World Academy of Science, Engineering and Technology
International Journal of Educational and Pedagogical Sciences
Vol:10, No:10, 2016

Understanding Evolutionary Algorithms through Interactive Graphical Applications

Javier Barrachina, Piedad Garrido, Manuel Fogue, Julio A. Sanguesa, Francisco J. Martinez

***Abstract*—**It is very common to observe, especially in Computer Science studies that students have difficulties to correctly understand how some mechanisms based on Artificial Intelligence work. In addition, the scope and limitations of most of these mechanisms are usually presented by professors only in a theoretical way, which does not help students to understand them adequately. In this work, we focus on the problems found when teaching Evolutionary Algorithms (EAs), which imitate the principles of natural evolution, as a method to solve parameter optimization problems. Although this kind of algorithms can be very powerful to solve relatively complex problems, students often have difficulties to understand how they work, and how to apply them to solve problems in real cases. In this paper, we present two interactive graphical applications which have been specially designed with the aim of making Evolutionary Algorithms easy to be understood by students. Specifically, we present: (i) TSPS, an application able to solve the "Traveling Salesman Problem", and (ii) FotEvol, an application able to reconstruct a given image by using Evolution Strategies. The main objective is that students learn how these techniques can be implemented, and the great possibilities they offer.

***Keywords*—**Education, evolutionary algorithms, evolution strategies, interactive learning applications.

I. Introduction

EVOLUTIONARY ALGORITHMS (EAs) are based on Darwinian theories of evolution to explain the origin of species [1]. They have been successfully used to solve several types of optimization problems [2]. In particular, these algorithms manage a set of entities that represent potential solutions, which are mixed, evolve, and compete among them, trying to obtain the best solution that prevails over time. EAs are able to provide a near optimal solution to a problem without checking all the possible solutions, while other classical methods are not able to find solutions in a reasonable time. Other techniques such as the Active-Set Approaches [3], [4] lead to solving very large-scale optimization problems very efficiently, although EAs are still more flexible to address a wide range of the complex problems.

Evolution Strategies are a kind of EAs with the particularity that the mutation steps are included in the chromosome. This kind of EAs obtains very good results in numerical optimization problems, especially when working with continuous variables.

In this work, we focus on the problems found when teaching EAs in the Bachelor Degree in Computer Science in the University of Zaragoza, Spain. Although these algorithms are suitable for many aspects related to Computer Science studies, and can be very powerful to solve relatively complex problems, they are usually presented in a series of lectures in which theories and concepts are simply communicated to the students. Therefore, it is very common to observe that students have difficulties to correctly understand the operation of EAs and Evolution Strategies for solving problems, and how to apply them to solve problems based in real cases. Additionally, the scope and limitations of most of these mechanisms are usually presented by professors only in a theoretical way, which does not help too much students to understand them adequately. To address this issue, many authors have proposed to apply useful techniques and applications (e.g., interactive games, tutorials, AI-based applications, etc.) [5]-[8]. These approaches allow professors to present theoretical concepts to the students in a more interactive and attractive way.

In this paper, we present two interactive graphical applications which were designed with the aim of making EAs easy to be understood by students. Specifically, we present: (i) TSPS, an application able to solve the "Traveling Salesman Problem", and (ii) FotEvol, an application able to reconstruct a given image by using Evolution Strategies. The main objective is that students can learn how these techniques can be implemented, and the great possibilities they offer.

The first application is based on a well-known optimization problem, i.e., the Traveling Salesman Problem (TSP), which was firstly formulated in 1930 [9]. In particular, the TSP is computationally complex; although any given solution to the problem can be verified quickly (in polynomial time), the time required to solve the problem by using any currently known algorithm highly increases as the size of the problem grows. This means that the time required to solve even moderately sized versions of many cases can easily require a high amount of computing power, even becoming unaffordable.

The second problem presented in this work is FotEvol, a graphical application able to reconstruct images by using an Evolutionary Strategy. This reconstruction consists on taking a photograph through a webcam, and, starting from a random generated image, adjusting it to the initially taken picture.

