Reducing User Fatigue in Interactive Genetic Algorithms by Evaluation of Population Subsets

Juan C. Quiroz, Sushil J. Louis, *Member, IEEE,* Amit Banerjee, and Sergiu M. Dascalu *Member, IEEE,*

***Abstract*—We present an analysis of user fatigue mitigation techniques in interactive genetic algorithms (IGAs). A user guides the evolutionary process by picking every t generations the solution the user likes best out of a small subset selected from a large population size. We explore how the selection of individuals from the population to be displayed to the user for subjective evaluation affects the convergence of the IGA. We show that we can reduce the number of evaluations performed by a user with our evaluation technique by tenfold compared to a standard IGA using a small population size. Our approach is tested on the Onemax problem in order to analyze how the IGA performance varies in a well known fitness landscape. In using the Onemax problem, we are interested in showing the ability of a user to guide the population towards areas of high fitness, rather than finding a single optimal solution. We study the effect of different subset selection strategies, different subset sizes, different choices of t, the presence or absence of collaboration, and differing amounts of noise on performance. Results show that using our fitness interpolation technique a user can effectively bias the interactive evolutionary search towards high fitness solutions despite only evaluating a small fraction of the total number of individuals.**

***Index Terms*—Interactive genetic algorithm, user fatigue**

I. Introduction

INTERACTIVE genetic algorithms (IGAs) differ from genetic algorithms (GAs) in that the objective fitness evaluation is replaced with user evaluation. This allows the user to guide an explorative evolutionary process when there is no better fitness measure than the one in the human mind [19]. As such, IGAs can incorporate intuition, emotion, and domain knowledge from the user. An often overlooked aspect of solving problems in subjective domains is the benefit gained by the user through exploration of the search space by interacting with the genetic algorithm. A user may not find an optimal solution, but through exploration and exposure to many solutions, the user can gain a deeper understanding of the problem, allowing the user to adapt and change the original concepts and requirements initially given.

While IGAs are a powerful tool, their reliance on user computation presents several issues, amongst the most notable being user fatigue. GAs usually rely on the use of large population sizes running for hundreds of generations to achieve satisfactory results [10]. Such computational dedication cannot be expected from the user due to psychological and physical fatigue, boredom or losing interest in continuing the evaluation of solutions. Thus, how best to incorporate user input into the IGA process remains a significant research challenge [19]. Related work on techniques for mitigating user fatigue in IGAs include using a small population size, presenting a representative subset of a large population for user evaluation, using a small number of generations, and using aggressive convergence methods to ensure convergence within a small number of generations [19].

In our fitness interpolation technique, instead of presenting the user with a small population, we ask the user to evaluate a small subset of a large population size every t generations, hence mitigating the effect of genetic drift as a consequence of using a small population size. This technique has been introduced as part of our previous work [1], [15], [16], [17]. The user is asked to select either the best, or the best and worst individuals from the subset displayed, and the fitness of every other individual in the population is estimated based on similarity to the user selection(s). In the work previously presented, we combined the user subjective input with an objective fitness, consisting of coded heuristics depending on the problem domain. Our results showed that the objective fitness component accounted for much of the increase in fitness performance [15], [16], [17]. However, in the absence of an objective fitness component, would the population converge to a local optima or would the user still be able to guide the population towards areas of high fitness? Building on our previous work, we show that by doing fitness interpolation based on the evaluation of the subset from a large population, and by asking for user input every t^{th} generation, $t > 1$, allows the user to move the population towards high fitness areas even without the use of an objective fitness. In addition, we show that our fitness interpolation technique can reduce the amount of feedback required by a user by tenfold when compared to a standard IGA with a small population size, and where the user evaluates the entire population every generation.

Our previous work focused on the design of simple user interface (UI) panels and floorplanning [1], [15], [16], [17]. In this paper we analyze techniques presented in our previous work on the Onemax problem of size 100. We follow along the lines of research that uses the Onemax problem for the empirical comparison of various techniques for IGAs [2], [10]. The challenge when conducting experiments with a subjective domain problem, such as with UI panels and in floorplanning, is being able to assess the fitness performance. Since solutions with our technique are evaluated based on similarity to the user selected best, solutions will always have a fitness value

Juan C. Quiroz, Sushil J. Louis, and Sergiu M. Dascalu are with the Evolutionary Computing Systems Lab, Department of Computer Science & Engineeering University of Nevada, Reno Reno, NV 89557, USA email: {quiroz, sushil, dascalus}@cse.unr.edu.

Amit Banerjee is with the School of Science, Engineering and Technology Pennsylvania State University, Harrisburg Middletown, PA 17057, USA email: aub25@psu.edu.

no greater than the fitness value of the user selected best. The Onemax problem allows us to explore and analyze how our interpolation techniques perform in a well known fitness landscape. In contrast, when dealing with a subjective domain problem, there is no way to know how close a solution is to the optimum because the landscape is defined by the user's subjective input, and there is no concept of a global optimum. In addition, the Onemax problem enables us to conduct repeatable experiments and to simulate user input, instead of having a user of a group of users interact with the IGA over a large period of time in order to obtain statistically significant results. In the experiments presented in this paper, similar to subjective domain problems, we are not interested in finding a single global optimal solution, rather in the ability of a user to bias the search towards areas of interest, in this case towards areas with an increasing number of ones.

The results in this paper include an analysis of our fitness interpolation techniques with noise introduced into the subjective evaluation provided by the simulated user input. Introducing noise in the Onemax problem is analogous to a user picking the solutions that would seem to contradict solutions picked by the user in earlier generations. This variation of user feedback is typical of actual users interacting with an IGA.

The rest of the paper is organized as follows. First, we present in Section II previous research on techniques to mitigate user fatigue and improve the user experience with IGAs. Section III presents a detailed description of the techniques for mitigating user fatigue in IGAs that were used in our study. In Section IV we present a discussion of the Onemax problem and its applicability to the testing of user fatigue mitigation techniques in IGAs. Section V presents the experimental setup for the Onemax case study. We discuss our results and analysis in Section VI. Finally, we conclude the paper in Section VII.

II. Related Work

Research on IGAs has been done on various hypothetical and real-world applications. Below we present a short survey of IGA work focused on mitigating user fatigue in IGAs, from the use of small populations to the use of machine learning algorithms to augment fitness evaluations.

A. User Fatigue in IGAs

Interactive genetic algorithms are a suitable tool for problems where *"there is no clear measure to give the evaluation of fitness other than the one in the human mind"* [3]. This applies to the evolution of UIs because users will be evolving UIs based on a mental model. Takagi identifies reducing human fatigue in the use of IGAs as the major remaining problem [19].

There are several fitness prediction techniques which try to abstract the user evaluation by building a model of the user, and using this model as a surrogate fitness function. Llorá et al. make the user pick the best solution from a small subset of the population displayed [10]. The displayed subset is a tournament used to define partial ordering of solutions; given that $s1$ and $s2$ are shown to the user, and the user picks $s1$, then we assume that the fitness of $s1$ is greater than the fitness of $s2$ [10]. The partial ordering of solutions, from the winners and losers of the tournaments, is used along with the dominance concepts of multi objective optimization to induce a complete ordering of solutions, which is subsequently used to train a support vector machine (SVM) to learn the user's preferences [4], [10].

A similar fitness prediction approach is presented by Watanabe et al. called PC-IGA: Interactive Genetic Algorithm based on Paired Comparison [20]. In PC-IGA the user evaluates binary tournaments as in the work by Llorá et al. The individuals in the population are randomly paired into tournaments. The winners of tournament brackets are subsequently paired into tournaments, until all the tournament brackets have been exhausted (all the way to the top of the tournament bracket). In contrast to the work by Llorá et al. the loser of each tournament bracket is replaced by the offspring of bracket winners. Watanabe et al. compare the number of generations between a typical IGA with the PC-IGA applied to hearing aid fitting. While they found users to take a similar number of generations with both approaches, the users provided more positive feedback to using PC-IGA, stating that it was easier to compare individuals than with the traditional IGA.

These tournament based approaches to fitness evaluation in IGAs are limited in that even when using a small population size, the number of tournaments that must be evaluated by the user is $L - 1$, where L is the population size. This is especially the case with the PC-IGA, in which the user must constantly evaluate binary tournaments, rather than evaluating the entire population via tournaments once and then offloading the fitness evaluation to a machine learning algorithm. In the work by Llorá et al., the approach is reasonable for small population sizes, for example, with a population size of 8, a total of 7 evaluations would be required to create the partial ordering of solutions. However, compared to our approach which uses a large population size of 100, a total of 99 evaluations would be required. Most importantly, the total population size along with their relative fitness values are used to train the SVM, thus with small population sizes only an SVM with a linear kernel would be successfully trained, whereas for more complex kernels, larger population sizes would be required.

