Genetic and Evolutionary Computation: Who, What, Where, When, and Why

Stefano Cagnoni
Department of Computer Engineering
University of Parma, Italy
`cagnoni@ce.unipr.it`

Riccardo Poli
Department of Computer Science
University of Essex, UK
`rpoli@essex.ac.uk`

Abstract

In this paper, we start by providing a gentle introduction to the field of genetic and evolutionary computation, particularly focusing on genetic algorithms, but also touching upon other areas. We then move on to briefly analyse the geographic distribution of research excellence in this field, focusing our attention specifically on Italian researchers. We then present our own interpretation of where and how genetic and evolutionary computation fits in the broader landscape of artificial intelligence research. We conclude by making a prediction of the future impact of this technology in the short term.

Introduction

Darwinian evolution is probably the most intriguing and powerful mechanism of nature mankind has ever discovered. Its power is evident in the impressive level of adaptation reached by all species of animals and plants in nature. It is intriguing because despite its simplicity and randomness it produces incredible complexity in a way that appears to be very directed, almost purposeful. Like for other powerful natural phenomena, it is no surprise then that several decades ago a few brilliant researchers in engineering and computer science started wondering whether they could steal the secrets behind Darwinian evolution and use them to solve problems of practical interest in a variety of application domains. These people were pioneers of a new field, which after more than 30 years from its inception is now big and well established and goes under the name of *Genetic and Evolutionary Computation* (GEC).

An almost endless number of results and applications of evolutionary algorithms have been reported in the literature that show that the ideas of these pioneers were indeed right. Nowadays evolutionary techniques can routinely solve problems in domains such as automatic design, optimisation, pattern recognition, control and many others.

What is Genetic and Evolutionary Computation

What were the main secrets behind Darwinian evolution, that the pioneers of GEC stole to make them the propelling fuel of evolutionary computation processes?

Inheritance: individuals have a genetic representation (in nature, the chromosomes and the DNA) such that it is possible for the offspring of an individual to inherit some of the features of its parent.

Variation: the offspring are not exact copies of the parents, but instead reproduction involves mechanisms that create innovation, as new generations are born.

Natural Selection: individuals best adapted to the environment have longer life and higher chances of mating and spreading their genetic makeup.

Clearly, there is a lot more to natural evolution than these forces. However, like for many other nature-inspired techniques, not all the details are necessary to obtain working models of a natural system. The three ingredients listed above are in fact sufficient to obtain artificial systems showing the main characteristic of natural evolution: *the ability to search for highly fit individuals.*

For all these ingredients (representation, variation, selection) one can focus on different realisations. For example, in nature variation is produced both through mutations of the genome and through the effect of sexually recombining the genetic material coming from the parents when obtaining the offspring's chromosomes (*crossover*). This is why many different classes of evolutionary algorithms have been proposed over the years. So, depending on the structures undergoing evolution, on the reproduction strategies and the variation (or *genetic*) operators adopted, and so on, evolutionary algorithms can be grouped into: Genetic Algorithms (GAs) (Holland, 1975), Genetic Programming (GP) (Koza, 1992), Evolution Strategies (ESs) (Rechenberg, 1973; Schwefel, 1981), etc.

The inventors of these different evolutionary algorithms (or EAs for brevity) have all had to make choices

Table 1: Nature-to-computer mapping at the basis of evolutionary algorithms.

Nature	Computer
Individual	Solution to a problem
Population	Set of solutions
Fitness	Quality of a solution
Chromosome	Representation for a solution (e.g. set of parameters)
Gene	Part of the representation of a solution (e.g. parameter or degree of freedom)
Crossover Mutation	Search operators
Natural Selection	Promoting the reuse of good (sub-)solutions

---

Algorithm 1 Generic evolutionary algorithm.