This paper is organized as follows: Section II introduces the EAs. In Section III, we present our two applications that allow students to better understand the effectiveness and execution time limitations of EAs. Section IV presents the development environment. Section V presents several existing graphical interactive applications specially designed to help students in their learning process, and finally, Section VI concludes this

Javier Barrachina, Piedad Garrido, Manuel Fogue, Julio A. Sanguesa and Francisco J. Martinez are with the University of Zaragoza, Spain (e-mail: barrachina@unizar.es, piedad@unizar.es, mfogue@unizar.es, jsanguesa@unizar.es, f.martinez@unizar.es).

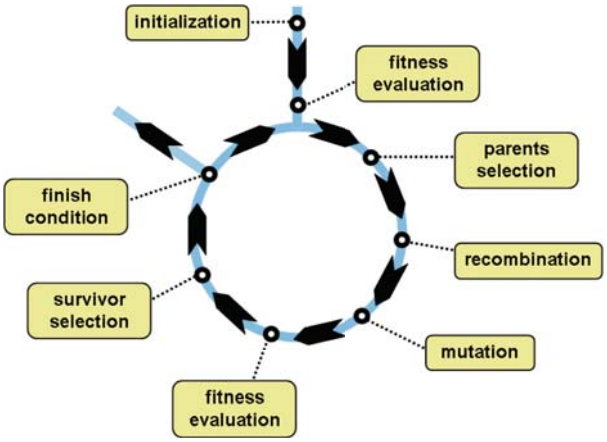


Fig. 1 Methodology of Evolutionary Algorithms

paper and presents some future work.

II. Background on Evolutionary Algorithms

EAs imitate the principles of natural evolution as a method to solve parameter optimization problems. The selection is inevitable in an environment that can only accommodate a limited number of individuals. Natural selection favors those individuals competing for resources in a more effective way. This selection based on competition and the phenotypic variations (i.e., physical or behavioral features which affect the interaction of the individual with the environment) are the essence of the evolution process. A favorable modification of these features produces a propagation of them to the offspring, since it increments the reproduction probability.

A microscopic view of natural evolution is obtained by molecular genetics. Each individual is considered as a dual entity formed by: (i) external phenotypic properties, and (ii) its internal chromosome or genotype (gene set). The genotype of an individual encodes its phenotype, and genes are the inheritance functional units.

Applied in Computer Science, EAs manage a set of entities that represent potential solutions, which are mixed, mutated, and compete among them, trying to obtain the best solution that prevails over time. EAs are able to provide a near optimal solution to a problem without checking all the possible solutions, thereby reducing the amount of time required to find a suitable solution. These algorithms are used to solve multi-dimensional, optimization, modeling, and simulation problems [10]. In particular, they consist on applying natural selection to an individual population in order to obtain individuals (i.e., solutions) better adapted to the environment. After a determined number of generations, these algorithms are able to obtain the best solution for a particular problem (i.e., the best adapted individual). Specifically, they consist on an individual population which generates descendants, and the best individuals are selected to obtain the next generation. In each generation, some of the best candidates are chosen to generate a new offspring by applying recombination and mutation operators, giving rise to new individuals who will return to compete in the environment.

All EAs follow the same methodology, presented in Fig. 1. As shown, this scheme belongs to the category search algorithms for generation-and-test. The parents selection is done by giving priority to the best adapted individuals using a particular algorithm, which can be proportional to their fitness value, ordering the individuals according to their adaptation degree, etc.

Regarding the recombination or crossover operator, it is usually the most important step in this type of algorithms. It is based on a combination of two (or more) genotypes called parents to generate new genotypes, the offspring. Parents are selected from existing individuals based on their level of adaptation or fitness, in order to obtain descendants which inherit good genes from their parents. By applying iteratively this operator (i.e., recombination), the probability of appearing the best chromosome genes in the population increases, leading to a convergence towards the best solutions. Fig. 2 shows an example of recombination.

As for the mutation operator, it is responsible for random changes in the characteristics of chromosomes, and it is typically performed at gene level. Usually, the mutation rate (probability of changing a gene) is very small, and depends on the length of the genotype; hence, the new produced genotypes after mutation will not be very different from the originals. The role of the mutation in EAs is also very important, since recombination typically implies that the population of individuals converges making individual chromosomes very similar. However, there may be a state space region which has not been properly scanned, and therefore, the mutation is used to reintroduce genetic diversity in the population and avoid local optimal.

