Interactive One-Max Problem Allows to Compare the Performance of Interactive and Human-Based Genetic Algorithms

Chihyung Derrick Cheng and Alexander Kosorukoff

University of Illinois at Urbana-Champaign, Urbana, Illinois 61801
{cdcheng,kosoruko}@uiuc.edu

Abstract. Human-based genetic algorithms (HBGA) use both human evaluation and innovation to optimize a population of solutions (Kosorukoff, 2001). The novel contribution of HBGAs is an introduction of human-based innovation operators. However, there was no attempt to measure the effect of human-based innovation operators on the overall performance of GAs quantitatively, in particular, by comparing the performance of HBGAs and interactive genetic algorithms (IGA) that do not use human innovation. This paper shows that the mentioned effect is measurable and further focuses on quantitative comparison of the efficiency of these two classes of algorithms. In order to achieve this purpose, this paper proposes an interactive analog of the one-max problem, suggests human-based innovation operators appropriate for this problem, and compares convergence results of HBGA and IGA for the same problem.

1 Introduction

Interactive genetic algorithms (IGA) had extended evolutionary computation (EC) to the areas where computational evaluation is not possible. This extension was made through the use of selection that is based on human evaluation (Herdy, 1996). Nowadays there is a growing field of IGA applications from music composition to architectural design (Takagi, 2001). A human-based genetic algorithm (HBGA) did a similar thing by extending EC even further to the area where it is hard or impossible to find a good representation and usable computational innovation operators (Kosorukoff, 2000). The task of searching solutions expressed in natural language is one of such problems. It could not be approached by IGAs simply because we do not know how to do computational recombination in a natural language. HBGAs have solved this problem by outsourcing innovation operators to humans in the same way as IGAs did earlier with outsourcing evaluation. However, so far it has remained unclear if HBGAs and human-based innovation operators are strictly limited to such areas where human recombination is the only option or they can play a role in a field where IGAs are applied. In this paper, we suggest an example of a problem for which both HBGAs and IGAs are applicable. Although it is not very useful practically,

K. Deb et al. (Eds.): GECCO 2004, LNCS 3102, pp. 983–993, 2004.

this problem makes a good simple benchmark for both IGAs and HBGAs. Hence it can play a role of the one-max problem for genetic algorithms using human interaction. In this paper, we describe this problem in detail and perform a set of experiments to compare the speed of convergence of those algorithms.

The rest of this paper is organized as follows: section 2 reviews recent research on HBGAs, section 3 describes the application, section 4 describes the problem we use for experiments, section 5 describes experimental settings, and section 6 contains results of experiments. Finally, we conclude with discussion of the results in section 7.

2 Background

Free Knowledge Exchange (FKE) project (3form.com, 1998) was the first application of a technique now known as *human-based genetic algorithm* (HBGA) for evolutionary knowledge management and collaborative problem solving. The main goal of the project was efficient knowledge discovery, sharing, and innovation. The technology got its current name and was systematically analyzed after it had received several successful evaluations from its users (Kosorukoff, 2000; Kosorukoff, 2001). The idea of HBGA was mainly borrowed from a business practice of outsourcing where operations that are not a part of the core competence of an organization are delegated to some external agents which are competent enough to choose their own method of executing those operations such that the purpose of the operations can be accomplished.

Kosorukoff and Goldberg (2002) discuss HBGAs among two approaches to evolutionary organization and innovation. Defaweux et al. (2003) discuss an application of HBGA for storytelling and developing of marketing slogans, examines similarities between HBGA and a traditional GA. Based on experiments with small groups of people, the authors show that HBGA stimulates creativity and consensus, concluding that it has a great potential in fields like marketing, industrial design, and creative writing. Goldberg et al. (2003) discuss HBGA as a part of distributed innovation support system.

3 HBGA for Color Fitting

It is appropriate to say that the first application of HBGA has unintentionally limited the range of its further applications to the tasks connected with evolution of text messages expressed in some natural language. However, in this section we will try to show that actually there is no such limitation. Our simple example will be from a different domain which is classical to an IGA, but new to HBGA: finding a color according to a user preference.