Gu et al. present a fitness prediction approach for an IGA which uses a general regression neural network (GRNN) in order to predict individuals which match the user's aesthetic intentions [6]. The computational framework consists of two parts. First the user conducts artificial selection, by selecting individuals the user likes. After several generations, a pool of selected and unselected individuals is used to train the GRNN. The GRNN uses a probability distribution function. The computation framework is tested on the creation of a 3D cartoon face, where the chromosome of each individual encodes NURBS parameters, such as scaling and rotation factors. No formal user studies are conducted, with two users conducting the described experiments. One of the key design issues when using the GRNN is the choice of a sigma value. So a large portion of the results focuses on showing the effect of various sigma values on the resulting approximations of the measured cases from the interactive evolutionary process.

Henmi et al. present a fitness prediction method in which they learn and build a model of the user's evaluation behavior using feed-forward neural networks [7]. Once the model of the user has been built, it is used as a surrogate fitness function, allowing for the user model to take over evaluation, and present the user with potentially high fitness individuals. With other model based predictors in IGAs, the user evaluates the population in early generations, while the user model is learnt. In contrast, Henmi selects a user model built for another user in the early generations of the IGA, as the system learns and builds the current user's model. The model from another user which is the most similar to the current user's selections is picked while the system learns a model for the current user. The authors conclude that the proposed method is effective in early generations and can reduce user fatigue by accelerating the search process, especially when the current user's evaluation is similar to other users. The main limitation is that when user preferences vary in a problem domain, the use of other users' model can have a hindering effect.

The work presented in this paper does not attempt to do any user modeling with machine learning techniques, yet it is discussed as future work for this line of work. Instead, we use a simple interpolation based on the user selection of the best solution or the best and worst solutions in a limited subset to determine the fitness of every other individual in the population. Thus we reduce the user input to either one or two decisions every generation. Furthermore, we have the user evaluate a subset of the population every t^{th} generation, putting the user in a supervisory role and thus reducing the amount of feedback needed from the user. This is similar to the technique presented in [8]. We address how the value for t affects fitness performance in section VI. The work presented by Kamalian et al. also allows the user to give either a promote or demote reaction to individuals displayed for user evaluation [8] . In addition, they use a validity constraint to determine viable and meaningful designs to be displayed to the user. While individuals matching the validity constraint can be numerous, we explore the effects of displaying a small subset of the population for user evaluation and how the individuals selected as part of the subset affect the IGA's performance.

Lee and Cho introduce the concept of reducing user fatigue and improving performance of an IGA by allowing a user to evaluate a representative subset of the population [9]. Individuals from the population are divided into groups by using a clustering method such as k-means. The individual in the center of a cluster is selected as the representative of the cluster. The user evaluates the representative individuals and the rest of the cluster members are assigned a fitness based on the Euclidean distance from the representative member. The experiments conducted by Lee and Cho were done without a user in the loop, but the results presented show that by using a large population size, the fitness performance is increased substantially. We take a similar approach in this paper, testing how the composition of subsets taken from a large population affect fitness performance. One of the methods tested consists of a subset with the most diverse individuals from the population by doing a PCA analysis combined with k-means.

Dun-wei et al. presents a similar approach to Lee and Cho, but where the maximum number of clusters changes in time as the population converges[5].The work of Dun-wei et al. and Lee and Cho require the user to evaluate the representative individual from each cluster. We propose reducing user fatigue to a greater extent by asking the user to make only one evaluation every generation. In addition, we do not require the user to enter a numeric evaluation for the user, instead we simply ask the user to pick the solution the user likes best.

B. Collaboration in IGAs

In our previous work, we proposed the use of collaborative IGAs as a computational model of creative design [1], [14]. Our computational model most closely resembles Picbreeder, a system used to evolve pictures collaboratively online [18]. Picbreeder supports collaboration by allowing users to branch from images created by other users. Users can start evolution from randomly generated images, or from an image generated by someone else in the Picbreeder community. Our work differs from Picbreeder in that we support collaboration in real time. Users of our system participate in individual evolutionary sessions concurrently, with users able to see each others' designs as evolution progresses. Picbreeder is a special type of case injection, where users select the individual from which to start evolution, referred to in their research as "branching" [11]. In our model users work in a group setting, with users able to inject at any time any individual they find interesting from any of their peers.

In this paper we present an analysis of our computational model on the Onemax problem, assessing how the number of peers affects the IGA performance and how collaboration compares versus having a single user evaluate a larger subset size. We do not focus on the creativity aspect of the model, instead we only look at the overall fitness performance of introducing peer collaboration.

III. Techniques for Mitigating User Fatigue in Interactive Genetic Algorithms

To mitigate user fatigue we use fitness interpolation by asking the user evaluate the individuals from a subset selected from a large population size. Compared to previous approaches, where the user has to evaluate all individuals in a small population or all the individuals in a representative subset of the population, we ask the user to make two selections: the individual the user likes best and the individual the user likes least. We refer to these two user selections as user selected best and user selected worst. Asking the user to make two selections every generation further reduces user fatigue and improves the overall experience between the user and the IGA. This also allows the user to make rapid evaluations, since the user can concentrate on what he/she likes and does not like, and use that to quickly discard individuals which do not meet the user's evaluation criteria. As a consequence, the user would be compelled to participate in more generations than with a typical IGA.

A. Subset Display Method

We compared three methods of selecting individuals to be displayed to the user, initially proposed in our previous work [16], [17]. The three methods are displaying the best n individuals, displaying n random individuals, and displaying the best $n/2$ and the worst $n/2$ individuals in the population. Displaying the best individuals in the population gives the user the opportunity to view individuals that show the greatest potential by both meeting the objective and subjective heuristics most effectively. Displaying random individuals gives the user an unbiased insight into the current state of the population; it can allow the user to see the degree to which the population is converging (by the number of individuals that are similar), but it suffers because it can present bad solutions to the user. Displaying both best and worst individuals allows the user to see what the population is converging to and where it is coming from.

In addition, we test a subset method which runs a principal component (PCA) analysis on the chromosomes in the population. The work by Najafi and Beigy presents the use of PCA to extract test scenarios with the maximum diversity in order to improve the performance of evolutionary cellular automata [13]. Najafi and Beigy sort the test scenario instances according to their first PCA coefficient. We use the binary chromosomes as the input to the PCA analysis. The input dimension of the PCA analysis is set to the chromosome length. In the case of the Onemax problem of size 100, the input dimension is equal to 100. We set the output dimension of the PCA analysis equal to two-in the case of the Onemax problem of size 100, each chromosome is reduced to two principal components. We then run k-means on the PCA components, with the number of clusters equal to the size of the subset. This results in running the k-means algorithm in 2D space. Finally, we pick an individual from each cluster to be part of the subset.

In the work presented in this paper we are assuming that the IGA solves problems with chromosomes of uniform length, which is the case for the Onemax problem. In the case of varying length chromosomes, we propose two solutions. The chromosome with the minimum length and the chromosome with the maximum length are identified. The first solution truncates all of the chromosomes to match the length of the minimum length chromosome. The second solution pads all of the chromosomes with zeros to match the length of the maximum length chromosome. The first solution has the advantage that by reducing the input dimension to the length of the minimum length chromosome, we reduce the computational complexity of the PCA analysis. If we were using a standard GA, the computational complexity might not be as important, since there is not a user waiting to evaluate the next subset from the population. A disadvantage is that chromosomes truncated could be losing critical genes which account for the diversity of the chromosomes. For example, an individual could have a chromosome with 10 bits longer than the average chromosome length, with these 10 bits accounting for what differentiates the chromosome from the rest of the population. If the 10 bits were truncated off the tail of the chromosome, then such diversity would be lost and not taken into account by the PCA analysis. This could be mitigated to an extent by taking the most important segments of the chromosome, instead of simply truncating off the tail or head of the chromosome. The second solution (by padding with zeros) allows all chromosomes to be represented with all their genes, but this in turn increases the computational complexity of the PCA analysis. This has the disadvantage of increasing the time a user has to wait between evaluations.

Once the individuals that make up the subset from the large population have been identified, the subset is displayed for user evaluation, and feedback is accepted from the user. In the next subsection we address how the fitness of the individuals in the population is interpolated based on the user feedback.

