Reference Chromosome to Overcome User Fatigue in IEC

Yago SAEZ, Pedro ISASI, Javier SEGOVIA
and Julio C. HERNANDEZ
Universidad CARLOS III de Madrid y UPM, Campus de Leganés
Avda. de la Universidad 30, 28911 Madrid, Spain
yago.saez@uc3m.es

Received 26 May 2004
Revised manuscript received 5 October 2004

Abstract Evolutionary Computation encompasses computational models that follow a biological evolution metaphor. The success of these techniques is based on the maintenance of the genetic diversity, for which it is necessary to work with large populations. However, it is not always possible to deal with such large populations, for instance, when the adequacy values must be estimated by a human being (Interactive Evolutionary Computation, IEC). This work introduces a new algorithm which is able to perform very well with a very low number of individuals (micropopulations) which speeds up the convergence and it is solving problems with complex evaluation functions. The new algorithm is compared with the canonical genetic algorithm in order to validate its efficiency. Two experimental frameworks have been chosen: table and logotype designs. An objective evaluation measures has been proposed to avoid user interaction in the experiments. In both cases the results show the efficiency of the new algorithm in terms of quality of solutions and convergence speed, two key issues in decreasing user fatigue.

Keywords: Interactive Evolutionary Computation, Genetic Algorithm, Micropopulations, Chromosome Appearance Probability Matrix, Fatigue, Design, Table, Logotype.

§1 Introduction

The evolutionary computation paradigm is a group of stochastic learning techniques that use computational models that follow a biological evolution metaphor.[1] In these techniques a population of individuals (solutions) evolves until the convergence criteria is reached. The difficulty here is how to determine which parameters of the genetic algorithm (GA), like population size, selection type, crossover operator, mutation operator, encoding strategy, etc., are better for typical Interactive Evolutionary Computation (IEC) problems. Adjusting those parameters for performance can be an optimization problem itself.[2] The population size that guarantees an optimal solution in a short space of time has been a topic of intense research.[5,6] The success of these EC techniques is based on the maintenance of genetic diversity for which it is necessary to work with large populations. Large populations generally converge to better solutions, but they require more computational cost and memory. Goldberg et al.[3] developed the first population-sizing equation based on the variance of fitness. They further enhanced the equation that permits accurate statistical decision making among competing building blocks (BBs).[5] Extending the decision model presented in Reference[5], Harik et al.[6] tried to determine an adequate population size that guarantees a solution of the desired quality. To show the real importance of the population size in Evolutionary Algorithms (EAs) He and Yao[7] showed that the introduction of a non random population decreases convergence time. However, it is not always possible to deal with such large populations and there are two main areas where it becomes necessary to work with micropopulations. The first one is the area that refers to those problems where adequacy values must be estimated by a human being and the second one relates to problems that involve high computational cost.

The focus of this article involves the first areas. IEC techniques can be applied to different fields such as the design of objects with artistic or functional purposes.[8]

Applications which make use of this technique for other practical purposes have also been developed, such as the digital treatment of images,[9] the composition of musical works,[10] automotive design,[16] the automatic design of figures[18] and general artistic design.[20,21] such as the design of sculptures,[22] or the generation of gestures in 3D figures, aimed at virtual worlds, pictures or games.

A more detailed sort of example can be found in the most complete survey about IEC written to date.[23]

To overcome the user fatigue there are some works based on different techniques:

- Mixed Fitness: the system makes an evaluation before showing the population to the user. This technique allows working with more individuals, and the system acts like a previous filter removing those individuals not interesting and reducing the number of user evaluations Reference[8,19,27] or Reference[28].
- Predictive Fitness: appear as an alternative solution. It makes an study of the selected individuals and looks for all the common features. The aim is to increase the population (≥ 100) applying an automatic fitness to all before showing a small subset to the user. There are two main different learning methods for predicting fitness, the Euclidean distances of the search space,[14,15] and the rule systems trained by neural networks.[12] There are another proposals for predicting fitness based on fuzzy logic or hybrid

techniques.[24~26)]

