Editorial

Riccardo Poli

Department of Computer Science, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, UK
Email: rpoli@essex.ac.uk

Stefano Cagnoni

Department of Computer Engineering, University of Parma, 43100 Parma, Italy
Email: cagnoni@ce.unipr.it

1. 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 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. Those people were pioneers of a new field which, after more than 30 years from its inception, is now big, 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, showing 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. Until recently, however, only very occasionally could one claim that GEC techniques approached the performance of human experts in these same domains, particularly in the case of large scale applications and complex engineering problems. This is why, initially, successful applications of GEC techniques to the fields of computer vision, image analysis, and signal processing were few and far in between. Towards the late 1990s, however, the research interest in these areas seemed to be rapidly growing, and the time seemed right for the creation of an infrastructure that could foster the interaction between researchers in this area. This is what led, in early 1998, the two editors of this special issue, together with people from various other European institutions, to create a working group of the European Network of Excellence in Evolutionary Computation, entirely devoted to the applications of evolutionary algorithms to image analysis and signal processing. The working group organises a regular meeting, the European Workshop on Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP), which reached its fifth edition this year. This event gives European and non-European researchers, as well as people from industry, an opportunity to present their latest research, discuss current developments and applications, besides fostering closer interaction between members of the GEC, image analysis, and signal processing scientific communities. However, the event is, and intends to remain, a workshop. Therefore, the work presented there can never have the depth allowed by more substantial and mature archival publications such as this journal.

This special issue of EURASIP JASP on GEC for signal processing and image analysis, being the first of its kind, has offered computer scientists and engineers from around the world a unique opportunity to submit their best mature research for inclusion in this unified, high-quality "venue." The timing of this special issue could not have been better; well over thirty papers were submitted by contributors from around the world. The papers were reviewed by a pool of over thirty international expert reviewers. Only about one third passed our strict criteria for acceptance and are now in this volume.

The rest of this editorial is organised as follows. In Section 2, we will provide a gentle introduction to the basics of evolutionary computation. In Section 3, we describe each of the papers present in this special issue, briefly summarising, for each one, the problem considered and the evolutionary techniques adopted to tackle it. In Section 4, we provide our final remarks and acknowledgments, while in the appendix, we give a brief commented bibliography with suggested further reading.

2. EVOLUTIONARY COMPUTATION: THE BASICS

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 offsprings 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 main 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, and 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 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 several evolutionary paradigms: genetic algorithms (GAs) [1], genetic programming (GP) [2], evolution strategies (ESs) [3, 4], and so forth.

All the inventors of these different evolutionary algorithms (EAs) have to make choices 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 offsprings 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).

TABLE 1: Nature-to-computer mapping at the basis of EAs.

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	Search operator
Mutation	Search operator
Natural selection	Promoting the reuse of good (sub-)solutions

Despite their differences, most EAs have the following general form.

(1) Initialise population and evaluate the fitness of each population member.
(2) Repeat.
   (a) Select subpopulation for reproduction on the basis of fitness (*selection*).
   (b) Copy some of the selected individuals without change (*cloning* or *reproduction*).
   (c) Recombine the "genes" of selected parents (*recombination* or *crossover*).
   (d) Mutate the mated population stochastically (*mutation*).
   (e) Evaluate the fitness of the new population.
   (f) Select the survivors based on fitness.

Not all these steps are present in all EAs. For example, in modern GAs [5] 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 GA, although most of what we will say also applies to other paradigms.

2.1. 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 nonbinary representations are available, which may be more suitable, for example, 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.

2.2. 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, some of which, despite fitting well the biologically inspired computational model and producing effective results, are not biologically plausible. We briefly describe them below.

Fitness proportionate selection, besides being the most direct translation into the computational model of the probabilistic 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 $\bar{f} = (1/N) \sum_j f_j$ the average population fitness. Then, in fitness proportionate selection, individual i is selected for reproduction with a probability

$$p_i = \frac{f_i}{\sum_j f_j} = \frac{f_i}{\bar{f} N}. \tag{1}$$

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

$$N_i = p_i N = \frac{f_i}{\bar{f}}. \tag{2}$$

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 $f_i > f_j$, then 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 rapidly decreasing function), with the constraint that $\sum_i p_i = 1$.