We do not have a problem with finding a genetic representation for color. The most common way to represent color is a triple of R, G, B values, where 8 bits are allocated for each component, making 24 bits in total (Foley 1990; Hunt, 1992). We will use this standard representation for our genetic coding. Actually, RGB colormap is not the best for the purpose of designing a practical

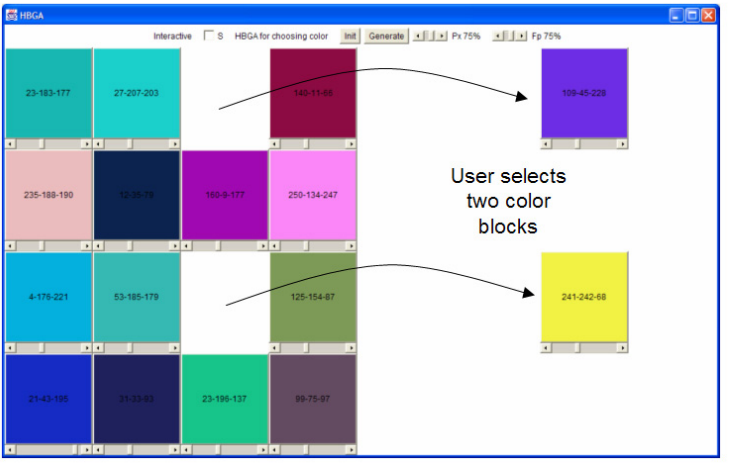

Fig. 1. An interface of HBGA for the interactive one-max problem

application, we would use HSV or LUV for this purpose (Douglas, 1996; Schwarz, 1987). However, since colormap conversions is far from the topic of this paper the performance was sacrificed for simplicity and clarity. Here is a straightforward example how purely white color $(R, G, B) = (255, 255, 255)$ is coded:

$$11111111\ 11111111\ 11111111$$

It is clear that we can use a classic set of genetic operators to work with this representation.

User interface is an important part of any interactive program. Repetitive use of the same interface in interactive evolutionary computation makes its design especially important, so we describe it in detail. A user is presented with 16 (population size) of colors (Figure 1). There are two panels for this purpose, the left one is for unselected colors, the right one is for selected colors. Each panel is divided into 4 by 4 grid and each cell color corresponds to an individual in the population. The user selects color by clicking on the left panel with a mouse that makes the selected cell move into the right panel. The cell in the right panel can be deselected by another click of a mouse and moved back into the left panel. This is all controls that IGA needs. HBGA, however, should allow user to create new colors in addition to those created by usual mutation/crossover operators (Goldberg, 1998) over the binary representation described earlier.

There are many ways to create an innovation interface. FKE application (Kosorukoff, 2000) uses cut and paste controls to allow for human-based recombination of textual information. This way would work here by allowing human to modify genotype directly, though this is not a natural way for people to modify color. In a practical HBGA application, different controls to adjust such parameters as brightness, hue, and saturation would be present, as well as a control element providing the opportunity to mix two or more colors as artists usually

do. However, some of these operators when applied to our task will allow convergence in one or two iterations and this is not good for our research purpose of algorithm comparison. Therefore, we had chosen operators from classical GAs for research purpose only.

Before we will proceed with description of human-based innovation operators, we have to clarify how their computational peers are performed in our algorithm. There is essentially no difference in crossover, except that parents are stored together with individuals because they will be used later. The computational mutation operation is the following. The program will first select a parent from the population, then another 24-bit binary string (mutation mask) will be generated randomly such that the probability of 1 in each position equals to the probability of mutation p_m. A mutant is generated by XOR operation over two strings. Both parents and a mutation mask are stored with an individual for further use.

For human-based operators, each color cell in our HBGA has a slider associated with it. If a cell represents a result of one-point crossover, the slider can be used to move the locus of crossover among 23 possible positions. If a cell represents a result of mutation, the slider can rotate the mutation mask (a binary string in which each locus of mutation is marked as 1, other loci are 0) to the right or to the left depending where slider is moved. These operations allow human user to modify the result of genetic operators, so innovation in this algorithm is a collaborative result of human-computer interaction.

There are two modes of operation of the user interface. When we are in the IGA mode sliders are not shown and we are only allowed to select. When the HBGA mode is enabled we are allowed both to select and to modify individual colors in the population. According to our experience, it does not take many generations to play with this toy and converge to the colors of our preference. This shows that both algorithms are working but we want something more than just user satisfaction, something that allows for a more precise comparison.

4 Interactive One-Max Problem

In a usual IGA or HBGA application each user follows her own goal determined by her preferences. However, in order to get some measurable results we need to fix the goal as we cannot compare different preferences which can require different amount of effort to achieve them. We had borrowed our goal from the one-max problem, so we require to converge to the white color $(255, 255, 255)$ or:

$$11111111\,11111111\,11111111$$

Despite of its apparent similarity with the one-max problem, the interactive one-max problem has also some important differences:

1. People rarely can distinguish colors that are close to each other in the color space, especially if they are not placed close to each other during comparison. This is unlike the usual notion of fitness which distinguish very fine grades of solution quality.