1: Initialise population
2: Evaluate the fitness of each population member
3: **loop**
4:   Select sub-population for reproduction on the basis of fitness (*Selection*)
5:   Copy some of the selected individuals without change (*Cloning* or *Reproduction*)
6:   Recombine the "genes" of selected parents (*Recombination* or *Crossover*)
7:   Mutate the offspring population stochastically (*Mutation*)
8:   Evaluate the fitness of the new population
9:   Select the survivors on the basis of their fitness
10:   If stopping criterion is satisfied then exit loop
11: **end loop**

---

as to which bits of nature have a corresponding component in their algorithms. These choices are summarised in the nature-to-computer mapping shown in Table 1. That is, the notion of individual in nature corresponds to a tentative solution to a problem of interest in an EA. The fitness (ability to reproduce and have fertile offspring that reach the age of reproduction) of natural individuals corresponds to the objective function used to evaluate the quality of the tentative solutions in the computer. The genetic variation processes of mutation and recombination are seen as mechanisms (search operators) to generate new tentative solutions to the problem. Finally, natural selection is interpreted as a mechanism to promote the diffusion and mixing of the genetic material of individuals representing good quality solutions, and, therefore, having the potential to create even fitter individuals (better solutions).

Despite their differences, most EAs have the general form shown in Algorithm 1, although not all the steps in Algorithm 1 are present in all evolutionary algorithms. For example, in modern GAs (Mitchell, 1996) and in GP phase (a) is part of phases (b) and (c), while phase

(f) is absent. This algorithm is said to be *generational* because there is no overlap between generations (i.e. the offspring population always replaces the parent population). In generational EAs cloning is used to simulate the survival of parents for more than one generation.

In the following we will analyse the various components of an EA in more detail, mainly concentrating on the genetic algorithm, although most of what we will say also applies to other paradigms.

Representations

Traditionally, in GAs, solutions are encoded as binary strings. Typically an adult individual (a solution for a problem) takes the form of a vector of numbers. These are often interpreted as parameters (for a plant, for a design, etc.), but in combinatorial optimisation problems these numbers can actually represent configurations, choices, schedules, paths and so on. Anything that can be represented on a digital computer can also be represented in a GA using a binary representation. This is why, at least in principle, GAs have a really broad applicability. However, other, non-binary representations are available, which may be more suitable, e.g., for problems with real-valued parameters.

Because normally the user of a GA has no ideas as to what constitutes a good initial set of choices/parameters for adult individuals (tentative solutions to a problem), the chromosomes to be manipulated by the GA are normally initialised in an entirely random manner. That is, the initial population is a set of random binary strings or of random real-valued vectors.

Selection in GAs

Selection is the operation by which individuals (i.e. their chromosomes) are selected for mating or cloning. To emulate natural selection, individuals with a higher fitness should be selected with higher probability. There are many models of selection. We briefly describe three of the most frequently used ones below.

Fitness proportionate selection, besides being the most direct translation into the computational model of the principles of evolution, is probably the most widely used selection scheme. This works as follows. Let N be the population size, f_i the fitness of individual i, and $\bar{f} = \frac{1}{N} \sum_j f_j$ the average population fitness. Then, in fitness proportionate selection, individual i is selected for reproduction with a probability:

$$p_i = \frac{f_i}{\sum_j f_j} = \frac{f_i}{\bar{f} N}.$$

In normal GAs populations are not allowed to grow or shrink, so N individuals have to be selected for reproduction. Therefore, the *expected number* of selected copies of each individual is:

$$N_i = p_i N = f_i / \bar{f}.$$

So, individuals with an above-average quality ($f_i > \bar{f}$) tend to be selected more than once for mating or

cloning, while individuals below the average tend not to be used.

Tournament selection, instead, works as follows. To select an individual, first a group of T ($T \geq 2$) random individuals is created. Then the individual with the highest fitness in the group is selected, the others are discarded (tournament).

Another alternative is *rank selection* where individuals are first sorted (ranked) on the ground of their fitness, so that if an individual i has fitness $f_i > f_j$ than its rank is $i < j$. Then each individual is assigned a probability of being selected p_i taken from a given distribution (typically a monotonic decreasing function), with the constraint that $\sum_i p_i = 1$.

Operators

Evolutionary algorithms work well only if their genetic operators allow an efficient and effective search of the space of tentative solutions.

One desirable property of recombination operators is to guarantee that two parents sharing a useful common characteristic always transmit such a characteristic to their offspring. Another important property is to also guarantee that different characteristics distinguishing two parents may be all inherited by their offspring. For binary GAs there are many crossover operators with these properties.