B. Fitness Interpolation

An IGA session between the user and an IGA with our fitness interpolation technique begins by initializing a large population size. From this large population we select a small subset to be displayed to the user for evaluation. The choice of subset method is important, and we present and analyze the performance of various subset selection methods in section VI. If the fitness evaluation includes an objective evaluation, with a set of objective heuristics coded into the GA, then the initial population can be first evaluated with the objective heuristics, and then a subset is selected by taking into account the objective fitness scores of the individuals of the initial population. The user then makes the selection of picking the solution the user likes best (b), or picking the solution the user likes worst (w), or the best and worst, etc., from the individuals in the subset displayed. In this paper we compare only picking the best, versus picking both the best and worst. After accepting the user feedback, the fitness of every other individual in the population is then interpolated based on the similarity to the user selected individuals. A new population is created, with the choice of crossover, mutation, selection algorithm, etc., depending on the problem. We evaluate the new population and select a subset that is displayed to the user again for evaluation. This process repeats until the user has found one or more satisfactory solutions.

The similarity comparison can be done in either the genotypic space, the phenotypic space, or a combination of both. A similarity comparison in genotypic space can consist of the hamming distance or the longest common subsequence (LCS). A similarity comparison in phenotypic space will depend on the problem domain.

C. Purely Subjective Similarity Metric

Typical IGAs are driven by the user, where the fitness evaluation is replaced with user evaluation, without any further computation done by the fitness function. If the user makes a selection of the individual the user likes the best and the user likes the least, then the similarity between the best individual b and individual i and between the worst individual w and individual i in the population would be computed as follows:

$$b_s = (MH - hamming(b, i)) \tag{1}$$

$$w_s = hamming(w, i) \quad (2)$$

where the term MH is the maximum hamming distance. The final fitness value is the sum of b_s and w_s. This similarity metric assumes a representation where the hamming distance can be used. Alternatively, if the user were to only pick the individual the user likes the best, then the final fitness value would be set to the result of equation 1. While the purely subjective fitness evaluation shown here uses the hamming distance, other similarity metrics could have been used, such as LCS, or Euclidean distance. In the results we address performance of using the hamming distance versus LCS.

In our previous work we have used both hamming distance and Euclidean distance in order to evolve the layout and color of simple user interface (UI) panels that reflected the user's preferences while respecting UI style guidelines [15], [16], [17]. The hamming and the LCS distance similarity metric can be used when little to no domain knowledge is known. In the majority of IGA applications, the user evaluates the phenotype of individuals. In this case, the use of hamming distance or LCS results in a process whose goal is to move the population towards the genes contained by the user selected best individual. For example, in the case of evolving the color of a simple UI, if the user picks a UI with blueish tone as the best, then by approximating the fitness of every other individual in the population with hamming or LCS distance, the population moves towards genes which when expressed result in blueish tones.

Moreover, the similarity metric could also be applied on the phenotype space. In the case of UI panels, we used Euclidean distance to assign the fitness of a UI panel based on the Euclidean distance of the UI panel color vector and the color vector of the solution picked as the best by the user [15], [16], [17].

IV. Onemax

The onemax problem consists of maximizing the number of ones in a bit string of size l. It is one of the most basic problems used with GAs. The onemax problem has a single optimal solution, the string with l ones, and a simple objective fitness function can be used to determine the fitness value of solutions by just counting the number of ones in the string. In contrast, IGAs consist of problems that are subjectively evaluated, there is not a single optimal solution, and the stopping criteria depends on the user finding one or more solutions that the user is satisfied with. Which leads to the question, how does the use of the onemax problem benefit our study of user fatigue mitigation techniques for IGAs?

The onemax problem allows us to explore and analyze how our proposed interpolation techniques perform in a well known fitness landscape. In contrast, when dealing with a subjective domain problem, there is no way of knowing how close a solution is to the optimum because the landscape is defined by the user's subjective input, and there is no concept of a global optimum. Although, the optimal solution is ill-defined, a subjective IGA problem at any given point of time (generation) also has a fitness landscape similar to the Onemax problem. As the user picks his/her way through the landscape, the landscape changes. In other words, the Onemax problem has a constant global optima, a subjective IGA problem has one too every generation, consisting of the user selected best; however, the landscape is such that the global optima constantly changes. Moreover, our efforts are concentrated on guiding the search towards optimal regions of the landscape instead of the global optima.

The onemax problem can be mapped to the special case of a subjective IGA problem where a user has a strict set of requirements. For example, assume that a user is interested in exploring the space of dress designs. In this case, if the user was interested in finding a dress with a specific set of properties, such as having a navy blue color and a length at least past the knee, then every generation the user can guide evolution by picking the solutions which most closely meets the user's specific requirements. Similarly, in the onemax problem the desired property is a higher number of ones.

How do we reconcile the onemax having a single optimal solution and a well defined stopping criteria with an IGA problem without a single optimal solution and a stopping criteria defined by the user? In the experiments presented in this paper, we are not interested in finding the optimal solution consisting of a bit string with all ones. Instead, we are interested in showing the ability of a user to guide the evolutionary process towards high fitness solutions. Finding the optimal solution is not the goal, rather the ability of the user, through exploration, to be able to guide the search towards strings with an increasing number of ones as the generation increases. Furthermore, one of the primary stopping conditions of IGAs tends to be user fatigue and rapid convergence, as small population sizes is one of the most popular techniques used to reduce user fatigue in IGAs. Similarly, we use a stopping condition consisting of 20 user evaluations. That is, we simulate solutions being displayed to the user a total of 20 times, and the user providing a fitness feedback during evaluation.

While many IGA applications make use of multi-objective optimization, including our previous work on floorplanning [1], alternatives can be implemented. Instead of using multi-objective optimization, the criteria being optimized can be combined using a linear weighted sum, subsequently using a standard GA to optimize this linear weighted sum. This approach to solving a multi-objective optimization problem is intuitive and easy to use. However, it has several deficiencies. The most notable deficiency is finding a precise value of the weight for each objective. For example, in experiments conducted related to simple UI design we used weight values of 0.5 for the objective fitness and 0.5 for the subjective fitness [15], [16], [17]. These weight values could be varied to reflect various preferences, such as higher preference for the objective fitness value over the subjective fitness value or vice-versa. Other difficulties associated with converting a multi-objective problem into a single value optimization problem include that only one Pareto-optimal solution can be found in one simulation run of the algorithm, not all Pareto-optimal solutions can be found in nonconvex multi-objective optimization problems, and the algorithm requires the use of problem domain knowledge to find a suitable weight val-

ues [4]. Nevertheless, the onemax problem can be mapped to multi-objective optimization problems in IGAs which combine the optimization criteria using a linear weighted sum.

The onemax problem also enables us to conduct repeatable experiments. With subjective domain problems, as there is no global optimum solution, there is no way of assessing the performance or effectiveness of various IGA strategies. It is important to mitigate user fatigue, but it is just as critical to determine how a user fatigue mitigation technique affects the IGA performance. For example, a certain technique may reduce user fatigue by requiring less user input, but as a consequence may expose the user to fewer number of solutions, or may lead to premature convergence, hindering the ability of the user to explore potentially interesting solutions.

One of the key experiments presented in the results section revolves around introducing noise into the fitness evaluation. Introducing noise is analogous to a user picking the second or third best solution, from a subset displayed to the user, as the best solution for the current generation. This is meant to simulate variation in user feedback, as is typical in IGAs. In the case of dress designs, a user may be interested in bluish tones rather than a specific navy blue color and between successive interactions may change his preference from navy blue to ultramine although maintaining his overall predilection for blueish tones. By varying the degree of noise in the fitness evaluation, we can assess and compare a directed search versus a highly explorative search.

A. Subjective Evaluation of Onemax

We use the onemax problem to explore and gain further understanding of user fatigue mitigation techniques because we know its fitness landscape. In subjective domain problems and IGAs, where optimization is user driven, we cannot fully assess how some techniques affect the search, because the landscape and the definition of optimal and high fitness solutions will be user dependent.

Our results for the UI panel case study showed that the objective fitness accounted for much of the increase in fitness performance [15], [16], [17]. This was especially the case when we varied the frequency of user input, by asking for the user to evaluate the solutions in the subset every t^{th} generation. The average fitness in the population continued to increase steadily due to the objective criteria. One benefit we found of the objective heuristics, is that it allowed the user to serve a supervisory role, lessening the burden on the user having to make an evaluation every generation. The objective heuristics could fine tune solutions to meet a set of requirements or criteria, but the subjective input also provided a balancing criterion, which allowed the user to explore solutions which reflected the user's preferences as well as the coded objective criteria.