2.3. Operators

EAs 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
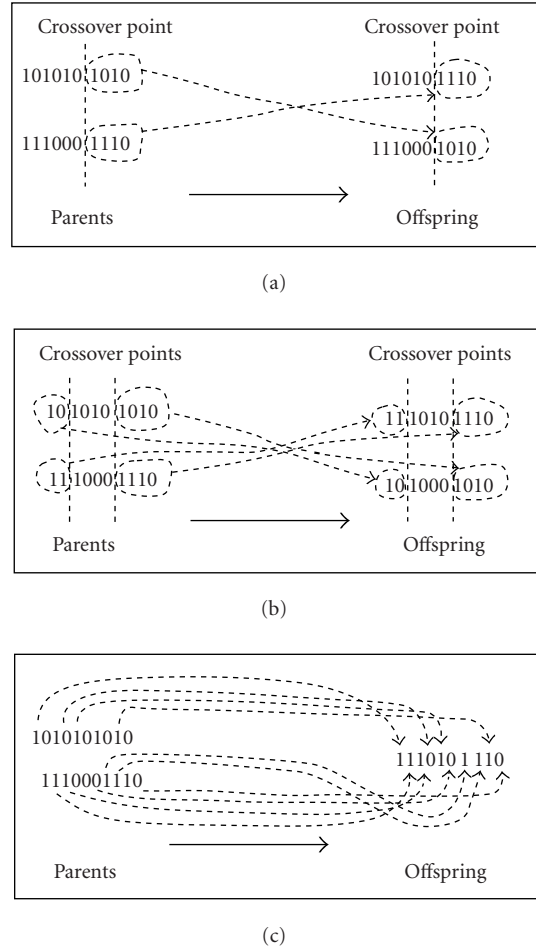

(a)

(b)

(c)

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

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) subchromosomes (see Figure 1a). In *two-point crossover*, chromosomes are cut at two randomly chosen crossover points and their ends are swapped (see Figure 1b). A more modern operator, *uniform crossover*, builds the offspring, one bit at a time, by randomly selecting one of the corresponding bits from the parents (see Figure 1c).

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
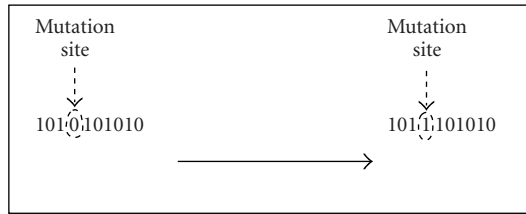
FIGURE 2: Bitwise mutation in binary GAs.

values of one or more genes in a chromosome. Mutation is often 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 preservation of diversity, which helps GAs to explore as much of the search space as possible. 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 floating-point number. In these GAs, crossover is often seen as an interpolation process in a multidimensional Euclidean space. So, the components of the offspring o are calculated from the corresponding components of the parents p' and p'' as follows:

$$o_i = p_i' + r(p_i'' - p_i'), \qquad (3)$$

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

$$[\min(p_i', p_i''), \max(p_i', p_i'')]. \qquad (4)$$

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

2.4. 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 shortly mention those paradigms, other than GAs, that are used in the papers included in this special issue.
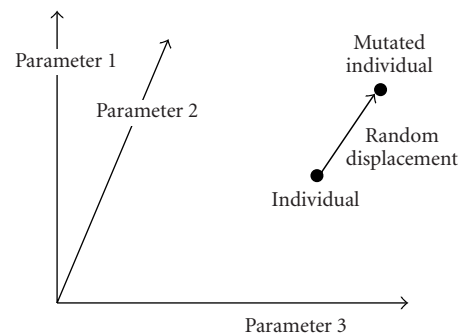
Genetic programming [2, 6] is a variant of GA in which the individuals being evolved are syntax trees, typically representing computer programs. The trees are created using user-defined primitive sets, which typically include input variables, constants, and a variety of functions or instructions.
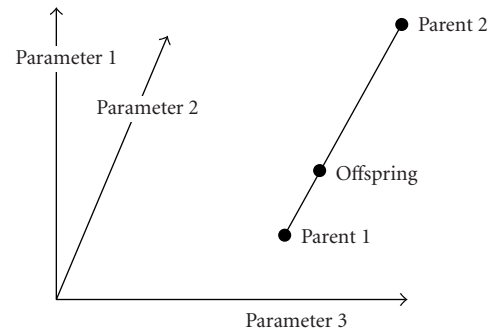
(a)
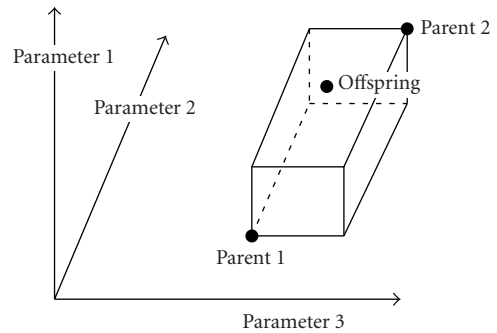
(b)

(c)

FIGURE 3: (a), (b) crossover operators and (c) mutation for real-valued GAs.