The evaluation function (degree of adaptation or fitness) represents a heuristic estimation of the quality of the solution. The survivors for the next generation are selected by using this function.

III. Our Proposal

As previously commented, in this work we focus on the Artificial Intelligence field. More specifically, on the problems found when teaching the EAs, which imitate the principles of natural evolution, and have been successfully used to solve several types of optimization problems.

A. TSPS: Traveling Salesman Problem Solver

One of the objectives of our work is to present the main features of EAs in a graphical and interactive way, making it possible that students can learn them adequately. Therefore, we implemented an application, i.e., the Traveling Salesman Problem Solver (TSPS) that solves the classical Traveling Salesman Problem (TSP) by using a genetic algorithm. The well-known TSP consists on determining a tour while minimizing the total distance or the cost involved in visiting several cities [9].

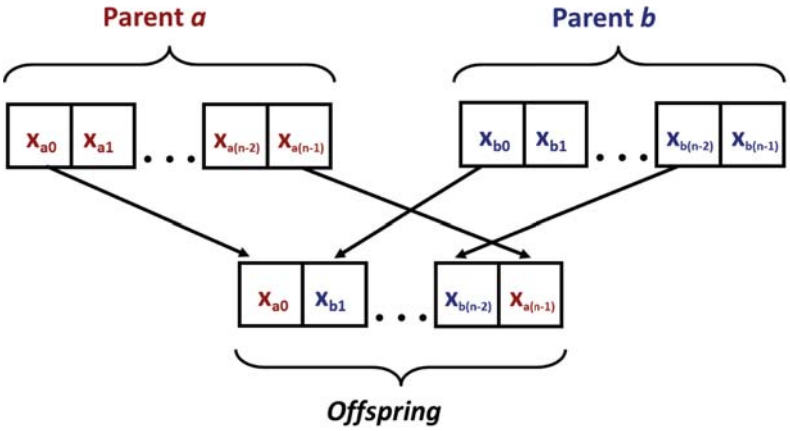


Fig. 2 Example of recombination

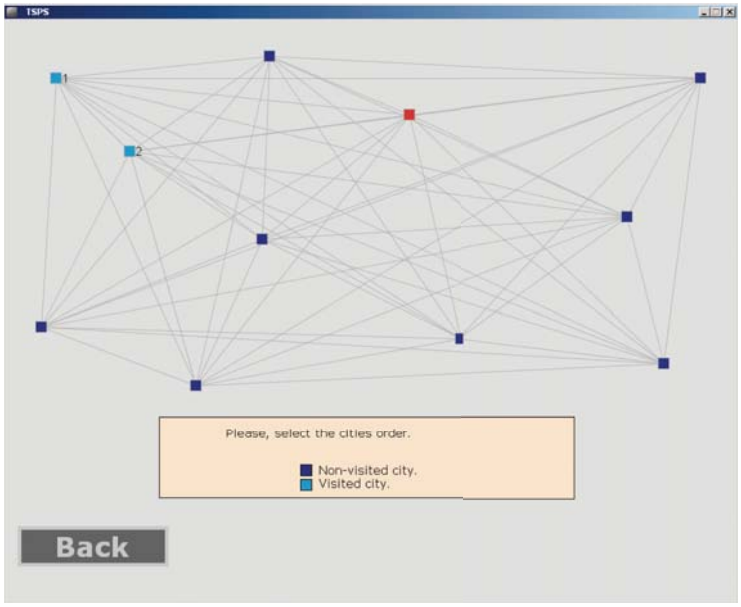


Fig. 3 Screenshot where students manually introduce the route that they consider the shortest

When TSPS is executed, the TSP is firstly explained. Then, students may introduce the route that they consider to be the shortest one (as shown in Fig. 3). Note that the distance between cities is equivalent to the real distance between pixels of the image. Once students validate their route, the system calculates a sub-optimal route by using a genetic algorithm. Table I presents the features of the algorithm.

Finally, (see Fig. 4), the system shows the results obtained, indicating: (i) the distance of the user's route, (ii) the distance of the route proposed by our system, and (iii) the number of generations required to improve the user's proposal. In addition, the system allows seeing both routes using an animation (as shown in Fig. 5).

