Understanding Evolutionary Computing: A Hands on Approach

Trevor D. Collins
Knowledge Media Institute
The Open University UK
t.d.collins@open.ac.uk

Abstract— **Evolutionary computing is the study of robust search algorithms based on the principles of evolution. An Evolutionary Algorithm (EA) searches a problem space in order to find regions containing good solutions. Typically EA users judge the quality of their algorithms by the quality of the solutions found. This approach ignores the behavior of the search algorithm and concentrates solely on the outcome. As a result, the user is unaware of their algorithm's actions or how the solutions were discovered. This paper describes how search space visualizations can be used to facilitate the user's understanding of evolutionary computing. A set of examples are presented showing how the user can take a "hands on" approach to explore the behavior of their algorithms and interact with the evolutionary search process.**

Keywords— **Software Visualization, Search Space Matrices, Human-EA Interaction, Parameter Configuration, Initialization Methods, Interactive Evolutionary Algorithms.**

I. Introduction

Evolutionary Algorithms (EAs) are robust search algorithms based on the guiding evolutionary principle of "survival of the fittest". A typical EA works as follows: a random initial population of possible solutions to a problem is generated, these possible solutions are then evaluated using a problem specific evaluation function, and evolved to create a new generation of hopefully better solutions. The evolutionary process (i.e. the application of fitness biased selection and reproduction operators) is repeated until an acceptable solution (or set of solutions) to the problem is discovered.

Software Visualization (SV) has been defined as "the use of the crafts of typography, graphic design, animation and cinematography with modern human-computer interaction technology to facilitate the human understanding and effective use of computer software" [1]. The application of SV techniques to facilitate the design and application of EAs has been receiving growing attention during the last few years; [2], [3], [4], [5], [6], [7], [8], and [9].

A recent study on the use of Genetic Algorithms (GAs) concluded that GA users typically judge the quality of their algorithms by the quality of the solutions they find [10]. For example; by repeatedly running an algorithm to check that the solutions found can be repeated, by comparing the GA's solutions with those found by other techniques such as simulated annealing, or by comparing the solutions with those of benchmark examples. This "black-box" approach ignores the behavior of the search algorithm and concentrates solely on the outcome. As a result, the user is unaware of their algorithm's actions and how the solutions are discovered.

It is proposed that through the use of interactive SV techniques an EA designer can adopt a "hands on" approach to evolutionary computing and gain a more comprehensive understanding of their algorithm's behavior. This paper investigates how a new EA visualization technique [9] can be applied to enable not only the observation of an EA's search path, but also the user's interactive exploration of the search space and their possible involvement in the evolutionary process. Section II describes the visualization technique. Section III illustrates how this technique may be used to explore the search space. Section IV examines some of the new opportunities visualization offers the user for interacting with and becoming a part of the evolutionary process. Section V concludes with a summary.

II. EA Visualization

EAs search a problem space for a suitable solution or set of solutions by sampling the problem space and evaluating the sampled points. The sampling method biases evolution toward those areas containing "fitter" (i.e. better) solutions. The sample points are strings of symbols referred to as "chromosomes". The individual symbols in a chromosome are called "alleles" and each symbol's position in the chromosome is known as its "locus". A problem-specific evaluation function is used to evaluate each chromosome and assign a "fitness rating". These populations of chromosomes and their fitness ratings are the data used in EAs and they are what must be visualized in order to understand an algorithm's search behavior.

Fitness ratings are commonly visualized on a 2 dimensional line graph of fitness rating (on the y axis) versus generation number (on the x axis). This visualization provides an extremely useful insight into the

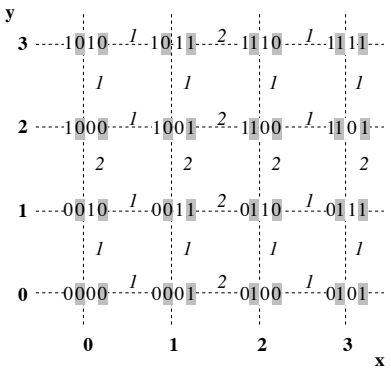

Figure 1. An example of how a four bit binary matrix can be constructed. Horizontal section's alleles are labeled with a white background and vertical section's alleles are labeled with a grey background. The Hamming distances between neighbors are shown in italics.

variation in fitness ratings during evolution, particularly when used to show the best, average and worst ratings in each generation. However, visualizing the sample points in the search space (i.e. the chromosomes) is less common and more problematic, due to the high dimensional nature of most problem spaces. Some techniques for displaying the population's chromosomes are reviewed in [11] and [9].