2. The actual form of fitness function is not known to us. One user can follow the brightness of colors, another can follow hue and saturation. Many strategies of achieving the goal are possible. The efficiency of these strategies will not be the same, but for the purpose of this paper we are interested only in the averaged performance and how it depends on the type of the algorithm that was used.

We have to address these two issues and we do it in the following two subsections.

4.1 Defining White

We address the first difference by allowing a certain range of colors to be "white" as determined by a margin. The RGB value of pure white color is (255, 255, 255). We relax our requirement considering any color with all components greater than 245 to be white. The ability of people to distinguish colors is not the same and we wanted that most of the users could achieve the goal with reasonable amount of effort. The range of colors that fall within a margin is generally dependent on the choice of progress measure. Criteria will be described shortly, but as a general rule we treat color as white if it is not inferior to (245,245,245) according to our selected criteria.

4.2 Progress Criteria

In our case, we do not have a direct access to a fitness function that humans use when doing their selections. However, it is convenient to have some value similar to fitness that we can track to know how close we are to achieve the goal. We have three candidates for this purpose. They are brightness (M1), Euclidean metric (M2), and minimum component metric (MS):

$$M1(R,G,B) = L1((255,255,255),(0,0,0)) - L1((255,255,255),(R,G,B))$$
$$= 255 * 3 - ((255-R)+(255-G)+(255-B)) = R+G+B$$

$$M2(R,G,B) = L2((255,255,255),(0,0,0)) - L2((255,255,255),(R,G,B))$$
$$= 255 * \sqrt{3} - \sqrt{(255-R)^2+(255-G)^2+(255-B)^2}$$

$$MS(R,G,B) = LS((255,255,255),(0,0,0)) - LS((255,255,255),(R,G,B))$$
$$= 255 - \max(255-R, 255-G, 255-B) = min(R,G,B)$$

We have no preference among them at this point, so we will use all three in our experiments. The identification of the fitness function people actually use in our experimental task could be a topic of another research.

5 Experiment Settings

We compare IGA and HBGA in two categories: generational and steady state. Each user is asked to achieve the white color through a sequence of selections only (IGA), and through a sequence of selections and modifications (HBGA).

The program will generate 16 color cells at each generation and present them to a user. Probability of computational crossover and mutation were as usual $p_x = 0.75$, $p_m = 1/24$. Another parameter F_p determines the probability of selection of preferred individuals (the right panel) for reproduction, and thus determines the selective pressure. In our experiment $F_p = 0.9$, i.e. colors picked by a user have 90% chance of being selected for reproduction, while other colors have only 10% chance.

We invited 10 people to compare the algorithms and use intra-subject experimental design that suggests that each person tries to achieve white color using four algorithms in the following sequence:

- IGA-Generational
- HBGA-Generational
- IGA-SteadyState
- HBGA-SteadyState

None of the users had any background in evolutionary computation and supposedly did not understand the mechanism behind evolutionary programs they are using. We had recorded every color in each generation, so that the best fit in brightness, Euclidean and MS metric can be calculated. Then we had averaged experimental results over ten replications.

6 Results

Figures 2-7 present the graphs of progress metrics: brightness (M1), Euclidean metric (M2), and minimum component metric (MS). For each metric first IGA-Generational and HBGA-Generational, then IGA-Steady State and HBGA-Steady State are compared. In each plot, there are two horizontal lines. The upper one represents the optimum values. The pure-white color $(255, 255, 255)$ should behave like this line. Similarly, the bottom one represents the margin cut-off. It is based on our minimum requirement of white color $(245, 245, 245)$. The experiment is considered successful if the experiment sample passes through the cut-off margin. The same information in summarized form is presented in Table 1.

Overall, the curves of four algorithms increase over the generations and show convergence. The curves for HBGA usually show an advantage starting from the first generation. Both HBGA and IGA are able to accomplish the task successfully. However, the number of generations spent differs in about 3 times favoring HBGA.