Using our TSPS application, students can verify visually the effectiveness of EAs to find better solutions. In particular, EAs allow to solve a problem that may seem as simple as the TSP in a near optimal way, compared to the solution proposed intuitively by the student. In the presented example, the system only accounts for 11 cities. After presenting this experiment to the students, we ask them what would happen if the number of cities increased. Analyzing the results obtained, students are able to realize the effectiveness of EAs, and using our TSPS,

TABLE I
GENETIC ALGORITHM FEATURES

Parameter	Value
number of cities	11
number of population individuals	10
number of generations	500
parents selection	tournament selection
tournament size	5
survivor selection	steady state model
mutation rate	0.07
fitness function	minimize distance

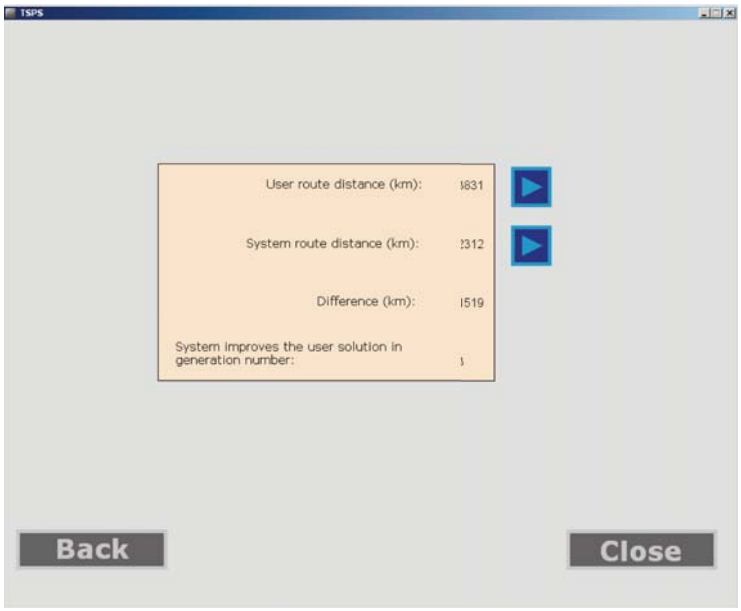


Fig. 4 Screenshot that shows the obtained results after applying the genetic algorithm

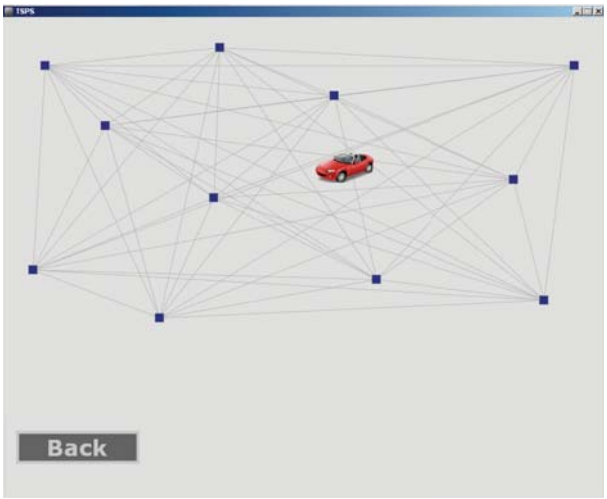


Fig. 5 Example of a route animation

they see a possible real application of their use.

B. FotEvol: An Image Reconstruction Application

The second objective of this work consists on studying and analyzing the required execution time to obtain a good solution when using EAs. Specifically, we chose an example in which the execution time is quite large. In particular, we implemented a graphical application able to reconstruct images by using an Evolutionary Strategy. This reconstruction consists on taking a photograph through a webcam, and, starting from an image generated with random pixels, adjusting it to the picture initially taken. Fig. 6 shows an example which presents a simple image reconstruction; Fig. 6 (a) shows the original image, Fig. 6 (b) shows the initial random pixels image, and Fig. 6 (c) shows the reconstructed image.

Once the image is captured, in the first step of this application, students can change the size of the picture, since the size of the image is a key factor to obtain a valid solution in a reasonable time. As shown in Fig. 7, the capture is made as a color image.