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 [3, 4] 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 standard deviation being varied dynamically so as to maximise the performance of the algorithm.

Artificial immune systems (see [7, Part III, Chapters 10–13] or [8] for an extensive introduction) are distributed computational systems inspired by biological immune systems, which can recognise patterns and can remember previously seen patterns in an efficient and effective way. These systems are very close relatives of EAs (sometimes involving an evolutionary process in their inner mechanics) although they use a different biological metaphor.

3. THE PAPERS IN THIS SPECIAL ISSUE

In their paper entitled *Blind search for optimal Wiener equalizers using an artificial immune network model*, Attux et al. exploit recent advances in the field of artificial immune systems to obtain optimum equalisers for noisy communication channels, using a technology that does not require the availability of clean samples of the input signal. This approach is very successful in a variety of test equalisation problems. The approach is also compared with a more traditional EA, a GA with niching, showing superior performance.

The paper by Dunn and Olague, entitled *Evolutionary computation for sensor planning*, shows how well-designed evolutionary computation techniques can solve the problem of optimally specifying sensing tasks for a workcell provided with multiple manipulators and cameras. The problem is NP hard, effectively being a composition of a set partitioning problem and multiple traveling salesperson problems. Nonetheless, thanks to clever representations and the use of evolutionary search, this system is able to solve the problem, providing solutions of quality very close to that of the solutions obtained via exhaustive search, but in a tiny fraction of the time.

The paper entitled *An evolution approach for joint blind multichannel estimation and order detection* by Fangjiong et al. presents a method for the detection of the order and the estimation of the parameters of a single-input multiple-output channel. The method is based on a hybrid GA with specially designed operators. The method shows performances comparable with existing closed-form approaches which, however, are much more restricted in that they either assume that the channel order is known or treat the problems of order estimation and parameter estimation separately.

In *Application of evolution strategies to the design of tracking filters with a large number of specifications*, Herrero et al. attack the problem of tracking civil aircrafts from radar information within the extremely tight performance constraints imposed by a civil aviation standard. They use interactive multiple mode filters optimised by using an ES and a multiobjective optimisation approach obtaining a high-performance aircraft tracker.

Making EAs more at hand and easy to apply for general practitioners by self-tuning their parameters is one of the main aims with which Pignalberi et al. developed GASE, a GA-based tool for range image segmentation. The system, along with some practical results, is described in the paper *Tuning range image segmentation by genetic algorithm*. A multiobjective fitness function is adopted to take into consideration problems that are typically encountered in range image segmentation.

The paper *Parameter estimation of a plucked string synthesis model using a genetic algorithm with perceptual fitness calculation* describes the use of GAs to estimate the control parameters for a widely used plucked string synthesis model. Using GAs, Riionheimo and Välimäki have been able to automate parameter extraction, which had been formerly achieved only through semiautomatic approaches, obtaining comparable results, both in quantitative and in qualitative terms. An interesting feature of the approach is the inclusion of knowledge about perceptual properties of the human hearing system into the fitness function.

Schell and Uhl compare results obtained with a GA-based approach to the near-best-basis (NBB) algorithm, a well-known suboptimal algorithm for wavelet packet decomposition. In their paper *Optimization and assessment of wavelet packet decompositions with evolutionary computation*, they highlight the problem of finding good cost functions in terms of correlation with actual image quality. They show that GAs provide lower-cost solutions that, however, provide lower-quality images than NBB.

In the paper entitled *On the use of evolutionary algorithms to improve the robustness of continuous speech recognition systems in adverse conditions*, Selouani and O'Shaughnessy show how a GA can tune a system based on state-of-the-art speech recognition technology so as to maximise its recognition accuracy in the presence of severe noise. This hybrid of evolution and conventional signal processing algorithms amply outperforms nonadaptive systems. The EA used is a GA with real-coded representation, rank selection, a heuristic type of crossover, and a nonuniform mutation operator.

The paper *Evolutionary techniques for image processing a large dataset of early Drosophila gene expression* by Spirov and Holloway describes an evolutionary approach to image processing to process confocal microscopy images of patterns of activity for genes governing early Drosophila development. The problem is approached using plain GAs, a simplex approach, and a hybrid between these two.

The use of GAs to track time-varying systems based on recursive models is tackled in *A comparison of evolutionary algorithms for tracking time-varying recursive systems*. The paper first compares a plain GA with a GA variant, called "random immigrant strategy," showing that the latter performs better in tracking time-varying systems even if it has problems with fast-varying systems. Finally, a hybrid combination of GAs and local search that is able to tackle even such hard tasks is proposed.