Measuring convergence in numbers of iterations does not give a complete picture. Time comparison in Table 2 adds the missing part. A generation of our

Table 1. Performance comparison for different progress metrics

	Generational			Steady State		
Progress metric	HBGA	IGA	HBGA Speedup	HBGA	IGA	HBGA Speedup
M1	4	13	3.25	4	13	3.25
M2	6	14	2.33	5	14	2.8
MS	11	27	2.45	6	19	3.17

Table 2. Time performance comparison. Generations are shown according to MS criterion

	Generational			Steady State		
	HBGA	IGA	HBGA Speedup	HBGA	IGA	HBGA Speedup
Generation time (s)	9.2	5.5	-	14.2	6.5	-
Generations	11	27	-	6	19	-
Time to converge (s)	103.8	147.6	1.42	92.3	125.7	1.36

HBGA is approximate 2 times slower than a generation of an IGA. The higher speed of convergence in terms of generations, however, compensates the slower time of each HBGA generation, so our experiments show that HBGA has an advantage in time to converge as well as the number of generations to converge.

The time of each generation in HBGA is dependent on a strategy that a particular user chooses when using HBGA. It is clear that one can use HBGA exactly as IGA just by ignoring all color modification sliders, in this case the time of each generation and the number of generations to converge should be equal to those of IGA. If one will use color modification heavily, then the time of each generation will likely to grow, but the number of generations needed to converge decreases. We didn't investigate the influence of different users' strategies in this research.

There is a discrepancy in the average time of the same algorithm in generational and steady state mode, which is quite high and unexpected. There are two possible reasons for those discrepancies. One possible reason connected with the sequence of our experiments and human factors (Berk, 1982), in particular, the fact that user fatigue increases over time, so we see slower response on successive experiments. We cannot reliably establish this fact from our set of 40 experiments which were not designed to take user fatigue into account and have no meaning to measure it.

7 Discussion

This paper has suggested a way to measure the performance of human-based genetic algorithms quantitatively. We compared the results of these two algorithms in their generational and steady-state implementations. The experimental results

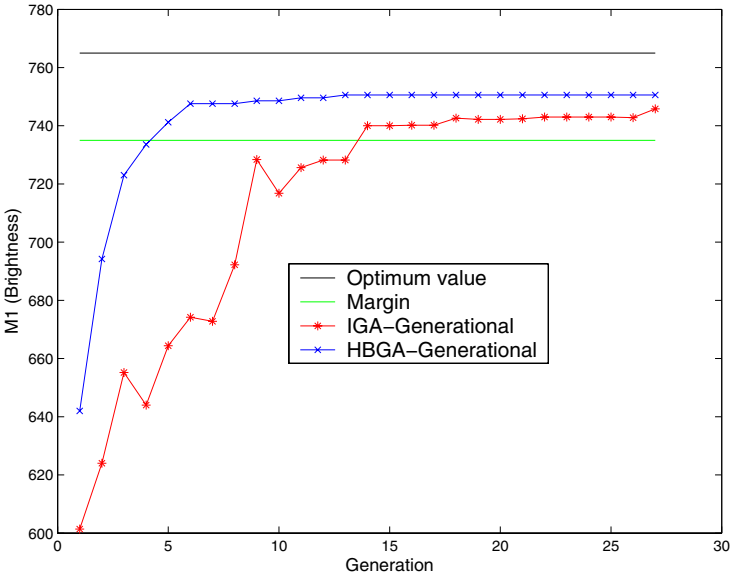

Fig. 2. Brightness (M1) of the brightest individual in the population for generational type of IGA and HBGA. The speedup factor of HBGA is 3.25

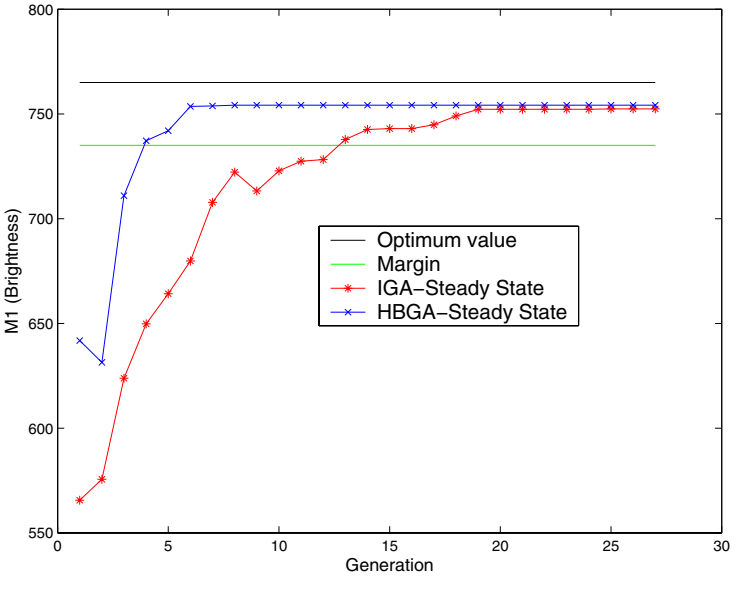

Fig. 3. Brightness (M1) of the brightest individual in the population for generational type of IGA and HBGA. The speedup factor of HBGA is 3.25