One-point crossover, for example, aligns the two parent chromosomes (bit strings), then cuts them at a randomly chosen common point and exchanges the right-hand-side (or left-hand-side) sub-chromosomes (see Figure 1(a)). In *two-point crossover* chromosomes are cut at two randomly chosen crossover points and their ends are swapped (see Figure 1(b)). A more modern operator, *uniform crossover*, builds the offspring, one bit at a time, by selecting randomly one of the corresponding bits from the parents (see Figure 1(c)).

Normally, crossover is applied to the individuals of a population with a constant probability p_c (often $p_c \in [0.5, 0.8]$). Cloning is then applied with a probability $1 - p_c$ to keep the number of individuals in each generation constant.

Mutation is the second main genetic operator used in GAs. A variety of mutation operators exist. Mutation typically consists of making (usually small) alterations to the values of one or more genes in a chromosome. Often mutation is applied to the individuals produced by crossover and cloning before they are added to the new population. In binary chromosomes mutation often consists of inverting random bits of the genotypes (see Figure 2). The main goal with which mutation is applied is the preservation of population diversity, since diversity prevents the evolutionary search from stagnating. However, due to its random nature, mutation may have disruptive effects onto evolution if it occurs too often. Therefore, in GAs, mutation is usually applied to genes with a very low probability.

In real-valued GAs chromosomes have the form $x = \langle x_1, \ldots, x_\ell \rangle$ where each gene x_i is represented by a
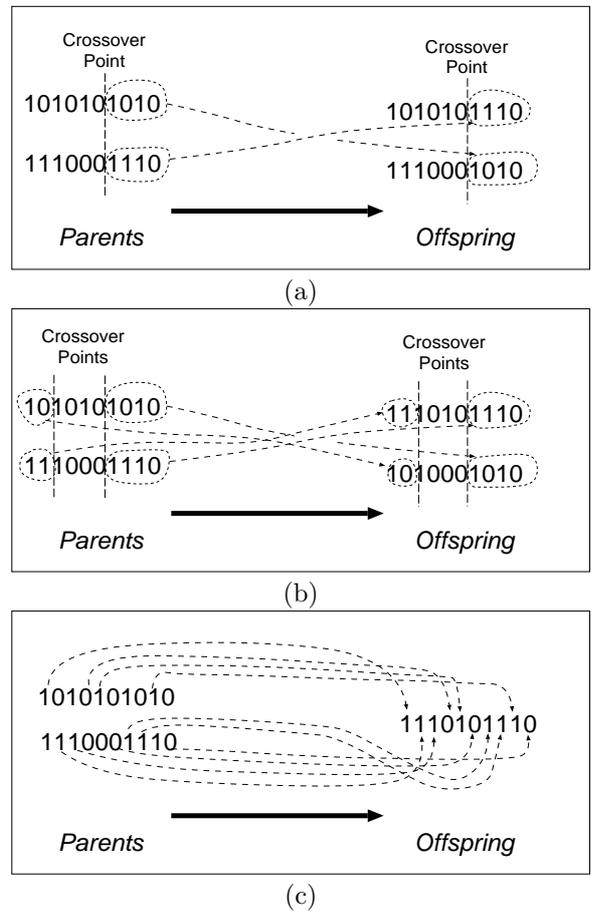

(a)

(b)

(c)

Figure 1: Three crossover operators for binary GAs: (a) one point crossover, (b) two point crossover, (c) uniform crossover.
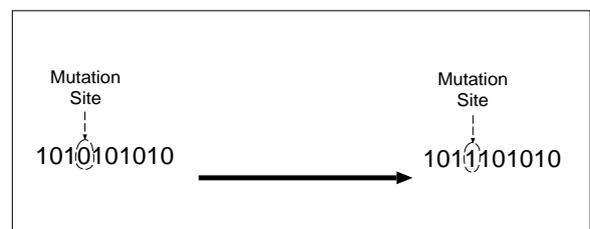

Figure 2: Bitwise mutation in binary GAs.