Based on the problem domain, and on the understanding of the problem and solution space, it might not always be possible to write a set of objective criteria which can be incorporated into an IGA to improve the subjective evolutionary search. Would a user be able to guide an IGA towards high fitness areas without an objective fitness value, incorporated via domain knowledge, resulting in a movement towards high fitness areas, or would the population get stuck on a local optima? Looking at the onemax problem of size l, we can hypothesize why a subjective evaluation based on a nearest-neighbor approach, such as hamming distance, might at first seem to fail to move the population towards high fitness areas without the pull of an objective fitness evaluation, as was the case in our examples of UIs [15], [16], [17].The expected fitness value of a randomly generated onemax string X of length l, where the random number generator follows a uniform distribution, is the following:

$$E(X) = \sum_{k=1}^{l} 0 * 0.5 + 1 * 0.5 = 0.5 * l \tag{3}$$

There are two possible outcomes when generating a bit, either a 1 or a 0. Using a uniform distribution, each outcome has a 0.5 probability, and this repeated over the length l of the string X gives $0.5 * l$. Consequently, the expected fitness value of an individual from the initial population will be equal to half the length of the individual's chromosome. If we pick a random subset from the initial population, this subset will consist mostly of solutions in the range of $l/2$ plus or minus some deviation, where l is the length of the chromosomes. Once the user makes a selection of the best individual from the subset, individuals whose number of ones in the chromosome is close to $l/2$ will get high fitness values, with individuals those with a high number of ones or zeros getting lower fitness values because of the hamming distance metric. Using a similarity metric based on LCS would yield a similar performance.

Regardless of the solution picked as the best by the user, the optimal solution, if already in the population or generated via crossover or mutation, will get a low score, with the individuals being in the median range getting the highest fitness values. The question we face is whether the population converges to a local optima, or if the proposed scheme allows a user to move the search towards areas of higher overall fitness (chromosome having increasingly higher number of ones). Building on our previous work presented in [15], [16], [17], we show that by doing fitness interpolation based on the evaluation of the subset from a large population, and by asking for user input every t^{th} generation, $t > 1$, allows the user to move the population towards high fitness areas.

We note that when using the PCA subset method for the Onemax problem, we do a naive PCA analysis, without the use of domain knowledge to improve the PCA analysis. For example, assuming a Onemax problem of size six, three possible individuals in the population could be x(1) = 010101, x(2) = 111000, and x(3) = 101111. The individuals x(1) and x(2) have the same number of ones, and hence they have the same fitness value. However, there is no guarantee that after the PCA analysis with k-means that these two individuals will belong to the same cluster, which would be desirable since both have the same fitness. On the other hand, we would want individual x(3) to be part of a different cluster, since x(3) has a different number of ones.

By using a naive approach we are assuming as little domain knowledge as possible. An approach that uses domain knowledge from the Onemax problem could arrange the bits in the chromosomes before doing the PCA analysis, such as grouping all ones at the beginning of the chromosome, or grouping all zeros at the beginning of the chromosome. This would make all chromosomes with an equal number of ones equivalent to the PCA analysis. We know that the number of ones is the only aspect of interest, and doing this would allow the PCA analysis with k-means to differentiate between chromosomes with a different number of ones.

B. Model of Collaboration in Interactive Genetic Algorithms

In our previous work, we proposed a computational model of creative design based on collaborative IGAs [1], [14]. The computational model is shown in figure 1. The figure illustrates three users collaborating with each other, with each of the peers denoted by the dotted boxes. Users interact with a GA by acting as subjective evaluators. The models shows an evaluation that is not purely subjective, instead the evaluation of design solutions consists of the multi-objective optimization of the subjective and objective criteria. We use Pareto optimality to maximize these criteria [4]. The arrows between the GAs of each of the peers represent the communication that takes place between them. If a user likes a solution from one of his/her peers, then the user has the option to inject that solution into his/her population, thus introducing a search bias.

In this paper we are interested in addressing strictly how the fitness performance of an IGA is affected in the presence of collaboration with peers. The user as well as the collaborating peers are simulated using virtual users, with a purely subjective evaluation.

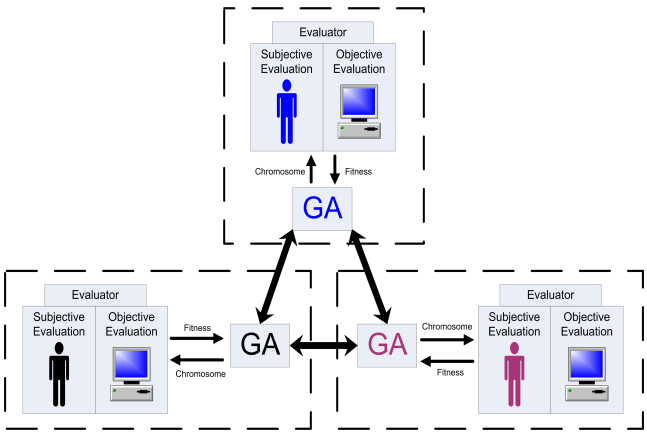


Fig. 1: Collaborative IGA Model

V. Experimental Setup

Experiments are conducted on the onemax problem of size 100, with the optimal solution being a string with 100 ones. However, as is the case in subjective problem domains, our goal is to show the ability of a user to guide the evolution towards areas of interest rather than finding a global optimum solution. In the case of the Onemax problem, areas of interest are solutions with an increasing number of ones. The IGA uses a large population size of 100, in contrast to a small population size, such as 10 or 20, used in standard IGAs [19]. The subset displayed for user evaluation is always of size nine. This subset size was determined empirically in our previous work and experiments [15], [16], [17], as too small a subset may not allow a user to view enough variety, whereas a large subset may overwhelm the user and can increase user fatigue. The effect of the subset size on the fitness performance of the IGA is an issue which we have not explored in our previous work, but which we address in the results presented in this paper.

Instead of having an actual user or a group of users conduct the experiments, we simulate user input. This gives us the leverage to conduct repeatable experiments and to obtain statistically significant results by running the IGA over 30 independent runs for each experiment. The simulated user input consists of always picking the solution from the subset with the most ones as the user selected best. In addition, we present results where we introduce noise into the simulated user pickings. This is analogous to the user picking the second best, third best, or n^{th} best solution from the subset as the user selected best, in order to make sure that the subset fitness interpolation works where there is not a clear solution, as tends to be the case in subjective domain problems.

The stopping condition for each independent IGA run is 20 user picks, that is, 20 generations. This is consistent with the number of generations typically completed by actual users interacting with IGAs [19]. From each subset displayed for user evaluation, we saved the individual picked by the user as the best. The user selected best was always the individual with the most ones, with the exception of the runs where we introduce noise. We base the performance of each of these algorithms on the subset constitution, since the user selections are all based on the individuals in the subset.

We are interested in determining through our experiments whether we can reduce user fatigue in IGAs with our fitness interpolation technique. In addition, we want to show that even in the absence of an objective fitness component, that a purely subjective fitness evaluation using our fitness interpolation technique allows a user to bias the population towards areas of high fitness. Finally, we want to assess and analyze how variations of our fitness interpolation technique affect the fitness performance of the IGA.

The fitness averages over the 30 independent runs of the resulting best individuals at the end of 20 generations are compiled in tables. Boxplots are used to analyze how widely fitness values vary across the 30 independent runs, and to establish statistical significance of the results. Looking at the variation will give us a better sense of how each of the methods performs in comparison to the others, and how the presented variations affect the performance.

We use box plots to analyze the 30 best individuals over the 30 independent runs for each of the methods. A box plot is a graphic representation of numerical data depicted through its five-number summary: minimum, lower quartile, median, upper quartile, and maximum. Furthermore, we use a variation of the box plot, a notched box plot, which in addition

shows notches around the median, the notches depicting the confidence intervals around the median [12]. If the notches around two medians do not overlap, then it can be said that the medians are statistically different at a 95% confidence level [12].

1) Subset Method and Step Size: Our experiments address two issues that affect convergence behavior of the IGA. First, should we show the user the top individuals in the population, a mixture of the best and worst individuals, a random set of individuals, or a set of the most diverse individuals from the population? Second, how often do we need to ask the user for feedback? And does lessening the frequency of user feedback actually improve fitness performance?