- Best individual function: this technique tries to build the perfect individuals during each generation applying an specific function that takes the best possible features. The aim of this technique is to speed up the convergence by making generational jumps with the best individual selections.[17)]
- Initialize the first generation: other proposal to speed up the algorithm consists on initialize the first generation with the user main preferences, making faster the evolution.[19)]
- Searching masks: the user introduces his preferences and they are applied like a mask for the mutation and crossover operator. This technique was used for a face search engine.[4)]

To overcome the restrictions imposed by micropopulations and reduce user burden, we propose a new algorithm that is compared with a version of the canonical genetic algorithm which is adapted to deal with micropopulations. The new proposed approach is based on a matrix of independent probabilities applied at the chromosome level. The algorithm is called Chromosome Appearance Probability Matrix CAPM because it updates a probability matrix with the features of the selected individuals and uses it within the mutation operator in order to evolve towards the preferences of the user.

§2 Chromosome Appearance Probability Matrix

A key motivation for the development of the new model with learning mechanisms was the desire to not destroy, through recombination, favourable partial blocks that appear in the individuals. It was presented as an unexploited source of knowledge that was initially called "linkage information" and is used in several different approaches. The basis for introducing learning in GA's was established with the Population Based Incremental Learning (PBIL) algorithm proposed by Baluja and Caruana (1995).[11)] In PBIL, the recombination operator is replaced by a vector of independent probabilities of each binary variable. Sampling this vector implies the study of the selections made by the user. This is done in order to speed the evolution with regards to user needs. The success of this probability approach opened a wide spectrum lines of research,[13,29~31)] etc.

The steps of the proposed algorithm is explained in the following scheme:

1. Initialize the population
2. Evaluate all members of the population
   While not enough good solution reached {
3. Select individual(s) in the population to be parent(s)
4. Update the probability matrix with α. The cells to update are those relative to the information about the selection made
5. Create new individuals by applying to the parents:
   - a. the crossover operator
   - b. the oriented mutation operator
   - c. the clone remover operator

6. Replace all the population but parents with the new individuals
7. Evaluate the new individuals }

The steps 1, 2, 3, 5a and 7 are the same steps as usual in the canonical genetic algorithm scheme, and the proposed algorithm introduces the following new features:

The inclusion of a probability matrix that guides mutation: when the user selects an element of the population, his or her selection is based on the collective combination of features in each element of an individual. As a result we keep information about the whole chromosome. To do this, a multidimensional array, with the same number of dimensions as genes has been included. The bounds of the dimensions are the number of alleles of the different genes.
The probability array 'M' is initialized by

$M(gene_1, gene_2, gene_3, gene_m) = 1/T$

where 'm' is the number of genes, and gene$_i$ could have values in

$[allele^i_1 allele^i_2, allele^i_{n_i}]$,

and n_i the number of alleles of gene 'i'. The total possible combinations of chromosomes 'T' is calculated by multiplying the maximum sizes of each gene $(T = \prod_{i=1}^{m} n_i)$.

This array shows the probability that each possible combination of alleles has of being chosen. Each iteration implies a selection of one or two individuals, and its chromosomes represent a position in the above array. After the selection, the corresponding position in the array is updated by a factor of α with the increment factor of the update rule, Δ_M. This Δ_M is calculated by the following equation:

$$\Delta_M = [M_{gen^1_s, \cdots, gen^n_s} \times (1.0 + \alpha))] - M_{gen^1_s, \cdots, gen^n_s} \tag{1}$$

The example in Fig. 1 shows how the update rule works for 1 chromosome with 2 different genes, gen_1 with 4 alleles {pos1,..,pos4}, and gen_2 with 10, {0..9}. It can be clearly seen how the probability matrix 'M' is updated with $\alpha = 0.005$ and how it affects to the rest cells.

The update operations make sure that the sum of all the elements of the array will be 1. This array is very useful for keeping information about the selection frequency of a determined chromosome and therefore helps the mutation process to evolve towards the preferences of the user.

