopinfo.txt	flow shop
jobshopinfo.txt	job shop
openshopinfo.txt	open shop
tableinfo.txt	timetabling problem

Steiner:

esteininfo.txt	Euclidean Steiner problem
rsteininfo.txt	Rectilinear Steiner problem
steininfo.txt	Steiner problem in graphs

Two-dimensional cutting:

assortinfo.txt	assortment problem
cgcutinfo.txt	constrained guillotine
ngcutinfo.txt	constrained non-guillotine
gcutinfo.txt	unconstrained guillotine

Vehicle routing:

areainfo.txt	fixed areas
fixedinfo.txt	fixed routes
periodinfo.txt	period routing
vrpinfo.txt	single period
multivrpinfo.txt	multiple depot vehicle routing problem

OTHER DATA

William Spears <spears@aic.nrl.navy.mil> maintains a WWW page titled: *Test Functions for Evolutionary Algorithms* which contians links to various sources of test functions. **http://www.aic.nrl.navy.mil:80/~spears/functs.html**

ENCORE (see Q15.3) also contains some test data. See directories under **/etc/data/**

Q42: What is Life all about?

42

References

Adams, D. (1979) "The Hitch Hiker's Guide to the Galaxy", London: Pan Books.

Adams, D. (1980) "The Restaurant at the End of the Universe", London: Pan Books.

Adams, D. (1982) "Life, the Universe and Everything", London: Pan Books.

Adams, D. (1984) "So long, and thanks for all the Fish", London: Pan Books.

Adams, D. (1992) "Mostly Harmless", London: Heinemann.

Q42b: Is there a FAQ to this group?

Yes.

Q98: Are there any patents on EAs?

Process patents have been issued both for the Bucket Brigade Algorithm in **CLASSIFIER SYSTEMs:** U.S. patent #4,697,242: J.H. Holland and A. Burks, "Adaptive computing system capable of learning and discovery", 1985, issued Sept 29 1987; and for **GP:** U.S. patent #4,935,877 (to John Koza).

This **FAQ** does not attempt to provide legal advice. However, use of the Lisp code in the book [KOZA92] is freely licensed for academic use. Although those wishing to make commercial use of any process should obviously consult any patent holders in question, it is pretty clear that it's not in anyone's best interests to stifle GA/GP research and/or development. Commercial licenses much like those used for CAD software can presumably be obtained for the use of these processes where necessary.

Jarmo Alander's massive bibliography of **GAs** (see Q10.8) includes a (probably) complete list of all currently know patents. There is also a periodic posting on **comp.ai.neural-nets** by Gregory Aharonian <srctran@world.std.com> about patents on Artificial Neural Networks (ANNs).

Q99: A Glossary on EAs?

A very good glossary of genetics terminology can be found at **http://helios.bto.ed.ac.uk/bto/glossary**

1

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.

A

ADAPTIVE BEHAVIOUR:
"...underlying mechanisms that allow animals, and potentially, **ROBOTs** *to adapt and survive in uncertain environments"* ––– Meyer & Wilson (1991), [SAB90]

AI: See **ARTIFICIAL INTELLIGENCE.**

ALIFE:
See **ARTIFICIAL LIFE.**

ALLELE :
(biol) Each **GENE** is able to occupy only a particular region of a **CHROMOSOME,** its locus. At any given locus there may exist, in the **POPULATION,** alternative forms of the gene. These alternative are called *alleles* of one another.

(EC) The value of a gene. Hence, for a binary representation, 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."*

B

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.

C

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 behaviour of the algorithm must be analysed using the **SCHEMA THEOREM.**

(life in general) The dogma that this all is useful in a way.

"You guys have a dogma. A certain irrational set of believes. Well, here's my irrational set of beliefs. Something that works."

––– Rodney A. Brooks, [LEVY92]

CFS: See **CLASSIFIER SYSTEM.**

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 datastructure 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'. See Q1.4 for more information.