The technique adopted here is referred to as the "Search Space Matrix" [9]. In this approach a two dimensional data matrix representing the complete search space can be constructed by successively dividing a blank data matrix into vertical and horizontal sections and filling each section with alternate alleles from the coding alphabet. For example, a four bit binary search space containing 16 different chromosomes, 0000 to 1111, can be represented on a 4 by 4 data matrix as shown in Figure 1.

As the variables in a binary data set are of base 2 the blank matrix is divided horizontally into two halves. In the bottom half the first value of each entry is set to 0 and in the upper half it is set to 1. The matrix is then split vertically with the second value in the first two columns being set to 0, and the second value in the second two columns being set to 1. This process is repeated for the third and fourth values; the third value with one row of 0s, one row of 1s, one row of 0s and one row of 1s, and finally the fourth value with one column of 0s, one column of 1s, one column of 0s and one column of 1s (see Figure 1).

The above matrix construction method can be applied to any genotypic search space, irrespective of the length of the chromosome, or the number of possible alleles at each locus. Furthermore, rather than constructing a data matrix of the complete search space prior to visualization, the following direct linear equation can be used to translate a (high-dimensional) chromosome to a (low-dimensional) matrix coordinate:

$$\begin{matrix} coordinate\ in \\ dimension\ d \end{matrix} = \sum_{i=d}^{i \geq p} W_i \times X_i$$

Where X is a chromosome of length p. Each locus i has B_i different values, and the contributing weight of each locus is given by; $W_{1\ldots r} = 1$ and $W_{r\ldots p} = W_{(i-r)} \times B_{(i-r)}$. X_i is the position of each allele in the coding alphabet for locus i. r is the number of matrix dimensions (typically 2 or 3) and i is the current position marker indexed from the right at $i = 1$ in increments of r.

This equation can also be reversed to translate any coordinate back into its corresponding chromosome. Although this technique was developed independently, see [11], some similar approaches have since been discovered; William Shine and Christoph Eick have presented a coverage map of a Genetic Algorithm's (GA's) search using "quadcodes" [8], Ted Mihalisin, John Timlin and John Schwegler have produced a "hierarchical axes" technique for visualizing multivariate functions, data and distributions [12], and Jeffrey LeBlanc, Matthew Ward and Norman Wittels have proposed a "dimensional stacking" technique for exploring N-dimensional databases [13]. The remainder of this paper presents findings not considered by any of these similar techniques.

III. Exploring the Search Space

This section illustrates how the matrix representation described in the previous section can be used in practice. Four aspects of exploring the search space are examined here; visualizing the search path (III-A), the problems associated with limited screen space (III-B), the use of interactive controls for the visualization (III-C), and the use of alternate projections of the search space (III-D). An EA visualization environment called "Gonzo" [14] is used here to produce a set of practical examples.

A. An Example Search Path

A Genetic Algorithm (GA) evolving a solution to the F1 test problem proposed by De Jong [15] is used here as an example. This is a three dimensional problem i.e. it has three variables, each variable has a value between -5.12 and +5.12, and the evaluation function simply sums the squared values of each variable. The overall aim of the algorithm is to discover regions of the search space that have a minimum fitness rating.

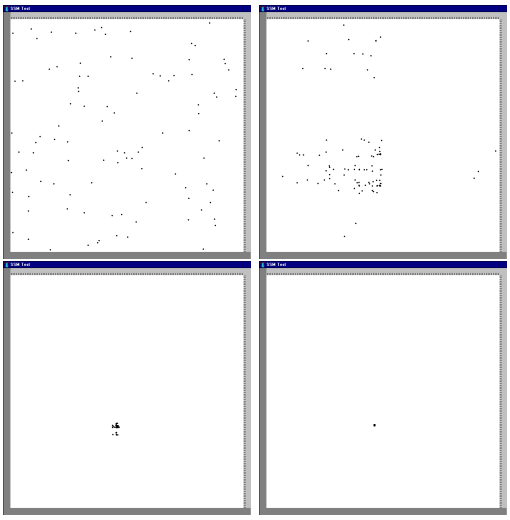

Figure 2. The evolution of a Genetic Algorithm solving De Jong F1 test problem. The four views show the search space matrix visualizations of the chromosomes contained in generation 0 (top left), 20 (top right), 40 (bottom left), and 64 (bottom right). Each dot represents a chromosome.