The inclusion of an oriented mutation operator: the mutation operator is responsible for the mutation of the individuals. Once a gene has been selected for mutation a specific chromosome is taken as the base of the mutation process (reference chromosome). This chromosome is selected from the whole range of possible chromosomes following a uniform distribution fixed by the probability array. The higher the value of the probable array of the chromosome, the better the probability of it being chosen. In the mutation process, a position in

Gen_1 = [1..4]

Gen_2 = [0..9]

	Pos1	Pos2	Pos3	Pos4
0	0.045	0.025	0.025	0.02
1	0.015	0.025	0.025	0.015
2	0.025	0.01	0.025	0.025
3	0.025	0.04	0.025	0.04
4	0.025	0.025	0.025	0.025
5	0.025	0.025	0.025	0.025
6	0.015	0.025	0.02	0.025
7	0.025	0.025	0.02	0.025
8	0.025	0.025	0.035	0.025
9	0.025	0.025	0.025	0.025

```
1. Δm = [M[2,3]*(1.0 + α)] - M[2,3]
   Δm = [0.04*(1+0.005)]- M[2,3]
   Δm = 0.0402 - 0.04 = 0.0002
2. M[2,3] = M[2,3] + Δm = 0.0402
3. M[1,0] = M[1,0] - (Δm / 40) = 0.044995
   M[1,1] = M[1,1] - (Δm / 40) = 0.014995
       (...)
   M[4,9] = M[2,9] - (Δm / 40) = 0.024995
```

Fig. 1 Update Rule Example for CAPM Algorithm, $\alpha = 0.005$

the chromosome to be mutated is randomly selected. Subsequently, the gene in this position is substituted by a gene from the reference chromosome in that same position. Thus, the mutation operator is the result of a function of the chromosome to be mutated and the reference chromosome:

$$newChrom' = mutOneGen(OldChrom, referenceChrom) \quad (2)$$

This approach has the ability to broadcast the preferences of the user towards all the chromosomes of the individuals.

The inclusion of a clone remover operator: the clone remover operator is responsible for mutating all those individuals that have exactly the same genetic structure as other individuals in the same population.

Replace all the population but parents with the new individuals: the proposed strategy is elitist however, due to the fact that the user is not interested in evaluating the same individuals between iterations and the algorithm mutates the parents for the next generation, making them slightly different.

§3 Experimental Domains

We have made two applications within two different frameworks, using both kinds of algorithms in order to find a good generic solution in IEC.

3.1 Table Design

The environment problem of interactive evolutionary computation that needs to be solved is the creation of a 3D design of general-purpose "tables"

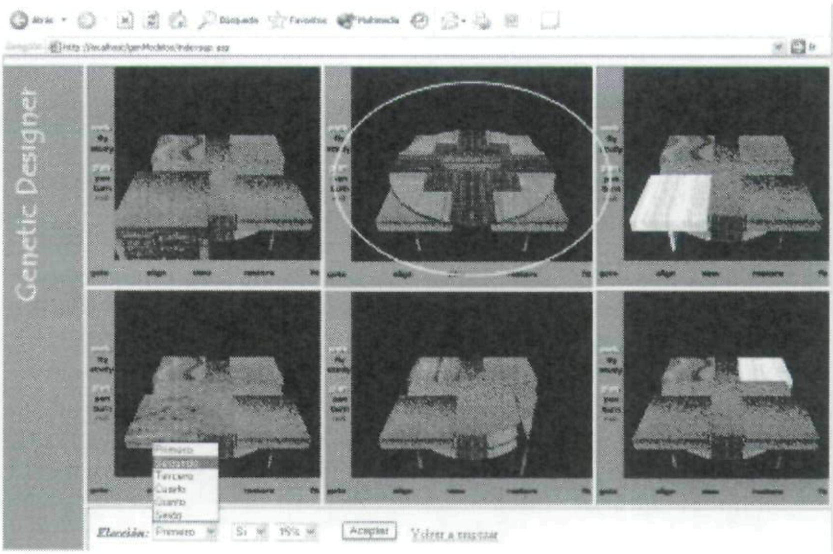

Fig. 2 Table Design Framework

(Fig. 2). In this creative process, the participation of a human being is needed in order to determine which tables are more creative, more functional, or which ones better fit their needs. In addition, different types of objects must be allowed to combine and modify within the same design, i.e., the tables are complex objects generated from the genetic mutation of their various pieces.