We'll be using the case where the user makes a selection of the best individual, using hamming distance as the similarity metric, as the base line for variations of our fitness interpolation technique. The performance of the base line case and variations of our fitness interpolation technique are further compared against a control group, consisting of a standard IGA with a small population size and where the user evaluates the entire population every generation. We compare the subset methods discussed herein: (1) displaying the top nine individuals; (2) displaying the best five and the worst four individuals; (3) displaying nine random individuals; and (4) displaying individuals selected from the PCA analysis with k-means. For each experimental run, we average fitness over the 30 independent runs of the IGA using each of the subset methods. The subset methods are compared against each other using step sizes of 1, 2, 5, 10, and 20. This is equivalent to asking the user to provide input every generation, every other generation, every five generations, every 10 generations, and every 20 generations respectively. Whenever we ask for user input every t^{th} generation, with $t > 1$, we continue to compute the subjective fitness evaluation, using the individual picked as the best by the user in the last user evaluation. For example, if $t = 20$, the user selects an individual as the best at generation zero. The IGA then uses the individual picked as the best at generation zero to evaluate the populations at generations one through 19. At generation 19, the user provides feedback, and the feedback is used to guide the evolutionary process for the next 20 generations.

2) Picking Best versus Picking Best and Worst: During evaluation, is it necessary to ask the user to select the solution the user likes best and the solution the user likes least, or is it sufficient to only ask the user to pick the solution the user likes best? The second experiment addresses this question. Our hypothesis is that asking the user to only pick the solution the user likes best yields similar fitness performance when asking the user to pick both the best and the worst solutions. This issue is important, as we can reduce the amount of user feedback further by only requiring one evaluation per subset every t generations.

3) Similarity Metric: While the similarity metric can be chosen based on a variety of factors, two methods which can be used for genotypic space comparison are hamming distance and LCS. The hamming distance can be computed in linear time, $O(l)$, where l is the length of the chromosome. LCS has a running time of $O(l_1 * l_2)$, where l_1 and l_2 are the lengths of the two chromosomes being compared respectively. In addition, LCS requires $O(l_1 * l_2)$ space complexity. In our previous experiments we had preferred the use of hamming distance over LCS because it noticeably increased the time it took to compute the fitness of the population, which in turn introduced a time lag between user pickings which affected the user experience [15], [16], [17]. In the experiments presented in the results section which test the performance of hamming distance versus LCS, our hypothesis is that LCS gives better fitness performance. For each experimental run, we average fitness over the 30 independent runs of the IGA using each of the four subset methods with step sizes of 1, 2, 5, 10 and 20.

4) Subset Size: How does the subset size affect the fitness performance of the IGA? We hypothesize that a larger subset size can improve the fitness performance, since a user would be able to provide feedback based on a larger sample of the population. However, a larger subset also increases the cognitive load on the user, and as a consequence can increase user fatigue faster than when using smaller subset sizes. The control group for this experiment is the use of a subset size of nine, which is the standard subset size used in the previously presented and the rest of the experiments conducted. We compare this against subset sizes of six individuals, 12 individuals, and 18 individuals. For each experimental run, we average fitness over the 30 independent runs of the IGA for each of the four subset methods with step sizes of 1, 2, 5, 10, and 20.

5) Noise: Solving the onemax problem with an IGA is a special case, where there is always a correct solution, the one with the most ones. This is certainly not always the case with IGAs in general. As has been pointed out in the discussion above, many factors influence the choices made by the user, among the most notable being user fatigue and the user changing the basis for evaluating individuals as the interactive session progresses. These factors contribute to an ever changing fitness landscape in a general IGA and we simulate it by introducing noise into candidate selection of the onemax problem. We also assess how fitness performance of our interpolation technique changes when we introduce noise into the fitness evaluation. Instead of displaying the actual number of ones contained in a string, we present a floating point number equal to the actual number of ones plus some Gaussian noise, with mean $\mu = 0$ and standard deviation $\sigma = 2$. We also ran the IGA with standard deviation values of $\sigma = 1$ and $\sigma = 5$, but we use the $\sigma = 2$ for comparison because it introduces a reasonable amount of noise (with $\sigma = 5$ the noise was quite large).

Introducing the Gaussian noise into the fitness evaluation is analogous to having the user pick the second best, or third best, etc., as the best solution from the individuals in the subset. The smaller the noise (the smaller the sigma value of the Gaussian noise), the more likely the virtual user is to pick one of the solutions from the subset with the highest fitness value. On the other hand, a larger Gaussian noise results in more erratic evaluations by the virtual user. For example, with a larger Gaussian noise the virtual user would be as equally likely to pick any individual in the subset as the best due to the introduced noise.

For each experimental run, we average fitness over the 30 independent runs of the IGA using each of the four subset methods with step sizes of 1, 2, 5, 10, and 20.

6) Collaboration: Previous work relating to our computational model of creative design has focused on assessing the creative quality of solutions. Here, we conduct a set of experiments in order to assess the performance of the collaborative computational model with regards to fitness performance. We address three issues regarding collaboration in the experiments. First, how does the number of peers affect the IGA fitness performance on the onemax problem? We compare the IGA fitness performance using collaboration with one, two, three, four, five and nine peers. Second, is there a benefit to displaying the solutions picked as the best from peers on the current user's screen? The experiment compares performance with solutions picked as the best from peers displayed on the current user's subset and without displaying the solutions picked as the best from peers. Third, what is the difference in fitness performance between working individually (without collaboration) with a large subset size of n versus working collaboratively viewing $n/2$ solutions from the user's own population and viewing $n/2$ solutions from peers. The experiment compares the use of a large subset size of 18 consisting of solutions from the user's own population, versus a subset consisting of nine solutions from the user's own population and nine solutions taken from peers.

For each experimental run, we average fitness over the 30 independent runs of the IGA using every subset method with step sizes of 1, 2, 5, 10, and 20.

VI. Results and Discussion

A. Subset Method and Step Size

Table I shows the average fitness value over 30 independent runs of the solution with the maximum fitness after 20 generations for each of the four subset methods - best, best-worst, random and PCA for varying step sizes of 1, 2, 5, 10 and 20.

From Table I we see that the fitness performance improves as we increase the step size, which is consistent with our previous results with the evolution of the UI panels. We notice that the performance does not increase linearly, in fact the performance decreases by a small fraction from step 10 to step 20, with the exception being the PCA subset method. Using a step size of either 5 or 10 provides the best results. As we compare the subset methods, we see that displaying the best individuals performs better than the other subset methods for steps 1 and 2. However, for steps 5 through 20, we see the random and PCA subset methods outperform the best and the best-worst subset methods. Using a step size greater than one allows the user to make a significant change in the large population. However, using too large of a step size can result in premature convergence to a local optima. We also see that with the higher step sizes, the random and PCA subset methods perform better. With the larger step sizes, the population converges faster, and hence the greater diversity from the PCA and subset methods improve the overall performance.

Figure 2 shows the performance of each of the subset methods: best, best-worst, random, and PCA. In addition, the performance of each of the subset methods is compared using a step size of 1, 2, 5, 10, and 20. In contrast to Table I, we can better appreciate the difference in performance of each subset method. In the box plot, the line at the top shows the maximum; the dotted line on the top of the box (or whisker) extends from the maximum individual to the upper quartile, shown as the top of the box; the median is the line inside the box, which depending on the data set may lie anywhere along the box; the lower quartile is shown as the bottom of the box; the bottom whisker extends from the lower quartile to the minimum individual, shown as a line; any outliers are depicted as points either above the maximum or below the minimum; and the notches around the median depict the 95% confidence intervals.

The first boxplot, labeled "Control", shows the fitness performance of the control case, an IGA with a population size of 10 with a stopping condition of 20 generations. The control case reflects the performance of a typical IGA with a small population size and where the user evaluates the entire small population every generation. From the figure we can see that the control IGA performs almost identically to the case where the user picks the best from a subset consisting of nine best individuals from a population of size 100. However, the figure shows the positive effect in the fitness performance by increasing the step size. Most importantly, our fitness interpolation technique using the subset method consisting of the best individuals from the population and using a step size of 1 performs the same as the control group, but with a total of only 20 evaluations by the user, i.e. only 20 user picks. In contrast, the control group achieves the same level of performance but by having the user provide feedback about 10 individuals for 20 generations for a combined total of 200 user picks. With our fitness interpolation technique, using the lowest possible step size, we can reduce the amount of user feedback by tenfold. Further, using a large population size reduces genetic drift, reduces the likelihood of premature convergence, and allows the user to explore a greater diversity of individuals.