floating-point number. In these GAs crossover is often seen as an interpolation process in a multi-dimensional Euclidean space. So, the components of the offspring o are calculated from the corresponding components of the parents p' and p'' as follows:

$$o_i = p'_i + r(p''_i - p'_i)$$

where r is a random number in the interval $[0, 1]$ (see Figure 3(a)). Alternatively crossover can be seen as the exploration of a multi-dimensional hyper-parallelepiped defined by the parents (see Figure 3(b)), that is the components o_i are chosen uniformly at random within the intervals

$$[\min(p'_i, p''_i), \max(p'_i, p''_i)].$$

Mutation is often seen as the addition of a small random variation (e.g. Gaussian noise) to a point in a multi-dimensional space (see Figure 3(c)).

Other GEC Paradigms

As mentioned before, the principles on which GAs are based are also shared by many other EAs. However, the use of different representations and operators has led to the development of a number of paradigms, each having its own peculiarities. With no pretence of being exhaustive, in the following we will briefly mention two important paradigms, other than GAs.

Genetic programming (Koza, 1992; Langdon and Poli, 2002) is a variant of GA in which the individuals being evolved are syntax trees, typically representing computer programs. The trees are created using user-defined primitive sets, which typically include input variables, constants and a variety of functions or instructions. The syntax trees are manipulated by specialised forms of crossover and mutation that guarantee the syntactic validity of the offspring. The fitness of the individual trees in the population is evaluated by running the corresponding programs (typically multiple times, for different values of their input variables).

Evolution strategies (Rechenberg, 1973; Schwefel, 1981) are real-valued EAs where mutation is the key variation operator (unlike GAs where crossover plays that role). Mutation typically consists of adding zero-mean Gaussian deviates to the individuals being optimised, with the mutation's standard deviation being varied dynamically so as to maximise the performance of the algorithm.

State of the art

Popularity of the Field

Among AI and AI-related disciplines, GEC is presently one of the most active. This is testified, for example, by:

- the numerous dedicated conferences and workshops (around 20 annual or biannual events) including the Genetic and Evolutionary Computation Conference (GECCO), the largest conference in the field, with around 600 attendees, organised by ACM SigEvo,
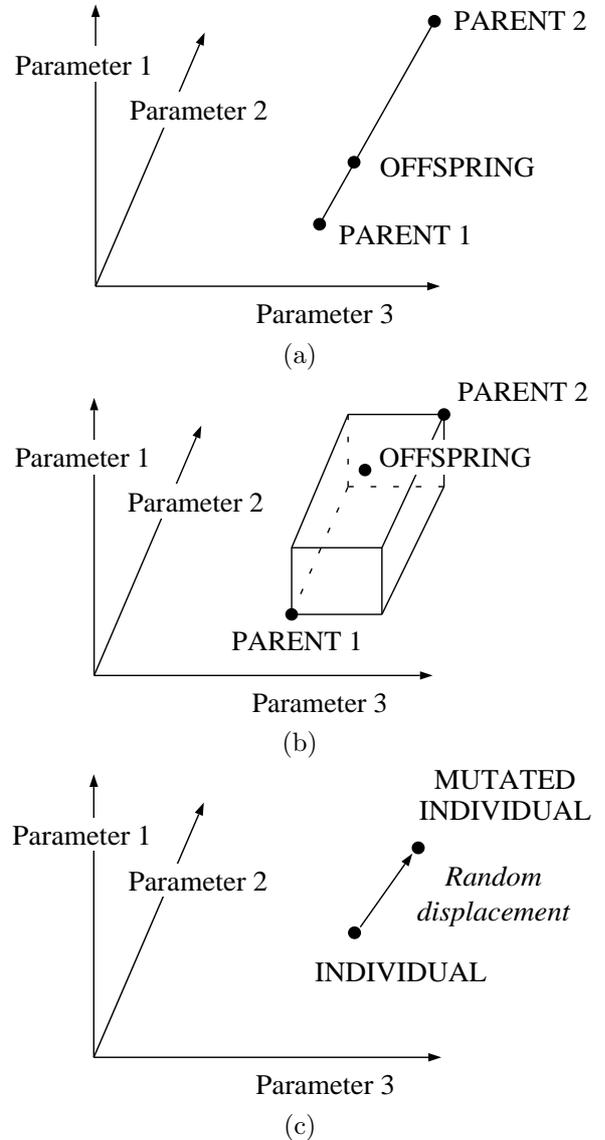
(a)