[1] Encoding the problem

In this scheme, each table is represented by an individual of the population to be evolved, and each table is made up of five pieces. Each kind of piece that makes up the table corresponds to different chromosomes. Each piece has its own genes, which correspond to the kind of piece, material, measurement, angle, position, etc.

Looking at the possible alleles for each gene, the total number of different possibilities for only one chromosome are $1.350.000 \equiv 30 * 15 * 15 * 20 * 10$. Since we have five chromosomes for each individual, then the solution space is

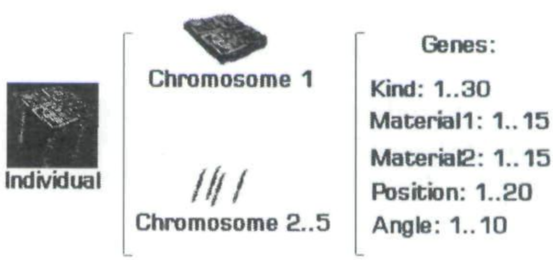

Fig. 3 Encoding the Tables

practically infinite, take into consideration that we are talking about $4{,}48 * 10^{30}$ different combinations ($1.350.000^5$).

[2] Making the evaluation

In order to facilitate the evaluation process the user preferences will be used to evolve the populations. The user receives six individuals as input, from the total population, and he/she will be able to choose and classify at least the three that best fit his or her personal preferences. As an initial option the user can also suggest to the system the rough measurements of the table he/she wants as well as the number of legs.

The use of mixed fitness evaluation allows for the widening of genetic diversity since the population can be increased without the need to complicate the operation on the part of the user. This one will evaluate a subset of the population which will have been previously filtered by the automatic evaluation function (implicit fitness).

3.2 Trademark Finder

The problem arises from an idea proposed in Reference.[28] The main objective of the trademark finder is to help the user in the difficult task of finding a specific logo that is new, different, or eye-catching for a product or a company. For this purpose, like in brainstorming sessions, the system offers different types of words, applies different colors, different backgrounds and different styles of fonts. It is not as realistic as the table designer because at this point the main objective is to show clearly the algorithm performance without graphic distractions. The user can select the algorithm and the default parameters to be used. From that point, the algorithm starts randomly with 6 words (population) each one made up of 5 letters with random colors and styles. The user must select from each iteration the two most interesting for him.

[1] Encoding the problem

An individual consists of five letters with four genes, representing color, font type, size and background respectively. Each word is an individual of the population and each letter is a chromosome with four genes.

Looking at the possible alleles for each gene, the total number of different possibilities for only one chromosome are $20.000 \equiv 10 * 20 * 10 * 10$. Since we have five chromosomes for each individual, then the solution space is practically infinite, take into consideration that we are talking about $3{,}2 * 10^{21}$ different combinations (20.000^5).

[2] Making the evaluation

As we learned with the design of tables experiment it is more advantageous not to give a numerical value for the evaluation of individuals and to choose the most preferable designs from the point of view of the user (subjective measure).

As an initial option (the base case), the user can also suggest to the system the word to search and the creativity factor used for the generation of

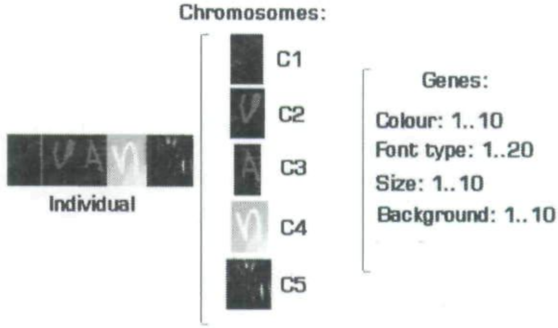

Fig. 4 Encoding the Logos

logotypes. As with the table designer, this application was also developed as an example and the user will receive a subset of six individuals as input from a total population of ten.