The genetic representation used for each variable is a 10 bit binary string, these are referred to as "genes", hence each 30 bit chromosome is made up of 3 genes, the first 10 bits (i.e. the first gene) refer to the first variable, the second 10 bits to the second variable, and the last 10 to the last variable. In order to map each (genotypic) binary gene to a (phenotypic) value, the gene is first translated into an integer, this is then divided by one hundred and 5.12 is subtracted from the result.

Figure 2 illustrates four of the sixty five steps taken along the GAs search path (i.e. the chromosomes in the population at generation 0, 20, 40 and 64). The initial population is a random distribution of points (Figure 2, top left) which converge over a number of generations to a single region (Figure 2, bottom right). This visualization can be used, either to present an on-line (live) view of an algorithm's progress, or an off-line (post-mortem) view based on the stored output data of the algorithm.

B. Screen Space vs Search Space

Typically, the resolution of the display medium (i.e. a computer monitor) is less than the resolution of the search space. In the example shown in Figure 2 the visualization of a 30 bit binary search space is a 32,768 by 32,768 matrix. This far exceeds the resolution of modern computer monitors and so raises the need for the user to be able to zoom in to and out of the search space visualization as well as pan across it.

Furthermore, by being able to use more than one pixel to represent each chromosome additional information can be presented. For example, the fitness rating of each chromosome can be indicated either by the dot's colour, size or value (i.e. gray scale). The schema structure of this representation also means that the user can directly identify the alleles contained within regions of the search space matrix (see section III-C, the schema highlighting controller). Hence, good and bad schema can be visually identified.

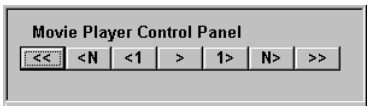

Figure 3. A movie player control panel used to step through an algorithm's evolutionary search path. Here the user can jump back to the first generation (<< generation 0), step back N generations ($< N$), step back one generation (< 1), play forward through the evolution a generation at a time ($>$), step forward one generation ($1 >$), step forward N generations ($N >$), or jump forward to the final generation ($>>$).

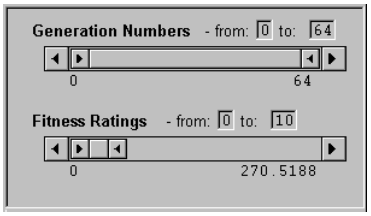

Figure 4. An image content controller used to identify the range of data to be displayed in the search space visualization. Two Alphasliders have been used here to identify two ranges; the generation numbers 0 to 64, and fitness ratings 0 to 10. The user can identify a range either by dragging the end sections of the central bar or inputing specific values. The range can then be moved either by dragging the middle section of the central bar, clicking on the outer scroll buttons on the left and right of the alphaslider, or by clicking in the gap between the central bar and outer scroll buttons.

C. Visualization Controls

Three control mechanisms are proposed here for the exploration of an EA's search behavior; a "movie-player" controller, an "image content" controller, and a "schema highlighting" controller. A *movie-player* controller enables the user to step backwards and forwards along an algorithm's search path and examine each step taken (see Figure 3).

As well as examining the evolution a step at a time, an *image content* controller can be used to identify a range of steps (i.e. generations) and fitness ratings, the display is then refreshed, and only the chromosomes contained within these ranges are displayed. Figure 4 shows how two "Alphasliders" [16], can be used to

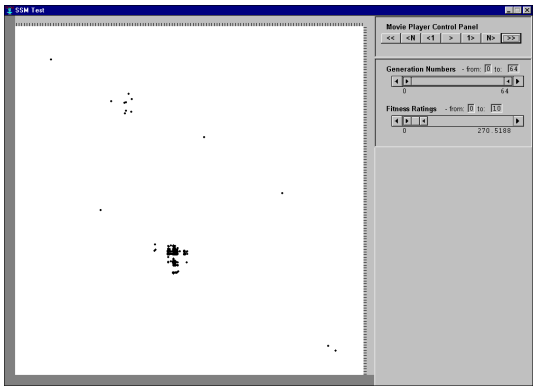

Figure 5. An example of how an image content controller can be used to identify ranges of data to be displayed. This figure illustrates the same search data as that shown in Figure 2, i.e. a GA solving the De Jong F1 test problem. In this view only the better chromosomes are shown from the GA's search path (i.e. those with a fitness rating in the range 0 to 10).