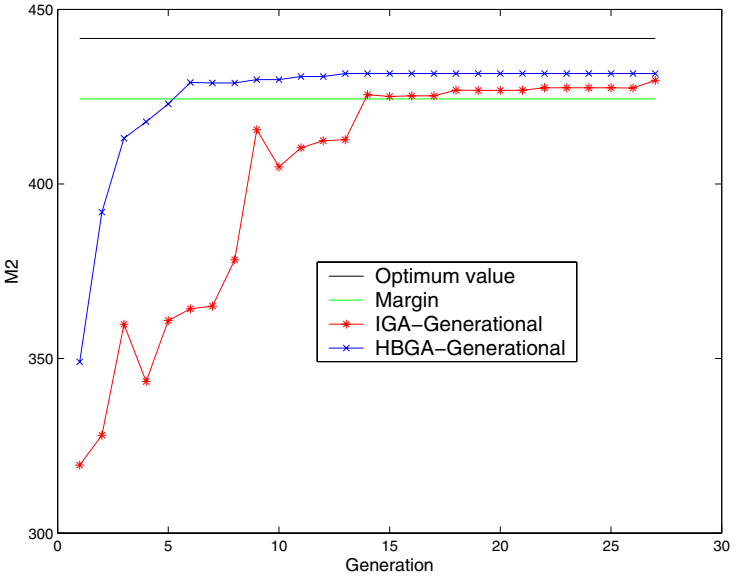

Fig. 4. Measure M2 for generational type of IGA and SGA. The speedup factor of HBGA is 2.33

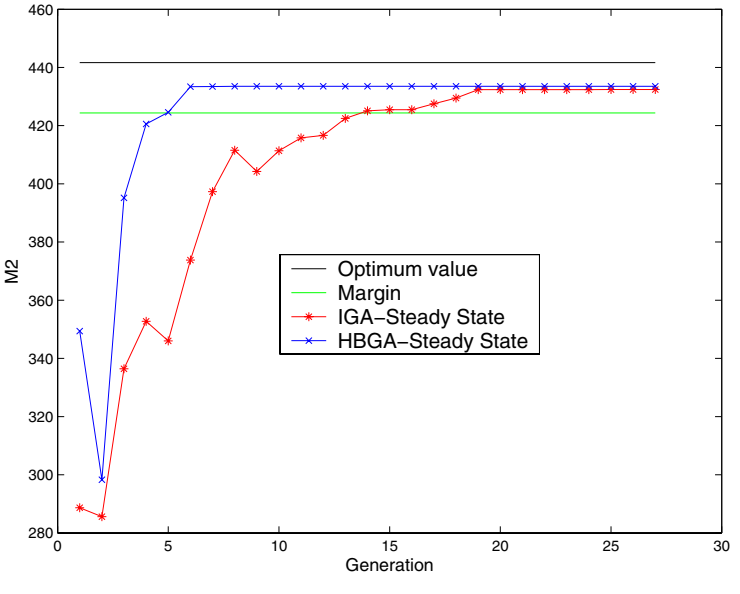

Fig. 5. Measure M2 for steady state type of IGA and HBGA. The speedup factor of HBGA is 2.8

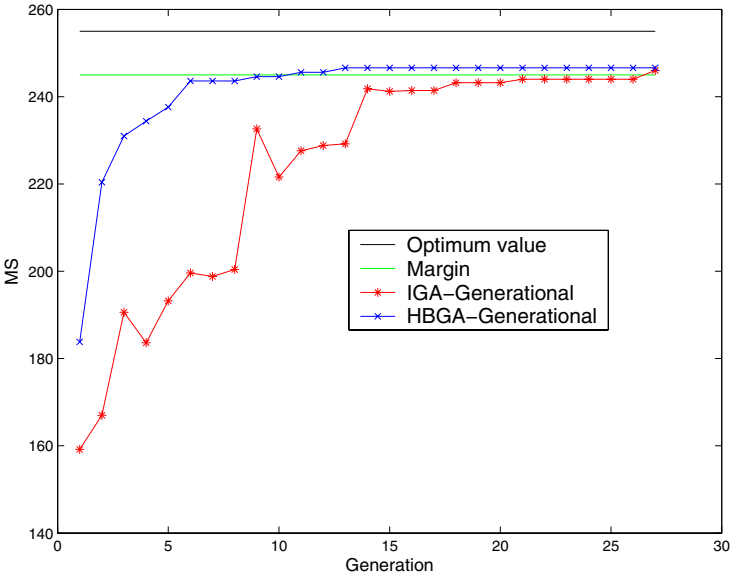

Fig. 6. Measure MS for generational type of IGA and HBGA. The speedup factor of HBGA is 2.45