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 "mu" **PARENTs** generates "lambda" **OFFSPRING** and the mu parents are discarded, leving only the lambda **INDIVIDUALs** to compete directly. Such a process is written as a (mu,lambda) search. The process of only competing offspring then is a "comma strategy." cf. **PLUS 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 their length, thus swapping genetic material"* is thus wrong, since these two activities occur in different parts of the life cycle. [eds note: If sexual reproduction (the Real Thing) worked like in **GAs,** then Holland would be right, but as we all know, it's not the case. We just encountered a Freudian slip of a Grandmaster. BTW: even the German translation of this article has this "bug", although it's well-hidden by the translator.]

CS: See **CLASSIFIER SYSTEM.**

D

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. cf **LAMARCKISM.**

The **talk.origins FAQ** contains more details (See Q10.7). Also, the "Dictionary of Darwinism and of Evolution" (Ed. by Patrick Tort) was published in early 1996. It contains a vast amount of information about what Darwinism is and (perhaps more importantly) is *not.* Further information from **http://www.planete.net/~ptort/darwin/evolengl.html** (in various languages).

(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]

(literature) Douglas Noel Adams, contemporary Science Fiction comedy writer. Published "The Hitch-Hiker's Guide to the Galaxy" when he was 25 years old, which made him one of the currently most successful British authors. [eds note: interestingly Watson was also 25 years old, when he discovered the DNA; both events are probably not interconnected; you might also want to look at: Neil Gaiman's (1987) "DON'T PANIC −− The Official Hitch-Hiker's Guide to the Galaxy companion", and of course get your hands on the wholly remarkable **FAQ** in **alt.fan.douglas-adams**]

DNS: (biol) Desoxyribonukleinsäure, German for **DNA.**

(comp) The Domain Name System, a distributed database system for translating computer names (e.g. **lumpi.informatik.uni-dortmund.de**) into numeric Internet, i.e. IP-addresses (129.217.36.140) and vice-versa. **DNS** allows you to hook into the net without remembering long lists of numeric references, unless your system administrator has incorrectly set-up your site's system.

E

EA: See **EVOLUTIONARY ALGORITHM.**

EC: See **EVOLUTIONARY COMPUTATION.**

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 mamimum **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.

ENCORE:

The EvolutioNary Computation REpository Network. An collection of **FTP** servers/World Wide Web sites holding all manner of interesting things related to **EC.** See Q15.3 for more information.

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 **POPULATION,** that is, the fitness of an organism's behaviour may depend upon how many others are also doing it. Over several **GENERATIONs,** biotic environments may foster co-evolution, in which fitness is determined with **SELECTION** partly by other **SPECIES.**

EP: See **EVOLUTIONARY PROGRAMMING.**

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.**

ES: See **EVOLUTION STRATEGY.**

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, of which some varieties (3) differ in **FITNESS** (reproductive success). (See the **talk.origins FAQ** for discussion on this (See Q10.7).)

"Don't assume that all people who accept **EVOLUTION** *are atheists"*

––– Talk.origins FAQ

EVOLUTION STRATEGIE:

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.** See Q1.3 for more information.

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 ALGORITHM:
A algorithm designed to perform **EVOLUTIONARY COMPUTATION.**

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 evolutionay 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.** See Q1.2 for more information.

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."

EXPECTANCY:
Or expected value. Pertaining to a random variable X, for a continuous random variable, the expected value is:
E(X) = INTEGRAL(-inf, inf) [X f(X) dX].
The discrete expectation takes a similar form using a summation instead of an integral.

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 hillclimbing, 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.

F

FAQ: Frequently Asked Questions. See definition given before the main table of contents.

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.**

FTP: File Transfer Protocol. A system which allows the retrieval of files stored on a remote computer. Basic **FTP** requires a password before access can be gained to the remote computer. Anonymous FTP does not require a special password: after giving "anonymous" as the user name, any password will do (typically, you give your email address at this point). Files available by FTP are specified as **<ftp-site-name>:<the-complete-filename>** See Q15.5.