When comparing the various subset methods, using a step size of 1 and displaying the best individuals from the population to the user for evaluation performs the best. However, by observing the overlap of the notches, it can be inferred that the medians of the best, best-worst, and PCA methods are not statistically different within the 95% confidence interval. The medians of the best and best-worst are significantly different than those of the random method, but the PCA method does have a small degree of overlap with the random method.

When using a step size of 2, the best subset method outperforms the random and PCA methods, but the median

TABLE I: Average Maximum Fitness for Different Combinations of Subset Methods and Step Sizes

Step	Best	Best Worst	Random	PCA
1	70.00	68.70	65.86	67.40
2	73.43	72.26	71.96	70.93
5	76.26	75.50	77.20	77.06
10	76.23	76.66	78.73	78.63
20	74.60	76.53	78.60	79.00

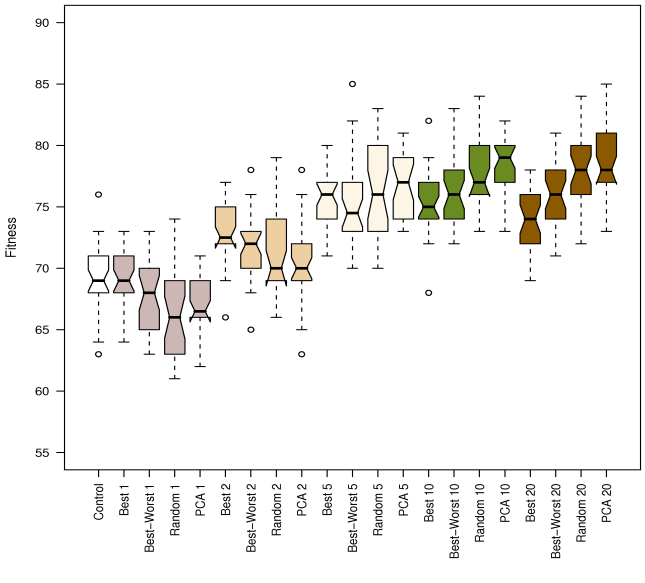


Fig. 2: Comparison of the various subset methods. The first boxplot shows the control group. The rest of the boxplots show the four subset methods using a step size of 1, 2, 5, 10, and 20.

confidence interval of the best method overlaps with that of the best-worst method. The boxplots for a step size of 2 include some odd ends on the bottom notches of the boxplot for the best and random methods. This depicts that the confidence interval lies outside of the lower quartile, which can occur if the median is close to the lower quartile and if the sample sizes are small.

The performance of the best and best-worst methods begin to degrade when using a step size of 5, when compared to the random and PCA methods, as we continue to increase the step size. Nevertheless, the confidence intervals around the medians of the best, random, and PCA methods overlap. The only statistically significant difference is between the medians of the PCA and the best-worst methods.

Picking the most diverse individuals from the population through PCA yields the best performance when using large step sizes of 10 and 20, as seen in Figure 2. With a step size of 10, the PCA method shows a statistically different median than the best, random, and best-worst methods. When using a step size of 20, the medians of the PCA and the random methods are nearly identical, with both of these methods being clearly superior to the best method, and the random and PCA methods having a fraction of overlap with the confidence interval around the median of the best-worst method.

While the tables shown present average of the best individual in the subset displayed for evaluation, it does not show how the fitness performance progresses on a step basis. Figure 3 shows how the four subset methods compare when using a step size of 1 versus using a step size of 5. From the results presented above, it can be seen that IGA with $t = 5$ performed as well as $t = 10$ and $t = 20$, while still allowing a user to intervene in the evolutionary process in an efficient manner.

The graph shows a considerable fitness improvement when using the step size of 5 compared to interacting the user every generation (step size of 1). Displaying the best individuals from the population for user evaluation provides a superior fitness performance over the three other subset methods when using a step size of 1. When using a larger step size, displaying a diverse set of individuals in the subset for user evaluation yields better fitness performance, as exemplified by both the random and PCA subset methods yielding the highest fitness performance.

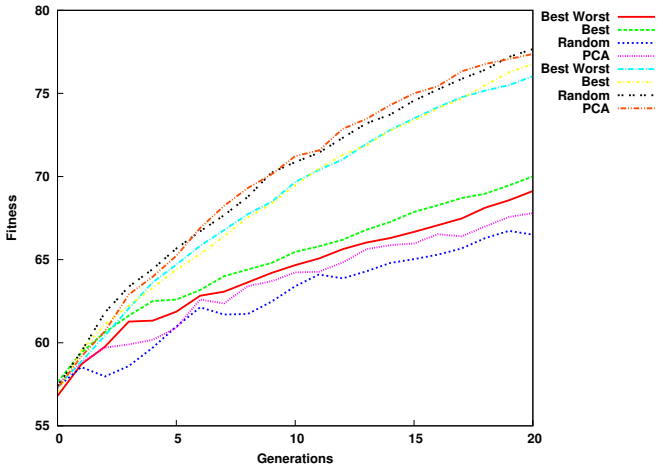


Fig. 3: Average Maximum Fitness Vs Generation: The top four line plots show the fitness performance of the subset methods for $t = 5$, while the bottom four line plots show the subset methods for $t = 1$.

B. Picking Best versus Picking Best and Worst

Next we analyzed the performance of picking the best individual versus picking both the best and worst individuals from the subset displayed to the user for evaluation. Our hypothesis was that asking the user to pick the best would yield the same performance as asking the user to pick the best and worst. Table II shows the performance of these two user evaluation alternatives. We see that for a step size of 1, the performance of the two evaluation alternatives is similar, with the best and worst user picking resulting in slightly higher fitness scores. With a step size of 2, the performance of the best and worst user picking gains a greater edge, however, the performance of the two evaluation methods is comparable across all subset methods. Furthermore, on step sizes of 5 through 20, we see that the picking of the best individual shows increasing better performance as the step size increases. This shows that asking the user to pick the best individual from a subset, instead of asking the user to pick both the best and worst individuals from the subset, allows us to reduce the amount of user feedback every generation, while not suffering from a loss in fitness performance.

C. Similarity Metric

Table III shows the result of the hamming versus LCS method comparison, where the user picking was simulated by picking only the best individual from the subset displayed for

TABLE II: Average Maximum Fitness for Different Combinations of Subset Methods, Step Sizes and Picking Variants

Step	Pick Variant	Best	Best Worst	Random	PCA
1	Best	70.00	68.70	65.86	67.40
1	Best Worst	71.33	69.36	67.36	67.90
2	Best	73.43	72.26	71.96	70.93
2	Best Worst	75.83	73.70	73.06	72.46
5	Best	76.26	75.50	77.20	77.06
5	Best Worst	76.26	74.93	76.13	75.20
10	Best	76.23	76.66	78.73	78.63
10	Best Worst	74.43	73.06	75.30	72.33
20	Best	74.60	76.53	78.60	79.00
20	Best Worst	70.00	69.40	69.36	69.60

TABLE III: Average Maximum Fitness for Different Combinations of Subset Methods, Step Sizes and Similarity Measures (Pick Variant: Best Only)

Step	Similarity Method	Best	Best Worst	Random	PCA
1	Hamming	70.00	68.70	65.86	67.40
1	LCS	70.06	69.33	67.20	67.96
2	Hamming	73.43	72.26	71.96	70.93
2	LCS	74.16	73.40	71.80	73.03
5	Hamming	76.26	75.50	77.20	77.06
5	LCS	75.36	76.16	76.86	77.70
10	Hamming	76.23	76.66	78.73	78.63
10	LCS	74.50	76.80	79.66	79.16
20	Hamming	74.60	76.53	78.60	79.00
20	LCS	74.40	77.40	79.80	79.53

evaluation. We see that both the hamming distance and the LCS perform competitively, with LCS giving a slightly higher fitness score in the majority of cases, but never being of a significant difference. From this results we can conclude that we can use the hamming distance as similarity metric, either when we have no further domain knowledge which can be used to define a stronger similarity metric or when the simplicity of the hamming distance is sufficient to compare two genomes in the genotypic space. Finally, we can compute the similarity between any two chromosomes in linear time $O(l)$ in contrast to the $O(l_1 * l_2)$ running time and space complexity of LCS.