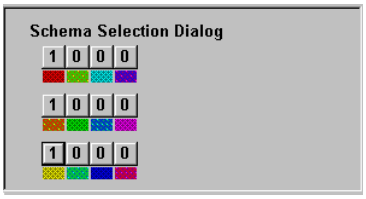

Figure 6. A schema selection dialog used to highlight the sections of the search space containing a chosen schema of interest. The three sets of four buttons represent the three genes in a 12 bit version of De Jong's F1 test problem. The user can click on any button to select the next allele in the selection set, this set contains each allele in the coding alphabet for that particular loci plus a wild card symbol (i.e. 01$*$). The schemata shown here identifies the optimum solution to the problem

enable the user to view all of the chromosomes considered during an algorithm's evolution (i.e. generation 0 to 64), and to reduce that view to focus in on the better chromosomes (i.e. those with fitness values in the range 0 to 10). This image content controller is also shown in Figure 5, this is being applied to illustrate the same data as used previously in Figure 2 (i.e. the output of a GA solving De Jong's F1 test problem). This view emphasizes how genotypically distinct the near-optimum solutions to this problem are.

Thirdly, the user can explore the schema contributions contained within their displayed search space using a *schema highlighting* controller (see Figure 6). Here the user can define a schemata of interest, in order to highlight the corresponding columns and rows of the matrix along which the defined values lie. Figure 7 shows how a schemata for a 12 bit (cut-down) version of the De Jong F1 test problem can be highlighted. The defined values in the schemata are highlighted by coloured strips in the legend of the search space visualization, each locus is indicated by a different colour. The location of the chromosome containing all of the defined values is indicated by the areas in the two legends containing the maximum number of coloured strips. A dashed line guide has been added to Figure 6 to emphasize the location of the chosen schemata.

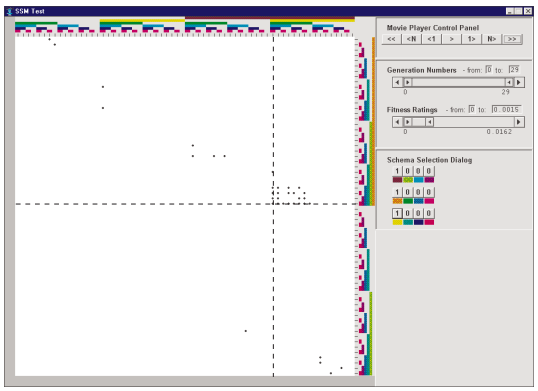

Figure 7. An example of how the regions of the search space containing a defined schemata can be highlighted. Here a 12 bit version of the De Jong F1 test problem is used, i.e. a 3D problem with variable values within the range of -0.08 to +0.08, these are represented here by three four bit binary genes. The defined values in the schemata are highlighted by coloured strips in the top and right hand legends of the matrix. A dashed line guide has been added here to emphasize the location of the chosen schemata.

Using these three control mechanisms the user can; follow the search path of their algorithm, identify the fitter areas of the search space, and examine the impact of individual building blocks on the chromosomes' fitness ratings.

D. Alternate Points of View

The projection used to identify each chromosome's coordinate position in Figure 2 was ordered by the most significant bit from each gene (i.e. 1st gene 1st locus, 2nd gene 1st locus, 3rd gene 1st locus, 1st gene 2nd locus, etc.) rather than by the most significant bit from the chromosome (i.e. 1st gene 1st locus, 1st gene 2nd locus, etc., as in Figure 1). Either projection is possible but as this problem is divided into three equally weighted variables this projection was chosen as the most appropriate. The ability to re-order the projection of the chromosomes loci means that the user can view the relationships between the chromosomes alleles from a number of different "angles".

IV. Human-EA Interaction

Having identified a technique for visualizing high-dimensional genotypic search spaces (II) and illustrated how that technique can be applied to realistic problems (III), the following section goes on to describe how interaction is currently being used in evolutionary computing and explores how this can be improved. Examples of three areas in which visualization facilitates EA interaction are explained here: EA parameter configuration, EA initialization, and interactive EAs. The following three subsections identify the limitations of the existing working practice in each of these areas before proposing an alternative.