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 {+, -, *}.

G

GA: See **GENETIC ALGORITHM.**

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 its 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.**

GAUSSIAN DISTRIBUTION:
See **NORMALLY DISTRIBUTED.**

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 the chromosome. However, the term may be defined in different ways for different purposes. For a fuller story, consult a book on genetics (See Q10.7).

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.

Everybody out of the gene-pool, now!

−−− Author prefers to be anonymous

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.** See Q1.1 for more information.

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. See Q1.5 for more information.

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 extremum (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.**

GP: See **GENETIC PROGRAMMING.**

H

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." In contrast, "soft selection" offers a probabilistic mechanism for maitaining individuals to be **PARENTs** of future progeny despite possessing relatively poorer objective values.

I

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.

L

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.**

LCS: See **LEARNING CLASSIFIER SYSTEM.**

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. See Q1.4 for more information.

M

MIGRATION:
The transfer of (the **GENEs** of) an **INDIVIDUAL** from one **SUB-POPULATION** to another.

MOBOT:
MOBile **ROBOT.** cf ROBOT.

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.

N

NFL: See **NO FREE LUNCH.**

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.**

NO FREE LUNCH:
Cocktail party definition:

For any pair of search algorithms, there are "as many" problems for which the first algorithm outperforms the second as for which the reverse is true. One consequence of this is that if we don't put any domain knowledge into our algorithm, it is as likely to perform *worse* than random search as it is likely to perform better. This is true for all algorimths including **GENETIC ALGORITHMs.**

More detailed description:

The **NFL** work of Wolpert and Macready is a framework that addresses the core aspects of search, focusing on the connection between **FITNESS** functions and effective search algorithms. The central importance of this connection is demonstrated by the *No Free Lunch* theorem which states that averaged over all problems, all search algorithms perform equally. This result implies that if we are comparing a genetic algorithm to some other algorithm (e.g., simulated annealing, or even random search) and the genetic algorithm performs better on some class of problems, then the other algorithm necessarily performs better on problems outside the class. Thus it is essential to incorporate knowledge of the problem into the search algorithm.

The NFL framework also does the following: it provides a geometric interpretation of what it means for an algorithm to be well matched to a problem; it provides information theoretic insight into the search procedure; it investigates time-varying fitness functions; it proves that *independent of the fitness function,* one cannot (without prior domain knowledge) successfully choose between two algorithms based on their previous behavior; it provides a number of formal measures of how well an algorithm performs; and it addresses the difficulty of **OPTIMIZATION** problems from a viewpoint outside of traditional computational complexity.

NORMALLY DISTRIBUTED:

A random variable is **NORMALLY DISTRIBUTED** if its density function is described as
f(x) = 1/sqrt(2*pi*sqr(sigma)) * exp(-0.5*(x-mu)*(x-mu)/sqr(sigma))
where mu is the mean of the random variable x and sigma is the **STANDARD DEVIATION.**

O

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. **TRAVELLING 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 its birth to death. cf **PHYLOGENESIS.**

P

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.

PERFORMANCE:

cf **FITNESS.**

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 "mu" **PARENTs** generates "lambda" **OFFSPRING** and all mu and lambda **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.**

R

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.**

ROBOT:
"The Encyclopedia Galactica *defines a* **ROBOT** *as a mechanical apparatus designed to do the work of man. The marketing division of the Sirius Cybernetics Corporation defines a robot as 'Your Plastic Pal Who's Fun To Be With'."*

−−− Douglas Adams (1979)

S

SAFIER:
An **EVOLUTIONARY COMPUTATION FTP** Repository, now defunct. Superceeded by **ENCORE.**

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-dont-care positions specified, while the 'defining length' is the distance between the furthest two non-dont-care positions. Thus #1##0# is of order 2 and defining length 3.

SCHEMA THEOREM:
Theorem devised by Holland [HOLLAND92] to explain the behaviour 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.