D. Subset Size

Figure 4a shows a box plot of step sizes compared up to step size five, similar to figure 2, but with each pair denoting the subset size of 9 with the first box and the subset size of 6 with the second box. By comparing every pair of boxes in sequential order we can see how reducing the subset size from nine to six affects the fitness performance. We can see that as we had hypothesized, reducing the subset size decreased the fitness performance. The difference in fitness performance between using a subset size of nine versus six grows as we increase the step size.

Figures 4b and 4c shows the performance comparison of using a subset size of nine versus a subset size of 12 and 18. We used factors of three, because that would still allow us to display the subset on the screen a grid of three individuals per row, or alternatively using a grid of six individuals per row. From figures 4b and 4c, we can see that as the subset size is increased, the fitness performance increases substantially. On figure 4c we see how the last pairs of boxes, using a step size of five, show a big gap in fitness performance from doubling the subset size from nine to 18. While the shown increase in fitness performance is highly desirable, it is also accompanied by asking the user to evaluate twice as many individuals at every interaction. This can result in the user stopping the evolutionary process after a few generations, leading to suboptimal solutions.

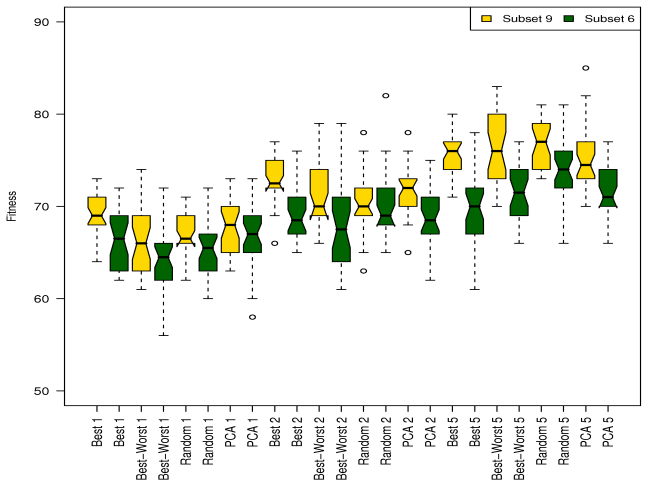
E. Noise

Introducing the Gaussian noise into the fitness evaluation is analogous to having the user pick the second best, or third best, etc., as the best solution from the individuals in the subset. The smaller the noise (the smaller the sigma value of the Gaussian noise), the more likely the virtual user is to pick one of the solutions from the subset with the highest fitness value. On the other hand, a larger Gaussian noise results in more erratic evaluations by the virtual user.

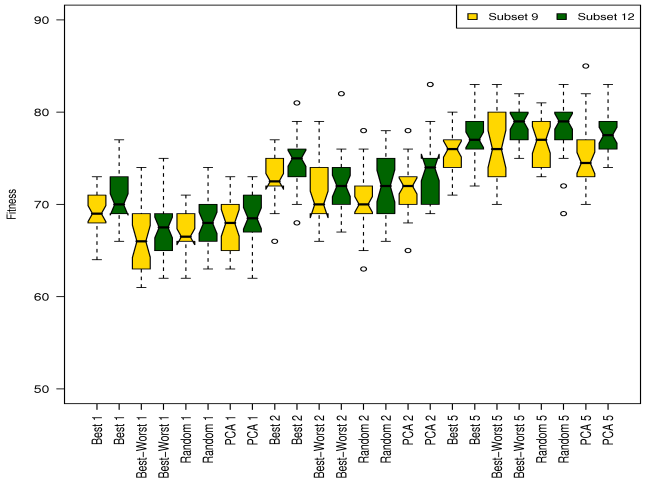
How does the amount of noise affect the fitness performance? Table IV compares the performance of the IGA without noise and with Gaussian noise with sigma values of $\sigma = 1, 2, 5$, from which we can see how increasing the Gaussian noise affects the IGA performance for each step size. The noise of $\sigma = 1$ has a small effect on the fitness performance, while the noise of $\sigma = 5$ has a significant effect on the fitness performance, especially for larger step sizes. It has already been shown that a larger step size improves fitness performance. However, the use of the larger step size does not help when there is a large degree of noise in the IGA evaluation. In fact, for all step sizes, 1 through 20, and with $\sigma = 5$ we see the fitness performance converging to values between the mid to low 60s. Given that the starting expected fitness value of the individuals in the population for the Onemax problem of size 100 is 50, a score in the low 60s means that the algorithm has difficulty extrapolating in the presence of large amount of noise. This would be analogous to a user making inconsistent decisions while providing user feedback. For example, if a user was evaluating a color scheme for a UI, the user could show a tendency for colors with a blue tone. However, it is still reasonable for the user to pick blue tones for a few generations, and then to decide to start picking UIs with a yellow tone. While enforcing consistency from the user lessens the complexity of the problem solving by the IGA, it can also inhibit the exploratory nature of the IGA. A user change is not necessarily something negative, since one of the key features which makes IGAs powerful is their ability to allow users to explore many solutions quickly.

Figure 5 shows how the fitness performance is affected with various Gaussian noise values. The box plots from figure 5 show pairwise comparison of the IGA without noise and with Gaussian noise for each of the four subset methods. The first eight box plots are for the IGA using a step size of one, $t = 1$, the following eight box plots are for the IGA using a step size of two, $t = 2$, and the last eight box plots are for the IGA using a step size of five, $t = 5$. We only show the box plots up to $t = 5$ because for the larger values of t, we saw similar fitness performance to $t = 5$.

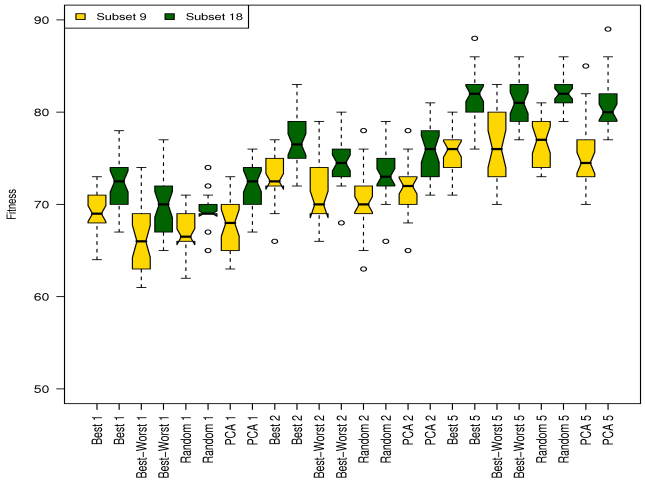
The boxplots on figure 5 are paired by color, wherein within each colored pair the left boxplot represents the IGA with no



(a) Subset 9 vs Subset 6



(b) Subset 9 vs Subset 12



(c) Subset 9 vs Subset 18

Fig. 4: Fitness performance comparison of using a subset size of 9 solutions displayed to the user for evaluation versus subset sizes of 6, 12, and 18.

TABLE IV: Fitness Degradation due to Noise

Step	Gaussian Noise - Sigma	Best	Best Worst	Random	PCA
1	No noise	70.00	68.70	65.86	67.40
1	1	68.13	67.4	66.23	66.17
1	2	63.77	66.93	66.27	65.77
1	5	61.07	63.3	61.5	64.03
2	No noise	73.43	72.26	71.96	70.93
2	1	70.87	70.77	70.2	69.9
2	2	65.83	68.17	66.6	68.47
2	5	59.87	64.37	64.9	64.4
5	No noise	76.26	75.50	77.20	77.06
5	1	70.9	73.6	75.27	76.23
5	2	66.97	70.23	72.17	72.33
5	5	60.43	64.5	64.77	65.23
10	No noise	76.23	76.66	78.73	78.63
10	1	70.5	74.83	76.27	76.37
10	2	64.6	69.93	72.53	72.73
10	5	59.63	62.63	61.77	64.2
20	No noise	74.60	76.53	78.60	79.00
20	1	69.53	72.1	76.23	76.6
20	2	64.43	69.9	70.03	71.87
20	5	59.6	62.67	63.6	63.67

noise and the right boxplot represents the IGA with noise. When using a Gaussian noise with $\sigma = 1$, as shown in figure 5a, the fitness performance of the IGA with noise shows a fitness performance comparable with the IGA with no noise. The performance is comparable across all subset methods as we increase the step size t, with the exception of the subset method that displays the best individuals from the population. The subset method that displays the best individuals from the population is sensitive to noise because of its lack of diversity. If a user were to pick the second or third best individual from the subset as the best, then during the next evaluation the subset would consist of individuals similar to the second or third best, with no considerable progress in the fitness of the population. Whereas a more diverse subset method, such as random or PCA, would allow the user to potentially correct a previous erroneous or inconsistent evaluation.