Zhang et al., in their paper *A domain-independent window approach to multiclass object detection using genetic programming*, propose an interesting approach in which GP is used to both detect and localise features of interest. The approach is compared with a neural network classifier, used as reference, showing that GP evolved programs can provide significantly lower false-alarm rates. Within the proposed approach, the choice of the primitive set is also discussed, comparing results obtained with two different sets: one comprises only the four basic arithmetical operators, and

the other includes also transcendental functions. The results reported in the paper provide interesting clues to practitioners that would like to use GP to tackle image processing tasks.

4. CONCLUSIONS

The guest editors hope that the readership of the journal will enjoy reading the papers in this special issue as we did ourselves. We hope that the broadness of domains to which EAs can be applied, demonstrated by the contents of this issue, will convince other researchers in image analysis and signal processing to get acquainted with the exciting world of evolutionary computation and to apply its powerful techniques to solve important new and old problems in these areas.

APPENDIX

POINTERS TO FURTHER READING IN GEC

1. 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.

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

3. Melanie Mitchell, *An introduction to genetic algorithms*, A Bradford Book, MIT Press, Cambridge, MA, 1996. A good introduction to genetic algorithms.

4. 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.

5. Thomas Bäck and Hans-Paul Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, 1993. A good introduction to parameter optimisation using EAs.

6. 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.

7. 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).

8. 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.

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

10. Proceedings of the International Conference on Genetic Algorithms (ICGA). ICGA is the oldest conference on EAs.

11. Proceedings of the Genetic Programming Conference. This was the first conference entirely devoted to GP.

12. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO). Born in 1999 from the "recombination" of ICGA and the GP conference mentioned above, GECCO is the largest conference in the field.

13. 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.

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

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

16. Proceedings of the European Workshop on Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP). This is a small workshop, reaching its fifth edition in 2003. It is the only event worldwide uniquely devoted to the research topics covered by this special issue.

17. Proceedings of the European Conference on Genetic Programming. EuroGP was the first European event entirely devoted to GP. Run as a workshop in 1998 and 1999, it became a conference in 2000. It has now reached its sixth edition with EuroGP 2003 held at the University of Essex. Currently, this is the largest event worldwide solely devoted to GP.

ACKNOWLEDGMENTS

Riccardo Poli
Stefano Cagnoni

REFERENCES

[1] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[2] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, Mass, USA, 1992.

[3] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog, Stuttgart, Germany, 1973.

[4] H.-P. Schwefel, *Numerical Optimization of Computer Models*, Wiley, Chichester, UK, 1981.

[5] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, Mass, USA, 1996.

[6] W. B. Langdon and R. Poli, *Foundations of Genetic Programming*, Springer-Verlag, New York, NY, USA, 2002.

[7] D. Corne, M. Dorigo, and F. Glover, Eds., *New Ideas in Optimization*, McGraw-Hill, London, UK, 1999.

[8] D. Dasgupta, Ed., *Artificial Immune Systems and Their Applications*, Springer-Verlag, New York, NY, USA, 1999.

Riccardo Poli received a Ph.D. degree in bioengineering (1993) from the University of Florence, Italy, where he worked on image analysis, genetic algorithms, and neural networks until 1994. From 1994 to 2001, he was a lecturer and then a reader in the School of Computer Science of the University of Birmingham, UK. In 2001, he became a Professor at the Department of Computer Science of the University of Essex, where he founded the Natural and Evolutionary Computation Group. Professor Poli has published around 130 papers on evolutionary algorithms, genetic programming, neural networks, and image analysis and signal processing, including the book *Foundations of Genetic Programming* (Springer, 2002). He has been Cochair of EuroGP, the European Conference on GP, in 1998, 1999, 2000, and 2003. He was Chair of the GP theme at the Genetic and Evolutionary Computation Conference (GECCO) 2002 and Cochair of the Foundations of Genetic Algorithms Workshop (FOGA) 2002. He will be General Chair of GECCO 2004. Professor Poli is an Associate Editor of *Evolutionary Computation* (MIT Press) and *Genetic Programming and Evolvable Machines* (Kluwer), a reviewer for 12 journals, and has been a programme committee member of 40 international events.

Stefano Cagnoni has been an Assistant Professor in the Department of Computer Engineering of the University of Parma since 1997. He received the Ph.D. degree in bioengineering in 1993. In 1994, he was a Visiting Scientist at the Whitaker College Biomedical Imaging and Computation Laboratory at the Massachusetts Institute of Technology. His main research interests are in computer vision, evolutionary computation, and robotics. As a member of EvoNet, the European Network of Excellence in Evolutionary Computation, he has chaired the EvoIASP working group on evolutionary computation in image analysis and signal processing, and the corresponding workshop since its first edition in 1999. He is a reviewer for several journals and a programme committee member of several international events.