"Give any one species too much rope, and they'll fuck it up"

−−− Roger Waters, "Amused to Death", 1992

STANDARD DEVIATION:

A measurement for the spread of a set of data; a measurement for the variation of a random variable.

STATISTICS:

Descriptive measures of data; a field of mathematics that uses probability theory to gain insight into systems' behavior.

STEPSIZE:
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.**

SUMMERSCHOOL:
(USA) One of the most interesting things in the US educational system: class work during the summer break.

T

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,* (cf Q42) or a function taking no arguments, such as (move-north).

TRAVELLING SALESMAN PROBLEM:
The travelling 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 minimise the total distance travelled (including returning home)? This is a classical example of an **ORDER-BASED PROBLEM.**

TSP: See **TRAVELLING SALESMAN PROBLEM.**

U

USENET:
"Usenet is like a herd of performing elephants with diarrhea −− massive, difficult to redirect, awe-inspiring, entertaining, and a source of mind-boggling amounts of excrement when you least expect it."

−−− Gene Spafford (1992)

V

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.

Z

ZEN NAVIGATION:

A methodology with a tremendous propensity to get lost during a *hike* from A to B. Zen Navigation simply consists of finding something that looks as if it knows where it is going, and following it. The results are often more surprising than successful, but its usually worth using for the sake of the few occasions when it is both.

Sometimes Zen Navigation is referred to as "*doing scientific research,*" where A is a state of mind considered as being pre-PhD, and B is a (usually a different) state of mind, known as post-PhD. Your time spent in state C, somewhere inbetween A and B, is usually referred to as "*being a nobody.*"

ACKNOWLEDGMENTS

Finally, credit where credit is due. I'd like to thank all the people who helped in assembling this guide, and their patience with my "variations on English grammar". In the order I received their contributions, thanks to:

Contributors,

Lutz Prechelt (University of Karlsruhe) the **comp.ai.neural-nets** FAQmeister, for letting me strip several ideas from "his" **FAQ.** *Ritesh* "peace" *Bansal* (CMU) for lots of comments and references. *David Beasley* (University of Wales) for a valuable list of publications (Q12), and many further additions. *David Corne, Peter Ross,* and *Hsiao-Lan Fang* (University of Edinburgh) for their TIMETABLING and JSSP entries. *Mark Kantrowitz* (CMU) for mocking about this-and-that, and being a "mostly valuable" source concerning FAQ maintenance; parts of Q11 have been stripped from "his" **ai-faq/part4** FAQ; Mark also contributed the *less verbose* archive server infos. The texts of Q1.1, Q1.5, Q98 and some entries of Q99 are courtesy by *James Rice* (Stanford University), stripped from his genetic-programming FAQ (Q15). *Jonathan I. Kamens* (MIT) provided infos on how-to-hook-into the **USENET** FAQ system. *Una Smith* (Yale University) contributed the better parts of the Internet resources guide (Q15.5). *Daniel Polani* (Gutenberg University, Mainz) "contributed" the **ALIFE** II Video proceedings info. *Jim McCoy* (University of Texas) reminded me of the **GP** archive he maintains (Q20). *Ron Goldthwaite* (UC Davis) added definitions of *Environment,* **EVOLUTION,** Fitness, **and** *Population* **to the glossary, and some thoughts why Biologists should take note of EC** (Q3). *Joachim Geidel* (University of Karlsruhe) sent a diff of the current "navy server" contents and the software survey, pointing to "missing links" (Q20). *Richard Dawkins* "Glossary" section of "The extended phenotype" served for many new entries, too numerous to mention here (Q99). *Mark Davis* (New Mexico State University) wrote the part on **EVOLUTIONARY PROGRAMMING** (Q1.2). *Dan Abell* (University of Maryland) contributed the section on efficient greycoding (Q21). *Walter Harms* (University of Oldenburg) commented on introductory EC literature. *Lieutenant Colonel J.S. Robertson* (USMA, West Point), for providing a home for this subversive posting on their **FTP** server **euler.math.usma.edu/pub/misc/GA** *Rosie O'Neill* for suggesting the PhD thesis entry (Q10.11). *Charlie Pearce* (University of Nottingham) for critical remarks concerning "tables"; well, here they are! *J. Ribeiro Filho* (University College London) for the pointer to the IEEE Computer **GA** Software Survey; the PeGAsuS description (Q20) was stripped from it. *Paul Harrald* for the entry on game playing (Q2). *Laurence Moran* (Uni Toronto) for corrections to some of the biological information in Q1 and Q99. *Marco Dorigo* (Uni Libre Bruxelles) gets the award for *reading the guide more thoroughly than* (including the editors). He suggested additions to Q1.4, Q2, Q4 and Q22, and pointed out various typos. *Bill Macready* (SFI) for the two defintions of the **NFL** theorem in Q99.