As in the previous case, this subset will be the result of applying the implicit evaluation function to the whole population. Subsequently, the user is prompted to insert the two most interesting logotypes according to his or her personal preferences(explicit fitness value).

The implicit evaluation function is very simple. To avoid using the same text color over the same background color, the implicit evaluation function compares them and penalizes individuals with equal features.

§4 Results

4.1 Experimental Framework

Numerous experiments (more than 1000) were required to validate the efficiency of the method and it was necessary to develop a simulator that replaced the human interaction.

Because we wanted to test the algorithm in the most general framework the implicit fitness was removed due to that fact that it depends on the problem and the requisites. However, the removal of the implicit fitness leads to slower convergence towards a valid solution.

We made the experiments with two different automatic evaluation functions:

[1] Evaluation based on ranking of preferences

For some type of products and business the preferences of corporate colors and fonts can be clear at the beginning of the search, perhaps all letters in red are preferred but the type of red selected depends on the user for example. For this reason, as a first test, we designed an automatic evaluation function based on rankings of preferences. This type of evaluation function makes the problem a little bit more complex, since the algorithm can initially evolve towards near to optimal solutions. With this simple automatic evaluation function we obtained

Table 1 Results of the Experiments for Both Algorithms

Evaluation mode	Ranking		Rules	
Algorithms	Classic	CAPM	Classic	CAPM
Exp. opt. solution (fit. = 100)	64.60%	86.30%	0%	43%
Aver. iteration opt. solutions	∞(344.875)	46.765	∞	159.069
Exp. valid solution (fit. >= 90)	99.30%	89.49%	0%	50%
Aver. iteration valid solutions	150.192	38.949	∞	153.084

very good results for the proposed algorithm as can be seen on Table 1.

However, the problem was that the automatic evaluation function was a non real world based function. So we decided to include realistic conditions that implied a non linear function and a complex evaluation.

[2] Evaluation based on fluctuating decisions rule sets

Like humans do, it is possible that the user could find more than one type of solution interesting. Furthermore, after the study of the experiments conducted with humans we realized that often the user does not have a clear idea of his own preferences and additionally his preferences could change as new designs are proposed. To simulate the doubts inherent in human behaviors, we included two sets of predefined evaluation rules that have very different preferences. Those rules randomly change with a probability of 70% using set 1, and 30% using set 2 each iteration. These rules are applied independently to each chromosome (character) in order to obtain an objective measure.

In order to make that search even harder, we included two conditions that forced the solution to search for words with the same background for first and last letter, and thus forcing a difference between them and the rest.

Evaluation set rules:

- Set 1
  - User prefers blue color (5 evaluation points) over navy (3 evaluation points), and navy over red (1 evaluation point), the other colors are not taken into account. The same occurs for the font and size of characters.
  - If the background of the first and last characters are the same, the evaluation is increased in some quantity (2,5). And if the background of the rest are different one each other, then the evaluation is also increased in some quantity (2,5).
- Set 2
  - Similar to set 1, but the preferences of color, size and font change, and they are different from set of rules 1. Now the yellow is preferred over green, the green over red, and so on.

The maximum evaluation for the best design is one hundred:
$f(x) = 5(chars) \cdot 5(pts) \cdot 3(features) + 5(chars) \cdot 2.5(pts) \cdot 2(bg) = 100$