A. EA Parameter Configuration

The effects that different parameter settings have on an EA is an important aspect of the designer's expert knowledge. Without an on-line search space visualization the designer can only retrospectively analyze the effects of their parameter settings from their algorithm's output data. Several graphical EA environments permit the user to interactively vary the parameter settings during evolution (for example "*GA Meter*" [3], "*GIGA*" [6]), however these tools do not visualize the algorithm's search space and therefore it is still difficult to relate any parameter changes to the search behavior of the algorithm. With a search space visualization (such as [14]) the user can observe their algorithm's evolution and monitor the effects of a variety of parameter settings on their algorithm. This is not just a useful pedagogical tool, it can also be applied to "fine tune" the behavior of an algorithm during evolution.

B. EA Initialization Methods

In a canonical GA (such as the SGA model described by Goldberg [17]) the initial population is created as a set of random valued chromosomes. Several alternative methods have since been explored in order to improve the performance of such algorithms. Bramlette [18] introduced an initialization method that selects the best individuals from a set of randomly created chromosomes. Kallel and Schoenauer [19] examined a "*Uniform Covering*" method that ensures an even distribution of each allele in the coding alphabet throughout the initial population, and a "*Homogeneous Block*" method that favors sequences of 0s and 1s in the chromosomes of the initial population.

Any of these approaches would benefit from visualization. Bramlette's selection of the best "N" individuals to create the initial population would be visually reinforced with the state space matrix view presented in section II. Similarly Kallel and Schoenauer's comparison of alternative initialization methods could be supported by visualizing the individual search paths taken by the same algorithm being applied a number of times to the same problem but from a variety of initial states.

An interactive visualization tool (such as Gonzo) can also be used to create or edit the initial population for an EA. As the translation method used in search space matrices is a two way mapping (i.e. any chromosome can be translated to a unique coordinate and any coordinate can be translated to a unique chromosome) the user can explore the search space and identify individuals for inclusion in the population. Hence, the user can not only set the initial population, but also re-introduce diversity or guide convergence toward interesting areas in the search space at any stage during evolution.

C. Interactive EAs

Interactive Evolutionary Algorithms (IEAs) are a sub-set of EAs whose origin has been attributed to Richard Dawkins book "The Blind Watchmaker" [20]. The use of IEAs can be described as a two stage process in which the user is first presented with every individual in the population in an appropriate (problem specific) form and then asked to evaluate each individual based on its perceived merit [21]. Essentially the user takes on the role of the evaluation function, removing the need to formally specify the problem evaluation criteria. This approach has been applied to several novel problems such as graphic art [22], music [23] and knowledge discovery in databases [21].

The current use of IEAs limits the user's view of the search space to those points sampled by the current population and provides no information relating a chromosome's structure to the features of the resulting representation. Therefore, the user is unable to do any form of credit assignment for a chromosome's schemata. By combining the first step in this process (i.e. displaying a chromosome in an appropriate form) with a search space visualization, such as that described in section II, the user can see each chromosome's place in the search space and judge an individual not just on its phenotypic appearance but also on its genotypic structure, and ability to contribute toward new solutions. This enables the user to see the "bigger picture" outside of the sub-space sampled by the current population.

V. Conclusion

Software visualization offers much more to evolutionary computing than a mere display: it offers the user an opportunity to interactively explore evolutionary computing. One such visualization technique, the "Search Space Matrix", has been described. A tool using this

technique was used in order to highlight how SV can be applied to interactively explore the behavior of a GA. By adopting more of a "hands on" approach to evolutionary computing it is intended that a better understanding of our algorithm's search behavior will be achieved.

In addition to exploring the search path of an algorithm, this technique can be used to enable user interaction at any stage in the evolutionary process. Three extensions to the current working practice of EA designers have been proposed in the areas of; parameter configuration, population initialization, and interactive evaluation. It is through the exploitation of interactive visualization techniques, such as those presented here, that people may further their understanding of EAs and the additional benefits of "Human-EA Interaction" may be achieved.

The use of SV to facilitate both the understanding and effective use of evolutionary computing is the focus of a number of current research projects. This offers new opportunities for those involved in evolutionary computing to interact with their algorithms during evolution. However, little work on the efficacy of this approach has been done. Future work will involve empirically assessing not only these forms of visualization, but also the opportunities for human interaction and intervention that they create.

Acknowledgments

I would like to thank the Engineering and Physical Sciences Research Council for funding this work (EPSRC Studentship 94315065), the members of the Open University's Knowledge Media Institute who helped comment on earlier drafts of this document, particularly John Domingue, Tony Hirst, Simon Masterton, Paul Mulholland, Marco Ramoni and Stuart Watt, and the anonymous reviewers for their constructive comments on the original submitted version of this paper.