Cedric Notredame (EBI) for providing information about applications of EC in biology (Q2). *Fabio Pichierri* (The Institute of Physical and Chemical Research) for information on the relevance of EC to chemists (Q3). *Moshe Sipper* (Swiss Federal Institute of Technology) for details of applications in Cellular Automata and Evolvable Hardware (Q2).

Reviewers,

Robert Elliott Smith (The University of Alabama) reviewed the TCGA infos (Q14), and *Nici Schraudolph* (UCSD) first unconsciously, later consciously, provided about 97% of Q20* answers. *Nicheal Lynn Cramer* (BBN) adjusted my historic view of **GP** genesis. *David Fogel* (Natural **SELECTION,** Inc.) commented and helped on this-and-that (where this-and-that is closely related to **EP),** and provided many missing entries for the glossary (Q99). *Kazuhiro M. Saito* (MIT) and *Mark D. Smucker* (Iowa State) caught my favorite typo(s). *Craig W. Reynolds* was the first who solved one of the well-hidden puzzles in the **FAQ,** and also added some valuable stuff. *Joachim Born* (TU Berlin) updated the **EVOLUTION** Machine (EM) entry and provided the pointer to the Bionics technical report **FTP** site (Q14). *Pattie Maes* (MIT Media Lab) reviewed the **ALIFE** IV additions to the list of conferences (Q12). *Scott D. Yelich* (Santa Fe Institute) reviewed the SFI connectivity entry (Q15). *Rick Riolo* (MERIT) reviewed the CFS-C entry (Q20). *Davika Seunarine* (Acadia Univ.) for smoothing out this and that. *Paul Field* (Queen Mary and Westfield College) for correcting typos, and providing insights into the blindfold pogo-sticking nomads of the Himalayas.

and Everybody...

Last not least I'd like to thank *Hans-Paul Schwefel, Thomas Bäck, Frank Kursawe, Günter Rudolph* for their contributions, and the rest of the Systems Analysis Research Group for wholly remarkable patience and almost incredible unflappability during my various extravangances and ego-trips during my time (1990-1993) with this group.

It was a tremendously worthwhile experience. Thanks!

——— The Editor, Jörg Heitkötter (1993)

EPILOGUE

"Natural selection is a mechanism for generating
an exceedingly high degree of improbability."

——— Sir Ronald Aylmer Fisher (1890-1962)

This is a GREAT quotation, it sounds like something directly out of a turn of the century Douglas Adams: Natural selection: the original "Infinite Improbability Drive"

——— Craig Reynolds (1993), on reading the previous quote

'The Babel fish,' said The Hitch Hiker's Guide to the Galaxy quietly, 'is small, yellow and leech-like, and probably the oddest thing in the Universe. It feeds on brainwave energy received not from his own carrier but from those around it. It absorbs all unconscious mental frequencies from this brainwave energy to nourish itself with. It then excretes into the mind of its carrier a telepathic matrix formed by combining the conscious thought frequencies with nerve signals picked up from the speech centers of the brain which has supplied them. The practical upshot of all this is that if you stick a Babel fish in your ear you can instantly understand anything said to you in any form of language. The speech patterns you actually hear decode the brainwave matrix which has been fed into your mind by your Babel fish. 'Now it is such a bizarrely improbable coincidence than anything so mindbogglingly useful could have evolved purely by chance that some thinkers have chosen to see it as a final and clinching proof of the non-*existence of God. 'The argument goes something like this: "I refuse to prove that I exist," says God, "for proof denies faith,*