(b)

(c)

Figure 3: Crossover operators (a)—(b) and mutation (c) for real-valued GAs.

and the large IEEE Congress on Evolutionary Computation (CEC);

- the several dedicated journals including Evolutionary Computation from MIT Press (the oldest journal in the field), the Journal on Genetic Programming and Evolvable Machines from Kluwer (which is specialised on Genetic Programming and evolvable hardware), and the IEEE Transactions on Evolutionary Computation;

- the numerous large bibliographies of GEC literature, including, for example, the large collection of AI bibliographies in "The Collection of Computer Science Bibliographies" available at `http://liinwww.ira.uka.de/bibliography/Ai/`, where the "Genetic Programming bibliography" by itself, which covers only papers on that specific GEC paradigm, is the third largest among the ones which are still updated on a regular basis (with 4919 papers at the time of writing), fifth overall;

- the constant stream of new books and doctoral theses on the subject.

International Situation

It is very difficult to say precisely when and where the field of Genetic and Evolutionary Computation was originated. Indeed, especially for the least recent evolutionary paradigms, as often happens, seminal ideas can be found in papers by authors who were not the ones who have finally popularised them (see (Fogel, 1998) for a comprehensive collection). Considering the latter as the actual 'fathers' of GEC, Genetic Algorithms (Holland, 1975), as well as Genetic Programming (Koza, 1992), were first studied and developed as independent research topics in the United States, while Evolution Strategies (Rechenberg, 1973; Schwefel, 1981), developed in Europe, already contain ideas which will be used later in GAs and GP. So, we can say that Germany and the United States, are the countries where it all started. However, GEC research is today performed in many, many countries worldwide. For example, the two traditional major actors in GEC (USA and Germany) have now been joined at the top by the United Kingdom, where research in AI and related fields has always been extremely active.

From the point of view of research topics, after the typical pioneering times of rapid development in which any empirical study or application of any paradigm to any problem was considered interesting and publishable in its own right, research in GEC has become more mature and structured into well defined topics. Such a classification comprises a set of theoretical topics regarding basic studies on the different GEC paradigms (Genetic Algorithms, Genetic Programming, Evolution Strategies, Evolutionary Programming, Particle Swarm Optimisers, Classifier Systems, Ant Algorithms, Artificial Immune Systems, etc.). Other research areas span more than one of these fundamental paradigms (Co-evolution, Evolutionary Multi-Objective Optimisation, Evolutionary Combinatorial Optimisation, Hybridisation, Evolutionary Meta-heuristics, Memetic Algorithms, etc.), and these are also hot theoretical research fields. Finally, a large set of more application-oriented topics are popular research areas, such as Evolvable Hardware, Evolutionary Robotics, Evolutionary Image Analysis and Signal Processing and, more generally, Real-World Applications at large. Some of these areas have recently grown significantly and can be considered independent GEC subfields in their own right.

GEC research in Italy

On a national basis, as it is, unfortunately, not uncommon, we have an anomalous situation in which the main Italian researchers on the field work for foreign institutions. For example, Riccardo Poli (UK) is a world leader in the field of Genetic Programming (being second only to John Koza for number of publications in this area), Marco Dorigo (Belgium) is the inventor and world leader of Ant Colony Optimisation, Marco Tomassini (Switzerland) is a leader on evolutionary algorithms and complex systems, etc.

However, progressively this situation has being balanced by an increasingly active national community, in which more than ten groups are specifically active in the field (at the Universities or Polytechnic Schools of Milan, Turin, Parma, Venice, Naples, Salerno, Calabria, Catania, to name a few), which made it possible to organise successful first edition of GSICE (Italian Workshop on Evolutionary Computation) in 2005 in Milan, which will be followed by the upcoming editions in Siena (2006) and Catania (2007).