Equation (1) presents the fitness function required to be used to adjust the image, in case we used RGB color model images. This function sums the distances between the values of red, green, and blue of both images, the real one and the evolutionary-based.

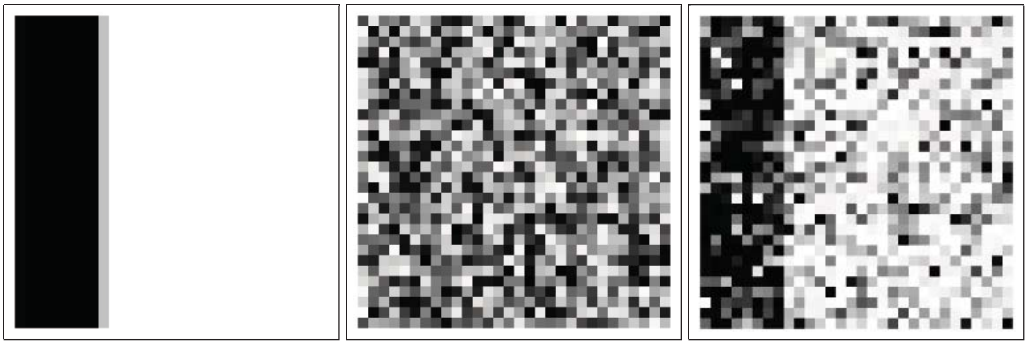
$$Fitness\ Function = \sum_{i=0}^{N}(|R_{real_i} - R_{evol_i}| + \\ +|G_{real_i} - G_{evol_i}| + |B_{real_i} - B_{evol_i}|) \quad (1)$$

In order to reduce the complexity of the problem, after capturing the photograph, our application directly converts it from the RGB color model to grayscale. Making this, FotEvol has only to adjust one value for each pixel. If the RGB color model were used, it should be necessary to adjust three different values (red, green, and blue), thereby requiring much more time to perform the reconstruction process. This transformation simplifies the fitness function, as shown in (2).

$$Fitness\ Function = \sum_{i=0}^{N}(|G_{real_i} - G_{evol_i}|) \quad (2)$$

Fig. 8 shows an example of the final image captured, using a smaller size and a grayscale converted format.

Once the image has been captured, there are three algorithm parameters that students have to configure: (i) The number of generations, (ii) the number of parents, and (iii) the number of offspring individuals. We consider interesting to allow



(a) (b) (c)

Fig. 6 Example of a simple image reconstruction

Fig. 7 Screenshot of the first step in FotEvol application

Fig. 8 Screenshot of a captured image using FotEvol

students to test different configurations in order to clearly observe how Evolutionary Strategies change their behavior. Then, the system generates a first generation image which consists on randomly deployed grayscale pixels. This image has the same size than the original image. Using this first generation image, the evolutionary strategy algorithm starts, using the configuration parameters presented in Table II. As shown, the system uses 1 as step size. We considered this for reducing the variability of the offspring. Regarding the survivor selection, we applied a $\mu+\lambda$ selection. This selection method avoids stopping the generation of children, returning back to parents when descendants in one generation are no good enough, since parents are included in the selection. This specific issue is discussed with students in the laboratory.

TABLE II
EVOLUTIONARY STRATEGY FEATURES

Parameter	Value
number of generations	defined by user
number of parents	defined by user
number of offspring individuals	defined by user
parents selection	randomly
survivor selection	$\mu + \lambda$
step size	1
fitness function	defined in (2)

Fig. 9 shows an example of the execution process. As shown, the system starts with a randomly generated image. In intermediate generations we can see that image begins to look like the original one. In particular, we can distinguish the form of the person who is at the bottom, the body and the hair of the person who is in front, and the bag which is on the table on the left. Finally, we can see the original image. The main problem presented in this kind of solutions is the time required to obtain a valid solution. Since genotypes contain all image pixels, Evolution Strategy-based operations imply a large computational cost. Although we have reduced the complexity of the problem by using grayscale images instead of RGB ones, it could take a long time to obtain a sharp image. This problem increases when using bigger images. For this reason, our application clearly shows to the students the disadvantages of Evolutionary Strategies in terms of execution time. To better study this effect, students are required to change the configuration parameters by using fewer generations, as well as reducing the number of parents, and offspring individuals. Hence, they can realize that, although the obtained image is not as well adjusted to the original, the

Fig. 9 Example of the evolution process of the picture

required execution time is drastically reduced. By contrast, if they select a larger number of generations, parents, and/or offspring individuals, the application adjusts better the final image, but it takes a larger time to finish the execution.