and without faith I am nothing." "But," says Man, "The Babel fish is a dead giveaway isn't it? It could not have evolved by chance. It proves you exist, and so therefore, by your own arguments, you don't. QED." "Oh dear," says God, "I hadn't thought of that," and promptly vanishes in a puff of logic. "Oh, that was easy," says Man, and for an encore goes on to prove that black is white and gets himself killed on the next zebra crossing.

––– Douglas Adams (1979)

"Well, people; I really wish this thingie to turn into a *paper babel-fish* for all those young ape-descended organic life forms on this crazy planet, who don't have any clue about what's going on in this exciting "new" research field, called **EVOLUTIONARY COMPUTATION.** However, this is just a start, I need your help to increase the usefulness of this guide, especially its readability for natively English speaking folks; whatever it is: I'd like to hear from you...!"

––– The Editor, Jörg Heitkötter (1993)

"Parents of young organic life forms should be warned, that
paper babel-fishes can be harmful, if stuck too deep into the ear."

––– Encyclopedia Galactica

"The meeting of these guys was definitely the best bang since the big one."

––– Encyclopedia Galactica

ABOUT THE EDITORS

Jörg Heitkötter,

was born in 1965 in Recklinghausen, a small but beautiful 750 year old town at the northern rim of the Ruhrgebiet, Germany's coal mining and steel belt. He was educated at Hittorf–Gymnasium, Recklinghausen, Ruhruniversität Bochum (RUB) and Universität Dortmund (UNIDO), where he read *theoretical medicine, psychology, biology, philosophy* and (for whatever reason) *computer sciences.*

He volunteered as a RA in the *Biomathematics Research Group* from 1987 to 1989, at the former "Max-Planck-Institute for Nutrition Physiology," in Dortmund (since March 1, 1993 renamed to "MPI for Molecular Physiology"), and spent 3 years at the "Systems Analysis Research Group," at the Department of Computer Science of UniDO, where he wrote a particularly unsuccesful thesis on **LEARNING CLASSIFIER SYSTEMs.** In 1995, after 22 semesters, he finally gave up trying to break *Chris Langton's* semester record, and dropped out of the academic circus. Amazingly, he's a fellow at UUNET Deutschland GmbH's Fun & Games division (in more traditional words: the R&D department), currently working on various things in parallel. You may visit his homepage for a mostly complete list at **http://alife.santafe.edu/˜joke/**

His electronic publications range from a voluntary job as senior editor of the **FAQ** in Usenet's **comp.ai.genetic** newsgroup, entitled *The Hitch-Hiker's Guide to Evolutionary Computation,* over many other projects he helped bootstrapping, for example Howard Gutowitz' FAQ on Cellular Automata, available on **USENET** via **comp.theory.cell-automata** ,to about a dozen of so-called "multimediagrams" written in HTML, the language that builds the World-Wide Web. The most useful ones being **ENCORE,** the **EVOLUTIONARY COMPUTATION** Repository Network that today, after several years of weekend hacking, is accessible world-wide. And the latest additions: Zooland, the definite collection of pointers to **ARTIFICIAL LIFE** resources on the 'net; And Webland a KISS model of the Internet at large, including ''guided tours'' across the myriads of info-bits out there.

With Adam Gaffin, a former senior newspaper reporter from Middlesex News, Boston, MA, who is now with Networks World, he edited the most read book on Internet, that was launched by a joined venture of Mitch Kapor's Electronic Frontier Foundation (EFF) and the Apple Computer Library, initially called *Big Dummy's Guide to the Internet* it was later renamed to *EFF's (Extended) Guide to the Internet: A round trip through Global Networks, Life in Cyberspace, and Everything...* If you want to find it, just fire up Netscape and select *About the Internet* from the *Directory* menu...voila! (which means: there you go!)