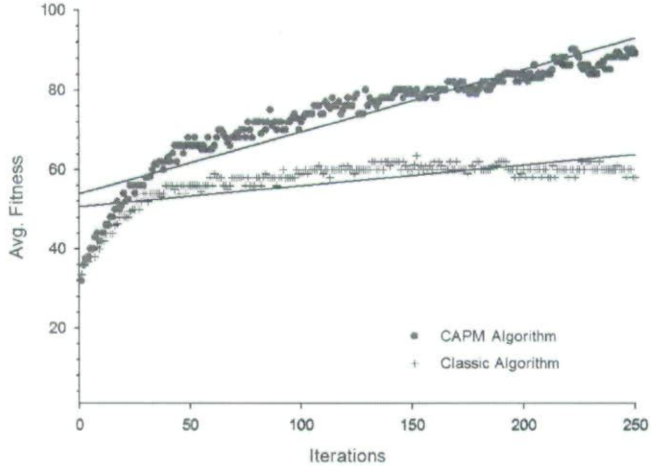

Fig. 5 Average Fitness Per Iteration

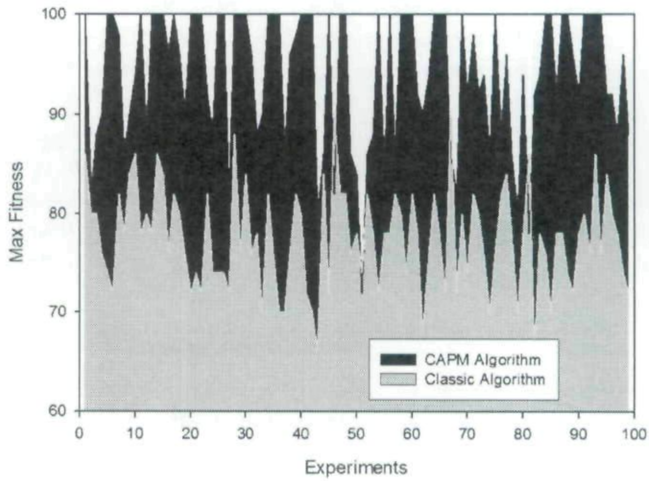

Fig. 6 Maximum Fitness Per Experiment

[3] Comparative study

This section presents a comparative study of some simulations in order to show the behaviour of the algorithms explained previously. In the experiments the final condition is reached when the fitness of one element is 100 points (optimal solution), or when the iterations surpass 250. A valid measure is the condition in which the fitness is greater than 90 over 100 and we refer to it as a near optimal solution or valid solution.

At this point, after making the first 100 different experiments ($\alpha = 0.005, Pm = 5\%$) for both algorithms and both types of automatic evaluation function, the following results were achieved:

As can been seen clearly in Figs 5, 6. When comparing the results, there is a significant difference between the algorithms: much faster solutions in terms of iterations and better quality solutions.

Looking at the results for the complex evaluation mode the problem is that the user must take at least 159 iterations on average to obtain an optimal solution. This is too much with regard to the user fatigue. However, the results are no so bad if one takes into account that the evaluation uses a non-linear worst case function. This has various consequences; it changes evaluation rules randomly (with 30% probability of change in each iteration). It also applies opposite evaluation preferences for each set of rules before finally finding the best solution. The evolution process is very complex and what is really going to happen with the user is unpredictable. We don't think that the user is going to change completely his preferences with each iteration (with 30% of probabilities) but we did decide to experiment with the worst case. In fact, in this evaluation mode the classic genetic algorithm is unable to find even a valid solution.

So if the proposed algorithm with a search space of of $3{,}2 * 10^{21}$ and a random evaluation function needs 159 iterations in average to find an optimal solution, and the simple genetic algorithm never converges, we can affirm that CAPM reduce effectively the user's fatigue.

§5 Conclusion

In this first approach to finding an efficient algorithm for solving generic problems in the field of IEC we have made the following conclusions:

1. The numeric analysis of the classic algorithm, shows that although it has been considered sufficient so far it is not good enough for IEC.
2. As an alternative we propose an algorithm that learns through the study of the selections made by the user and that improves the results given by the classic algorithm, both in terms of the quality of the solutions and in terms of the number of iterations involved in finding valid/optimal solutions.
3. The success of the algorithm depends on the user and the problem, but we suggest the proposed algorithm as a good generic solution for any IEC framework. Furthermore, to imitate human characteristics the evaluation function (simulator) is non predictable, it changes opinions or preferences randomly with a probability of 30%. Also, it has been decided to make the search based on a non linear fitness function.