IV. Programming Environment

To implement the two applications presented in this paper (i.e., TSPS and FotEvol) we used Processing [11], an open source development environment focused on visual applications.

Processing was initially designed as a programming language inspired by earlier languages, such as BASIC and Logo. It was created to serve as a software sketchbook and to teach computer programming fundamentals within a visual context. Processing has different programming modes to make it possible to deploy sketches on different platforms and program in different ways. The current default programming modes are Java, JavaScript, and Android. It incorporates over 100 libraries for handling images, shapes and video. Also, it has own methods included in its distribution, to draw simple geometric figures, to control mousse events, to capture images, etc. Other development environments, such as Eclipse, allow importing Processing projects as Java projects. It is interesting, since it allows to insert Processing applications into other Java applications (e.g., web pages, desktop or mobile applications). In addition, Processing allows the automatic generation of executable files for Windows, Mac OS, and Linux, in both versions (32 and 64 bits). We consider that using this developing environment, new useful applications to be used in the education environment could be developed.

V. Related Work

So far, the use of Graphical applications, such as games or interactive applications, has been widely extended as complementary learning, or learning assistants in different education areas.

More specifically, in Computer Science, some applications which rely on interactive or graphical learning platforms have been widely used by academia. Natvig and Line [6] presented ”Age of Computers” (AoC), an approach useful to be used on learning activities that supplements the auditorium lectures in a computer fundamentals course with 250 students. In particular, AoC is a computer game that presents to the students a diverse set of problems from the course topics linked to computer history. The application was firstly used in 2003, and it was very well welcomed by the students. This approach demonstrated to be useful in course activities which complement ordinary lectures, and the majority of the students clearly felt that they learned more through the game than solving traditional exercises.

Wang and Zhu [12] developed a software engineering education game in Second Life, a 3D online virtual environment. By playing the game, the students are able to correctly understand the software developing process. According to the data collected from student surveys, authors found that 92.3 percent of the 52 students who played their online software engineering process game thought that game was at least somewhat helpful.

Papastergiou [13] made a study to assess the learning effectiveness and motivational appeal of a computer game for learning computer memory concepts. The study consisted on 88 students randomly assigned to two groups, one of which used the gaming application, and the other one the non-gaming one. Data analyses showed that the gaming approach was both more effective in promoting students’ knowledge of computer memory concepts and more motivational than the non-gaming approach. The results suggest that within high school computer science, educational computer games can be exploited as effective and motivational learning environments.

Computer games were not only use in Computer Science learning environments. The Electronic Games for Education in Math and Science project (E-GEMS) [14], was a collaborative project centered at the University of British Columbia. E-GEMS involved researchers in Computer Science and mathematics education as well as teachers, children and professional game developers. In particular, E-GEMS was created in late 1992 to explore the potential of specially designed electronic games to increase learning and appreciation of mathematics and science by children. The obtained results demonstrated that games can be very effective in increasing both motivation and achievement in mathematics learning. However, E-GEMS project is no longer an active project. In addition, the use of games has also been tested in language teaching. For instance, Cai et al. [15] proposed an education game design model to teach Chinese as a foreign language. The development of the game not only allowed players to complete various tasks in virtual situations to achieve the educational purpose of learning Chinese language and culture, but also provided some reference for the future development of similar games. They tested their game with some foreign students from a Chinese university. After two months of testing, students that used the game improved their abilities in listening, speaking, and reading more than other students that used traditional teaching methods.

Closer to our proposal, other works are focused on

using graphical applications to the visualization of EAs. For example, ElAarag and Romano [16] presented an implementation of a system for understanding the Traveling Salesman Problem (TSP). Their goal was to graphically represent the states of two popular solutions to the TSP problem, namely backtracking, and branch and bound. In addition, they animated the transitions between those states. According to this, students can understand how to solve the problem, but unlike our work, their application is not interactive, and it was not designed to help students to clearly understand the advantages and drawbacks of this kind of optimization algorithms.