Since a very special event, he has severe problems to take life seriously, and consequently started signing everything with ''-joke'', while developing a liquid fixation on all flavours of whiskey. He continues to write short stories, novels and works on a diary-like lyrics collection of questionable content, entitled *A Pocketful of Eloquence,* which recently was remaned to *Heartland,* and published on the web as: **http://surf.de.uu.net/heartland/**

He likes *Mickey Rourke's* movies (especially Rumblefish and Barfly), *Edmund Spenser's* medieval poetry, McDonald's Hamburgers, diving into the analysis of complex systems of any kind, (but prefers the long-legged ones) and the books by *Erasmus of Rotterdam, Robert Sheckley, Alexei Panshin,* and, you name it, *Douglas Adams.*

Due to circumstances he leads a life on the edge, is usually single, would love to have children, but has none (he'd know of), has no time to get married, and still doesn't live in Surrey. He is known to reject job offers that come bundled with Porsches and still doesn't own a BMW Z3 roadster, for he recently purchased a 1996 MGF.

NOTABLE WRITINGS

Nothing really worth listing here.

David Beasley,

was born in London, England in 1961. He was educated at Southampton University where he read (for good reasons) Electronic Engineering.

After spending several years at sea, he went to the Department of Computing Mathematics of the University of Wales, Cardiff, where he studied **ARTIFICIAL INTELLIGENCE** for a year. He then went on to write a thesis on **GAs** applied to Digital Signal Processing, and tried to break *Joke's* publications record.

Since a very special event, he has taken over writing this **FAQ,** and consequently started signing everything with ''The FAQmaster'' (He's had severe problems taking life seriously for some time before that, however.) He likes *Woody Allen's* movies, English clothing of medieval times, especially *Marks and Spencer,* hates McDonald's Hamburgers, but occasionally dives into the analysis of complex systems of any kind, (but prefers those with pedals and handlebars) and the books by (of course) *Douglas Adams.*

He is not married, has no children, and also also doesn't live in Surrey.

He now works for Deloitte and Touche Consulting Group, (which used to be a (mostly interesting) software company, Praxis (**http://www.praxis.co.uk/**)) in Bath, England.

NOTABLE WRITINGS

A number of publications related to **GENETIC ALGORITHMs.** The most notable ones being:

A Sequential Niche Technique for Multimodal Function Optimization, Evolutionary Computation, **1**(2) pp 101–125, 1993. Available from **ralph.cs.cf.ac.uk/pub/papers/GAs/seq_niche.ps**

Reducing Epistasis in Combinatorial Problems by Expansive Coding, in S. Forrest (ed), Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan-Kaufmann, pp 400–407, 1993. Available from **ralph.cs.cf.ac.uk/pub/papers/GAs/expansive_coding.ps**

An Overview of Genetic Algorithms: Part 1, Fundamentals, University Computing, **15**(2) pp 58–69, 1993. Alailable from **ENCORE** (See Q15.3) in file: **GA/papers/over93.ps.gz** or from **ralph.cs.cf.ac.uk/pub/papers/GAs/ga_overview1.ps**

An Overview of Genetic Algorithms: Part 2, Research Topics, University Computing, **15**(4) pp 170–181, 1993. Available from Encore (See Q15.3) in file: **GA/papers/over93-2.ps.gz** or from **ralph.cs.cf.ac.uk/pub/papers/GAs/ga_overview2.ps**

THAT'S ALL FOLKS!

"And all our yesterdays have lighted fools the way to dusty death;
out, out brief candle; life's but a walking shadow;
a poor player that struts and frets his hour upon the stage;
and then is heared no more;
it is a tale; told by an idiot,
full of sound and fury,
signifying nothing."

––– Shakespeare, Macbeth