The main research topics on which the activity of Italian researchers in GEC is focused are: genetic programming theory (R. Poli, Tomassini, Vanneschi), learning classifier systems (Lanzi), ant algorithms (Dorigo, Gambardella), particle swarm optimisation (R. Poli), evolutionary models in artificial life (Nicosia, I. Poli), hybrid systems (Tettamanzi), co-evolution (Cagnoni, Vanneschi), evolutionary robotics (Floreano, Nolfi), evolvable hardware (Squillero). The main application fields are biology (Marchiori), game strategy (Squillero), finance (Tettamanzi), computer vision and pattern recognition (Cagnoni, Cordella, De Falco, R. Poli).

Open problems and research directions

Despite its increasing degree of maturity, there are still many partially unanswered questions which face researchers in GEC. Here we limit ourselves to mention only a few of the main open challenges:

- How do we classify GEC algorithms? When do we expect the behaviour and performance of two different evolutionary systems to be qualitatively (and maybe at some point quantitatively) similar and why?

- How do we classify problems? When do we expect the performance of a particular algorithm on two dif-

ferent problems to be the substantially the same and why?

- Although the development of mathematical models of evolutionary equations has been rapid there remains a disquieting lack of tools with which we obtain solutions to evolution equations. In addition, these models have immense numbers of degrees of freedom, which makes them hard to simulate even for the most powerful computers. So, mathematical models can shed only some light on EA dynamics.

- How can we develop models of EAs that can provide theoretically-sound recipes for practitioners, such as which operators, fitness function, search algorithm, population size, number of generations, number of runs, crossover probability, etc. one should use for a given problem or a given class of problems.

Interactions with other AI disciplines and other research areas

GEC is intrinsically a transversal field of research, since, on the one side, its techniques are based on biological models, but also, on the other side, since it provides a set of tools which can be effectively applied to other disciplines. Quite naturally there are several examples of synergetic applications (Tettamanzi and Tomassini, 2001) of GEC techniques along with other techniques which are comprised in the set of disciplines (Neural Networks, Fuzzy Sets, Probabilistic Networks) usually termed as Computational Intelligence (CI).

CI is considered by many the new modern AI. However, although some GEC researchers might object to this viewpoint, we believe that most of the work currently going on in GEC can be seen as AI. In particular, when EAs are successfully used in practical applications, in many cases a lot of the credit for the success goes the specialised representation, operators and fitness function designed to tackle the problem, rather than the fact that this or that EA was used to perform the search. That is, the search is often successful thanks to the good knowledge engineering efforts of the users of the EA. This should not come as a surprise: after all AI people has always known that a good representation, good expansion operators and good heuristics can tame search and avoid the problems inherent in exponentially large search spaces. GEC researchers, however, have started accepting these good-old-fashioned AI guidelines after painfully digesting a now-famous result that goes under the name of No-free Lunch Theorem for search (Wolpert and Macready, 1997).

Applications

Getting machines to produce human-like results is the reason for the existence of AI and machine learning. However, it has always been very difficult to assess how much progress these fields have made towards their ultimate goal. Turing understood the need to evaluate objectively the behaviour exhibited by machines, to avoid human biases when assessing their intelligence. This led him to propose an imitation game, now known as the Turing test for machine intelligence. Unfortunately, the Turing test is not usable in practice, and so, it has become clear that there is a need for more workable objective tests for progress.

John Koza (Koza et al., 1999) recently proposed to shift the attention from the notion of intelligence to the notion of *human competitiveness*. An automatically-created result is considered "human-competitive" if it satisfies at least one of the eight criteria below:

1. The result was patented as an invention in the past, is an improvement over a patented invention, or would qualify today as a patentable new invention.

2. The result is equal to or better than a result that was accepted as a new scientific result at the time when it was published in a peer-reviewed scientific journal.

3. The result is equal to or better than a result that was placed into a database or archive of results maintained by an internationally recognised panel of scientific experts.

4. The result is publishable in its own right as a new scientific result, independent of the fact that the result was mechanically created.

5. The result is equal to or better than the most recent human-created solution to a long-standing problem for which there has been a succession of increasingly better human-created solutions.

6. The result is equal to or better than a result that was considered an achievement in its field at the time it was first discovered.