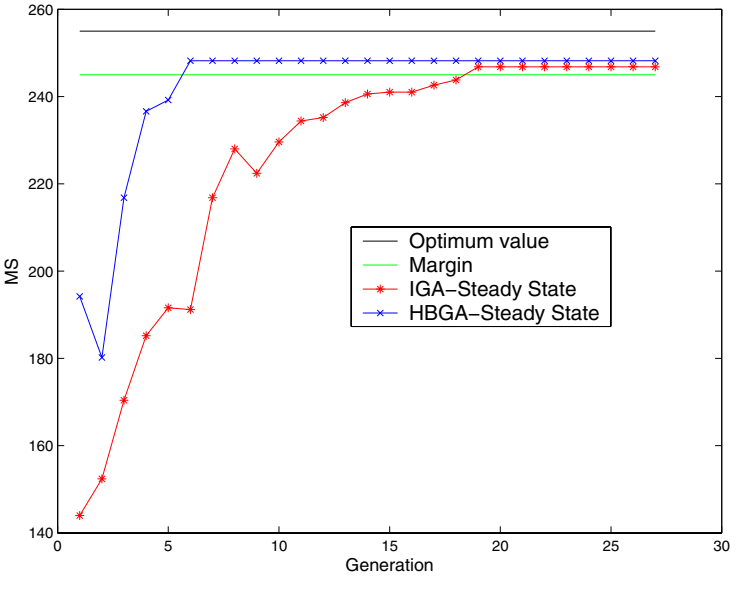

Fig. 7. Measure MS for steady state type of IGA and HBGA. The speedup factor of HBGA is 3.17

strongly support the idea that using human-based innovation operators can be advantageous even in the area where we actually can perform all innovations computationally. Algorithm using human-based innovation operators requires 2-3 times less generations to converge. Supplemental materials for this research can be found online at
`http://www.derrickcheng.com/Project/HBGA/index.html`.

Acknowledgements. The authors would like to thank Alexander Kirlik and Sarah Miller for sharing their expertise in human factors and experimental design. We also thankful to all people who helped us to evaluate the performance of the algorithms examined in our experiments. Finally, we want to express our thankfulness to the reviewers of GECCO for their valuable suggestions.

References

1. Berk, T., Brownston L. and Kaufman, A. (1982). A human factors study of color notation systems for computer graphics. *Communications of the ACM 25, 8*, pp. 547-550.
2. Defaweux, A., Grosche, T., Karapatsiou, M., Moraglio, A., Shenfield, A. (2003). Automated Concept Evolution. Tech Report
3. Douglas S., Kirkpatrick T. (1996). Do color models really make a difference?. *Proceedings of CHI-1996.*
4. Foley, J., van Dam, A., Feiner, S. and Hughes, J. (1990). Computer Graphics: Principles and Practice, 2nd. ed. Addison Wesley.
5. Goldberg, D., Welge, M., Llora‘, X. (2003). Distributed Innovation and Scalable Collaboration In Uncertain Settings. IlliGAL Report No. 2003017.
6. Herdy, M. (1996). Evolution strategies with subjective selection. In *Parallel Problem Solving from Nature, PPSN IV,* Volume 1141 of *LNCS*, pp. 22-31.
7. Hunt, R. (1992). Measuring Color, 2nd. ed. Ellis Horwood.
8. Kosorukoff, A. (2000). Human based genetic algorithm. Online at `http://www.hbga.org/hbga.html`.
9. Kosorukoff, A. (2001). Human based genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-2001*, 3464-3469.
10. Kosorukoff, A., & Goldberg, D. (2002). *Genetic algorithm as a form of organization, Genetic and Evolutionary Computation Conference, GECCO-2002.*
11. Takagi, H. (2001). Interactive Evolutionary Computation: Fusion of the Capacities of EC Optimization and Human Evaluation. *Proceedings of the IEEE 89, 9*, pp. 1275-1296.