References

[1] B. Price, R. Baecker, and I. Small, "A principled taxonomy of software visualisation," *Journal of Visual Languages and Computing*, vol. 4, pp. 211–266, 1993.

[2] H. Cartwright and G. Mott, "Looking Around: Using clues from the data space to guide genetic algorithm searches," in *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA'91)* (R. Belew and L. Booker, eds.), pp. 108–114, 1991.

[3] A. Kapsalis and G. Smith, *The GAmeter Development Toolkit User Interface Manual*, 1992.

[4] T. Routen and T. Collins, "Visualisation of A.I. techniques," in *Proceedings of the International Conference on Computer Graphics and Visualization (COMPUGRAPH'93)*, (Portugal), ACM Press, 1993.

[5] B. Nassersharif, D. Ence, and M. Au, "Visualisation of evolution of genetic algorithms," in *Proceedings of the World Congress on Neural Networks*, vol. 1, (San Diego, CA., USA), pp. 1–560–1–565, 1994.

[6] T. Dabs and J. Schoof, "A graphical user interface for genetic algorithms," Tech. Rep. 98, Lehrstuhl fur Informatik II, University Wurzburg, DE., February 1995.

[7] R. Dybowski, T. Collins, and P. Weller, "Visualization of binary string convergence by sammon mapping," in *The Fifth Annual Conference on Evolutionary Programming (EP96)* (L. Fogel, P. Angeline, and T. Baeck, eds.), (San Diego, CA.), pp. 377–383, MIT Press, 1996.

[8] W. Shine and C. Eick, "Visualizing the evolution of genetic algorithm search processes," in *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*, pp. 367–372, 1997.

[9] T. Collins, "Using software visualization technology to help evolutionary algorithm users validate their solutions," in *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA'97)*, (East Lansing, MI., USA), pp. 307–314, August 1997.

[10] T. Collins, *The application of software visualization technology to evolutionary computing: A case study of genetic algorithms*. PhD thesis, Knowledge Media Institute, The Open University, Milton Keynes, UK., 1998.

[11] T. Collins, "Genotypic-space mapping: Population visualization for genetic algorithms," Tech. Rep. KMi-TR-39, The Knowledg Media Institute, The Open University, Milton Keynes MK7 6AA, UK., September 1996.

[12] T. Mihalisin, J. Timlin, and J. Schwegler, "Visualizing multivariate functions, data, and distributions," *IEEE Computer Graphics and Applications*, pp. 28–35, May 1991.

[13] J. LeBlanc, M. Ward, and N. Wittels, "Exploring n-dimensional databases," in *Proceedings of the IEEE Conference on Visualization 1991*, (San Francisco, CA., USA), pp. 230–237, October 1991.

[14] T. Collins, "Gonzo: An evolutionary algorithm visualization tool," January 1997. Details available from http://kmi.open.ac.uk/~trevor/Viz/tools/.

[15] K. D. Jong, *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.

[16] C. Ahlberg and B. Shneiderman, "The Alphaslider: A compact and rapid selector," in *Proceedings of International Conference on Human Factors in Computing Systems (CHI'94)* (B. Adelson, S. Dumais, and J. Olson, eds.), (Boston Massachusetts, USA), pp. 365–371, ACM Press, 1994.

[17] D. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*. Addison Wesley, 1989.

[18] M. Bramlette, "Initialisation, mutation and selection methods in genetic algorithms for function optimization," in *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA'91)* (R. Belew and L. Booker, eds.), pp. 100–107, 1991.

[19] L. Kallel and M. Schoenauer, "Alternative random initialization in genetic algorithms," in *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA'97)*, (East Lansing, MI., USA), pp. 268–275, August 1997.

[20] R. Dawkins, *The Blind Watchmaker*. Longman, 1986.

[21] G. Venturini, M. Slimane, F. Morin, and J. A. de Beauville, "On using interactive genetic algorithms for knowledge discovery in databases," in *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA'97)*, (East Lansing, MI, USA), pp. 696–703, August 1997.

[22] S. Todd and W. Latham, *Evolutionary Art and Computers*. Academic Press, 1992.

[23] G. Nelson, "Sonomorphs: An application of genetic algorithms to the growth and development of musical organisms," in *Proceedings of the Fourth Biennal Art and Technology Symposium*, (Connecticut, USA), pp. 155–169, March 1993.