Finally, the algorithm is going to be compared against others (Population Based Incremental Learning, PBIL and Fitness Predictive Genetic Algorithm, FPGA), in order to prove the effectiveness of the proposed algorithm.

Acknowledgements

Thanks for the reviewers work, it has been useful to improve our manuscript. This article has been financed by the Spanish founded research MCyT project TRACER, Ref: TIC2002-04498-C05-04.

References

1) Goldberg, D.E., "Genetic Algorithms in Search, Optimization, and Machine Learning," *Ed. Addison-Wesley, Reading, MA.*, 1989.

2) Grefenstette, J.J., "Optimization of Control Parameters for Generic Algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, *16*, *1*, pp. 122-128, 1986.

3) Goldberg, D.E. and Rundnick, M., "Genetic Algorithms and the Variance of Fitness," *Complex Systems*, *5*, *3*, pp. 265-278, 1991.

4) Caldwell, C. and Johnston, V.S., "Tracking a Criminal Suspect through Face Space with a Genetic Algorithm," *ICGA-4*, pp. 416-421, 1991.

5) Goldberg, D.E., Deb, K. and Clark, J.H., "Genetic Algorithms, Noise, and the Sizing of Populations," *Complex Syst.*, *6*, *4*, pp. 333-362, 1992.

6) Harik, G., Cantu-Paz, E., Goldberg, D.E. and Miller, B.L., "The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations," *Transactions on Evolutionary Computation*, *7*, pp. 231-253, 1999.

7) He, J. and Yao, X., "From an Individual to a Population: An Analysis of the First Hitting Time of Population-based Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, *6*, pp. 495-511, Oct., 2002.

8) Bentley, P., "From Coffee Tables to Hospitals: Generic Evolutionary Design," *Evolutionary design by computers*, Morgan-Kauffman, pp. 405-423, 1999.

9) Sims, K., "Artificial Evolution for Computer Graphics," *Comp. Graphics*, *25*, *4*, pp. 319-328, 1991.

10) Moore, J.H., "GAMusic: Genetic Algorithm to Evolve Musical Melodies," *Windows 3.1 Software available in: http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/genetic/ga/systems/gamusic/0.html.*, 1994.

11) Baluja, S., Pomerleau, D. and Jochem, T., "Towards Automated Artificial Evolution for Computer-generated Images," *Connection Science*, pp. 325-354, 1994.

12) Biles, P., Anderson G. and Loggi, L.W., "Neural Network Fitness Functions for a Musical IGA," in *Proceedings of IIA96/SOCO96. International ICSC Symposia on Intelligent Industrial Automation and Soft Computing.*, 1996.

13) Gonzalez, C., Lozano, J. and Larranarraga, P., "Analyzing the PBIL Algorithm by Means of Discrete Dynamical Systems," *Complex Systems*, 1997.

14) Hsu, F.-C. and Chen, J.-S., "A Study on Multi Criteria Decision Making Model: Interactive Genetic Algorithms Approach," in *Proceedings of IEEE Int. Conf. on System, Man, and Cybernetics (SMC99)*, pp. 634-639, 1999.

15) Lee, J.-Y. and Cho, S.-B. "Sparse Fitness Evaluation for Reducing User Burden in Interactive Genetic Algorithm," in *Proceedings of FUZZ-IEEE 99*, II, pp. 998-1003, 1999.

16) Graf, J. and Banzhaf, W., "Interactive Evolutionary Algorithms in Design," *Procs of Artificial Neural Nets and Genetic Algorithms*, Ales, France, pp. 227-230, 1995.

17) Ingu, T. and Takagi, H., "Accelerating a GA Convergence by Fitting a Single-peak Function," *IEEE International Conference on Fuzzy Systems FUZZ-IEEE'99*, pp. 1415-1420, 1999.

18) Vico, F.J., Veredas, F.J., Bravo, J.M. and Almaraz, J., "Automatic Design Sinthesis with Artificial Intelligence Techniques," *Artificial Intelligence in Engineering 13*, pp. 251-256, 1999.