7. The result solves a problem of indisputable difficulty in its field.

8. The result holds its own or wins a regulated competition involving human contestants (in the form of either live human players or human-written computer programs).

Over the years, a list of tens of results have passed the human-competitiveness test (see (Koza and Poli, 2005) for a recent list). Also, since 2004, a competition is held annually at GECCO (termed the "Human-Competitive awards - the 'Humies',"). The prize (\$ 10,000) is awarded to automatically-created applications which have produced results which are equivalent to human achievements or, better, are unpaired by humans. The Gold Prizes in 2004 and 2005 were awarded to applications of GEC to high-tech fields such as an antenna for deployment on NASA's Space Technology 5 Mission, automatic quantum computer programming, two-dimensional photonic crystals design, applications to attosecond dynamics of high-harmonic generation, shaped-pulse optimisation of coherent soft-x-rays. The 2006 competition is still to be held at the time of writing.

Some pre-2004 human-competitive results include:

- Creation of a better-than-classical quantum algorithm for Grover's database search problem
- Creation of a quantum algorithm for the depth-two AND/OR query problem that is better than any previously published result
- Creation of a soccer-playing program that won its first two games in the Robo Cup 1997 competition
- Creation of four different algorithms for the transmembrane segment identification problem for proteins
- Creation of a sorting network for seven items using only 16 steps
- Synthesis of 60 and 96 decibel amplifiers
- Synthesis of analog computational circuits for squaring, cubing, square root, cube root, logarithm, and Gaussian functions
- Synthesis of a real-time analog circuit for time-optimal control of a robot
- Synthesis of an electronic thermometer
- Creation of a cellular automata rule for the majority classification problem that is better than the Gacs-Kurdyumov-Levin (GKL) rule and all other known rules written by humans
- Synthesis of topology for a PID-D2 (proportional, integrative, derivative, and second derivative) controller
- Synthesis of NAND circuit
- Simultaneous synthesis of topology, sizing, placement, and routing of analog electrical circuits
- Synthesis of topology for a PID (proportional, integrative, and derivative) controller
- Synthesis of a voltage-current conversion circuit
- Creation of PID tuning rules that outperform the Ziegler-Nichols and Astrom-Hagglund tuning rules
- Creation of three non-PID controllers that outperform a PID controller that uses the Ziegler-Nichols or Astrom-Hagglund tuning rules

Even from the short list provided, which is far from being exhaustive, nor sufficient to fully describe their potential, the use of GEC techniques as invention machines appears to be effective, and subject to further increase in competitiveness in a future, which the following closing section tries to forecast.

Conclusions

Let try to look a bit ahead into a not too distant future, say 2012, and let us make some predictions.

We start from some simple back-of-an-envelope calculations. In 2012 CPUs will be 25 times faster than today. A Beowulf desktop computer (12 CPUs) will be 300 times faster than current PCs. A Beowulf tower computer (96 CPUs) will be 2400 times faster than current PCs and approx 10 times faster than the 1000 node supercomputer Koza used to obtain his human competitive results (new inventions). These were produced in approx 1 week of computer time.

It follows from this that in 2012 it will be possible to produce patentable new inventions in a day on a 96 node Beowulf workstation! So, by 2012 it is not unthinkable that EAs will be used routinely as invention machines, design machines, optimisers and problem solvers.

So, GEC has effectively started fulfilling the AI dream by providing us with a systematic method, based on Darwinian evolution, for getting computers to automatically solve difficult problems for us. To do so, EAs simply require a high-level statement of what needs to be done (and enough computing power). Today GEC certainly cannot produce computers that would pass the full Turing test for machine intelligence, but GEC has been able to solve tens of difficult problems with human-competitive results, and we should expect to see this trend accelerate.

These are small steps towards fulfilling the founders of AI dreams, but they are also early signs of things to come. By 2012 EAs and AI techniques will be able to routinely and competently solve important problems for us in a variety of specific domains of application, becoming essential collaborators for many of human activities.

This will be a remarkable step forward towards achieving true, human-competitive machine intelligence.