To sum up, graphical interactive applications can be used in order to help students to understand theoretical lessons in a more effective way. For this reason, we consider interesting to develop new graphical and interactive applications focused on facilitating the correct understanding of EAs' effectiveness and limitations.

VI. Conclusions

In this paper, we presented TSPS and FotEvol, two interactive graphical applications which were designed with the aim of helping students to correctly learn the operation of EAs in the Bachelor Degree in Computer Science. Using the proposed applications, we intend to give students a vision closer to reality of the scope of EAs and Evolution Strategies, encouraging them to apply these techniques to solve many optimization problems related to diverse fields, such as engineering, economics, social sciences, etc.

Although this is the first year that we are testing both applications, initial results are very promising. Students' motivation in the laboratory has grown, and we expect that the final results reflect the improvements obtained by the use of our applications.

In the future, we plan to develop additional learning applications focused on other Artificial Intelligence approaches (e.g., Artificial Neural Networks, Decision trees, Ant Colony optimization algorithms, etc.), since students also find these approaches difficult to understand.

Acknowledgments

This work was partially supported by the Government of Aragón and the European Social Fund (T91 Research Group).

References

[1] C. Darwin, *The Origin of Species*, Murray, London, UK, 1859.
[2] Greenwood, G., Lang, C., Hurley, S., *Scheduling tasks in real-time systems using evolutionary strategies*, Third Workshop on Parallel and Distributed Real-Time Systems, pp. 195-196, Apr. 1995.
[3] Saito, G., Corley, H.W., Rosenberger, J.M., Sung, T.-K., Noroziroshan, A., *Constraint Optimal Selection Techniques (COSTs) for nonnegative linear programming problems*, Applied Mathematics and Computation, vol. 251, pp. 586-598, 2015.
[4] R.E. Bixby, J.W. Gregory, I.J. Lustig, R.E. Marsten, D.F. Shanno, *Very large-scale linear programming: a case study in combining interior point and simplex methods*, Oper. Res., vol. 40, pp. 885-897, 1992
[5] Albu, A.B., *Learning Artificial Intelligence clip by clip: Post class reflections on the first online Norvig-Thrun-Stanford-Know Labs Artificial Intelligence course,* Frontiers in Education Conference (FIE), pp. 1-7, Oct. 2012
[6] L. Natvig, and S. Line, *Age of Computers: Game-based Teaching of Computer Fundamentals*, SIGCSE Bull, vol. 36, no. 3, pp. 107-111, 2004.
[7] Whitson, G., *An application of artificial intelligence to distance education,* Frontiers in Education (FIE), Nov. 1999
[8] Markov, Z.; Russell, I.; Neller, T.; Coleman, S., *Enhancing undergraduate AI courses through machine learning projects,* Frontiers in Education (FIE), Oct. 2005
[9] T. C. Raymond, *Heuristic Algorithm for the Traveling-salesman Problem*, IBM Journal of Research and Development, vol. 13, no. 4, pp. 400-407, 1969.
[10] E. Eiben, and J. E. Smith, *Introduction to Evolutionary Computing*, Springer-Verlag, 2003.
[11] B. Fry, and C. Reas, *Getting Started with Processing*, O'Reilly Media, 2010.
[12] T. Wang, and Q. Zhu, *A Software Engineering Education Game in a 3-D Online Virtual Environment*, First International Workshop on Education Technology and Computer Science (ETCS), pp. 708-710, 2009.
[13] M. Papastergiou, *Digital Game-Based Learning in high school Computer Science education: Impact on educational effectiveness and student motivation*, Computers & Education, vol. 52, no. 1, pp. 1-12, 2009.
[14] M. M. Klawe, *Computer Games, Education and Interfaces: The E-GEMS Project*, Proceedings of the Conference on Graphics Interface, pp. 36-39, 1999.
[15] L. Cai, F. Liu, and Z. Liang, *The research and application of education game design model in teaching Chinese as a Foreign Language*, IEEE International Conference on Progress in Informatics and Computing (PIC), pp. 1241-1245, 2010.
[16] H. ElAarag, and S. Romano, *Animation of the Traveling Salesman Problem*, Proceedings of IEEE Southeastcon, pp. 1-6, 2013.