19) Saez, Y., Sanjuan, O. and Segovia, J., "Algoritmos Geneticos para la Generacion de Modelos con Micropoblaciones," in *Proceedings Algoritmos Evolutivos y Bioinspirados (AEB 02)*, 2002.

20) Santos, A., Dorado, J., Romero, J., Arcay, B. and Rodliguez, J., "Artistic Evolutionary Computer Systems," *Proc. of the GECCO Workshop*, Las Vegas, 2000.

21) Unemi, T., "SBART 2.4: An IEC Tool for Creating 2D Images, Movies and Collage," *Proc. of the Genetic and Evolutionary Computation, Conference Program*, Las Vegas, 2000.

22) Rowland, D., "Evolutionary Co-operative Design Methodology: The Genetic Sculpture Park," *Proc. of the GECCO Workshop*, Las Vegas, 2000.

23) Takagi, H., "Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation," *Proc. of the IEEE*, *89*, *9*, pp. 1275-1296, 2001.

24) Sugimoto, F. and Yoneyama M., "Robustness Against Instability of Sensory Judgment in a Human Interface to Draw a Facial Image Using a Psychometrical Space Model," *Proceedings of IEEE International Conference on Multimedia and Expo*, pp. 653-638, 2000.

25) Dozier, G., "Evolving Robot Behavior via Interactive Evolutionary Computation: from Real-world to Simulation," *Proceedings of the 2001 ACM symposium on Applied computing*, pp. 340-344, 2001.

26) Ohsaki, M. and Takagi, H., "Application of Interactive Evolutionary Computation to Optimal Tuning of Digital Hearing Aids," *International Conference on Soft Computing IIZUKA 98*, pp. 849-852, 1998.

27) Machado, P. and Cardoso A., "All the Truth about NEvAr," *Applied Intelligence, Special issue on Creative Systems*, *16*, *2*, pp. 101-119, 2002.

28) Saez, Y., Sanjuan, O., Segovia, J. and Isasi, P., "Genetic Algorithms for the Generation of Models with Micropopulations," *Proc. of the EUROGP'03*, Univ. of Essex, UK. Apr., 2003.

29) Larrañaga, P. and Lozano, J.A., "Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation," *Kluwer*, *Boston*, *MA*, 2001.

30) Larrañaga, P., Lozano, J.A. and Bengoetxea, E., *Estimation of Distribution Algorithms Based on Multivariate Normal and Gaussian Networks*, *KZZA-IK-1-01*, 2001.

31) Kern, S., Muller, S.D., Hansen, N., Buche, D., Ocenasek, J. and Koumoutsakos, P., *Learning Probability Distributions in Continuous Evolutionary Algorithms, A Comparative Review*, *Kluwer Academic Publishers*, 2004.

Yago Saez: He received the Computer Engineering degree from the Universidad Pontificia de Salamanca in 1999 Spain. He now is a Ph.D. student and works as assistant professor at the EVAN-NAI Group at the Computer Science Department of CARLOS III, Madrid, Spain. His main research areas encompasses the interactive evolutionary computation, the design applications and the optimization problems.

Pedro Isasi, Ph.D.: He received Computer Science degree and Ph.D. degree from the Universidad Politécnica de Madrid(UPM), Spain in 1994. He is now working as professor at the EVAN-NAI Group at the Computer Science Department of CARLOS III, Madrid, Spain. His main research areas are Machine Learning, Evolutionary, Computation and Neural Networks and Applications to Optimization Problems.

Javier Segovia, Ph.D.: He is a receiving physicist, Ph.D. degree in Computer Science (with honours) from the Universidad Politécnica de Madrid (UPM). Currently Dean of the UPM School of Computer Science, and is editor and/or author of more than 70 scientific publications in the fields of genetic algorithms, data and web mining, artificial intelligence and intelligent interfaces.

Julio C. Hernandez, Ph.D.: He has received degree in Maths, Ph.D. degree in Computer Science. His main research area is the artificial intelligence applied to criptography and net security. His unofficial hobbies are chess and go. Currently, he is working as invited researcher at INRIA, France.