References

Fogel, D. B., editor (1998). *Evolutionary Computation. The Fossil Record. Selected Readings on the History of Evolutionary Computation.* IEEE Press.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, USA.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Natural Selection.* MIT Press, Cambridge, MA, USA.

Koza, J. R., Bennett III, F. H., and Stiffelman, O. (1999). Genetic programming as a Darwinian invention machine. In Poli, R., Nordin, P., Langdon, W. B., and Fogarty, T. C., editors, *Genetic Programming, Proceedings of EuroGP'99*, volume 1598 of *LNCS*, pages 93–108, Goteborg, Sweden. Springer-Verlag.

Koza, J. R. and Poli, R. (2005). Genetic programming. In Burke, E. K. and Kendall, G., editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, chapter 5. Springer.

Langdon, W. B. and Poli, R. (2002). *Foundations of Genetic Programming.* Springer-Verlag.

Mitchell, M. (1996). *An introduction to genetic algorithms.* Cambridge MA: MIT Press.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann–Holzboog, Stuttgart.

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

Tettamanzi, A. and Tomassini, M. (2001). *Soft Computing: Integrating Evolutionary, Neural and Fuzzy Systems*. Springer, Berlin, Heidelberg, New York, New York.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Pointers to Further Reading in GEC

- David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts, 1989.
  A classic book on genetic algorithms and classifier systems.

- David E. Goldberg. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Boston, 2002.
  An excellent, long-awaited follow up of Goldberg's first book.

- Melanie Mitchell, *An introduction to genetic algorithms*, A Bradford Book, MIT Press, Cambridge, MA, 1996.
  A more modern introduction to genetic algorithms.

- John H. Holland, *Adaptation in Natural and Artificial Systems*, second edition, A Bradford Book, MIT Press, Cambridge, MA, 1992.
  Second edition of a classic from the inventor of genetic algorithms.

- Thomas Bäck and Hans-Paul Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.
  A good introduction to parameter optimisation using EAs.

- T. Bäck, D. B. Fogel and T. Michalewicz, *Evolutionary Computation 1: Basic Algorithms and Operators*, Institute of Physics Publishing, 2000.
  A modern introduction to evolutionary algorithms. Good both for novices and more expert readers.

- John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.
  The bible of genetic programming by the founder of the field. Followed by GP II (1994), GP III (1999) and GP IV (forthcoming).

- Wolfgang Banzhaf, Peter Nordin, Robert E. Keller and Frank D. Francone, *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications*, Morgan Kaufmann, 1998.
  An excellent textbook on GP.

- W. B. Langdon and Riccardo Poli, *Foundations of Genetic Programming*, Springer, Feb 2002.
  The only book entirely devoted to the theory of GP and its relations with the GA theory.

- Proceedings of the Genetic and Evolutionary Computation Conference.
  Born in 1999 from the "recombination" of the International Conference on Genetic Algorithms and the Genetic Programming Conference, GECCO is the largest conference in the field.

- Proceedings of the Foundations of Genetic Algorithms (FOGA) workshop.
  FOGA is a biannual, small but very-prestigious and highly-selective workshop. It is mainly devoted to the theoretical foundations of EAs.

- Proceedings of the Congress on Evolutionary Computation (CEC).
  CEC is large conference under the patronage of IEEE.

- Proceedings of Parallel Problem Solving from Nature (PPSN).
  This is a large biannual European conference, probably the oldest of its kind in Europe.

- Proceedings of the European Conference on Genetic Programming.
  EuroGP was the first European event entirely devoted to Genetic Programming. Run as a workshop in 1998 and 1999, it became a conference in 2000. It has now reached its tenth edition.

Some Web Resources

- http://www.genetic-programming.com/ and http://www.genetic-programming.org/
  Genetic Programming Inc.,

- http://www.cs.bham.ac.uk/~wbl/biblio/README.html
  http://liinwww.ira.uka.de/bibliography/Ai/genetic.programming.html
  GP bibliography maintained by W. B. Langdon

- http://www.geneticprogramming.com/
  The Genetic Programming Notebook

- http://evonet.lri.fr/CIRCUS2/node.php?node=1
  The EvoNet online tutorial