When using a Gaussian noise with $\sigma = 2$, figure 5b, the fitness performance degrades further than when using $\sigma = 1$ as expected. However, even when using this larger Gaussian noise, the fitness performance continues to improve when using larger step sizes as in the case with the IGA with no noise. This shows that a user is able to guide the evolution towards high fitness solutions with a reasonable amount of noise. With the Gaussian noise of $\sigma = 1$, the degradation in fitness performance of the best subset method was statistically significant for $t > 1$, since for $t > 1$ the confidence intervals of the IGA without noise and with $\sigma = 1$ did not overlap. With the larger Gaussian noise of $\sigma = 2$ the confidence intervals on the box plots are spread farther apart, as is especially obvious for $t = 5$, where the degradation in fitness performance is statistically significant for all subset methods.

Figure 5c shows the fitness performance when using a Gaussian noise of $\sigma = 5$. As the noise in the fitness evaluation continues to increase, the fitness performance ends up averaging in the low to mid 60s. With larger noise, the use of a larger step size t yields no improvement on fitness performance. Further, regardless of the value of t, all of the subset methods perform similarly. Figure 6 shows the

comparison of the IGA with Gaussian noise with $\sigma = 1$ versus $\sigma = 5$. From this we can conclude that the proposed interpolation techniques are robust with a reasonable amount of noise in the fitness evaluation. A user who provides poor and inconsistent evaluations introduces noise into the fitness landscape, resulting in the IGA converging to suboptimal solutions.

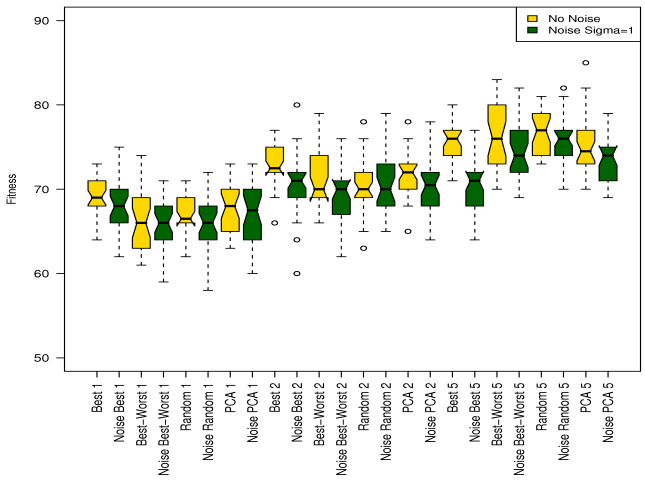
F. Collaboration

We ask how the number of peers affects the IGA fitness performance on the onemax problem. Figure 7 shows boxplot for the fitness performance, with the IGA running with one peer, two peers, three peers, four peers, five peers, and nine peers. We can see that increasing the number of peers does improve the fitness performance of the IGA. However, the increase in fitness performance is moderate.

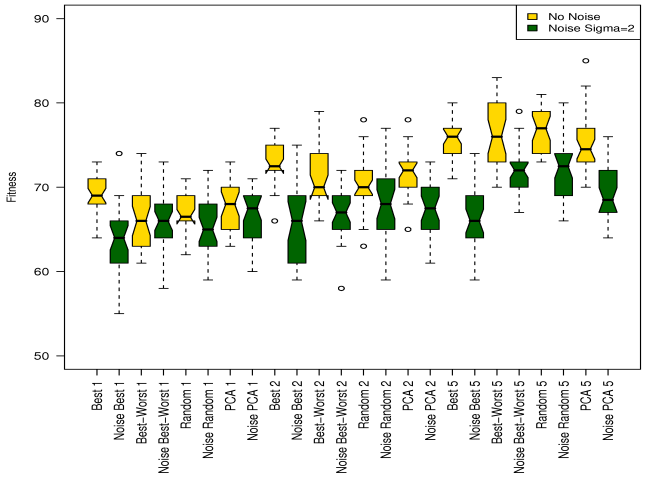
In the computational model of creative design, the individuals selected as the best from peers are always displayed on the peer panel of the all the collaborators [1]. However, in this study we analyze the relative importance of enforce this requirement. Figure 8 shows the fitness performance of the four subset methods when not displaying the individuals selected as the best from peers, and when displaying the individuals selected as the best from peers. We can see that the performance of the random and the PCA subset methods are the ones that improve the most. Overall, we can see the four subset methods are evened out in terms of fitness performance. By displaying the individuals selected as the best from peers, the PCA and random subset methods improve in performance, as these methods gain the benefits of displaying the individuals with the highest fitness.

How does increasing the number of peers affects the fitness performance? Figure 9 shows the fitness performance of using an IGA with a large subset size, with 18 individuals in the subset, versus displaying nine individuals from the user's population and displaying nine individuals belonging to peers. From the boxplots we can see that using a larger subset size yields similar fitness performance to collaborating with peers while using an IGA. As we increase the number of peers, the fitness performance improves. When allowing for collaboration amongst a large number of peers, the chances of one user finding a high fitness solution increases, since the user can inject a solution from one of the many peers into his/her own population. The collaborative process can be thought of as parallel IGAs, and taking the maximum of all individuals selected as the best by users, and using this maximum as the user selected best of all users.

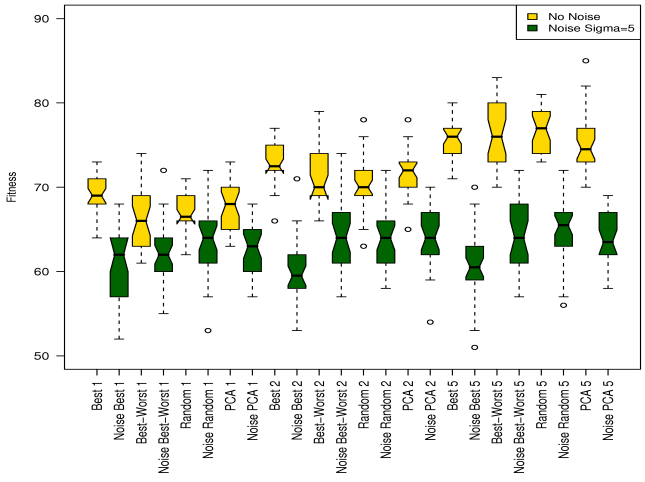
While the results show similar fitness performance for using collaboration and having using a larger subset, this does not imply that these two methods are equivalent and would yield similar results. The collaborative IGA model was introduced as a computational model of creative design, and the results presented here do not reflect any assessment of creativity. The results also do not reflect on the degree of diversity in the corresponding populations of the two methods. From our empirical observations we have seen that when using the collaborative model with subjective problems, the population



(a) No Noise vs Gauss $\sigma = 1$



(b) No Noise vs Gauss $\sigma = 2$



(c) No Noise vs Gauss $\sigma = 5$

Fig. 5: Fitness performance comparison of various subset methods without Gaussian noise in the user evaluation versus Gaussian noise with $\sigma = 1, 2, 5$.

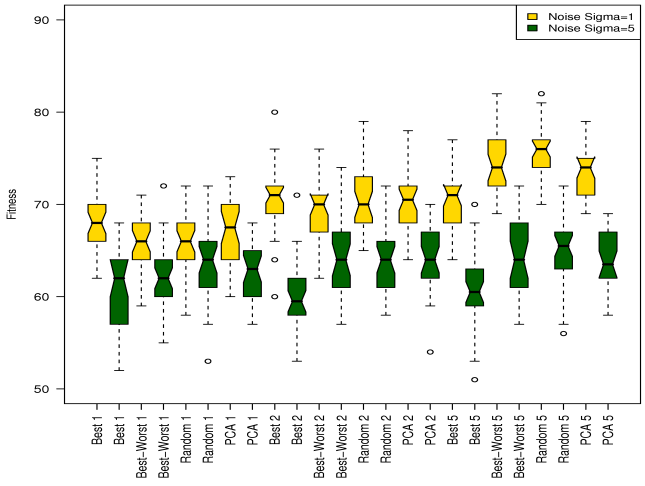


Fig. 6: Fitness performance comparison for various subset methods with Gaussian noise of varying magnitude: $\sigma = 1$ vs $\sigma = 5$

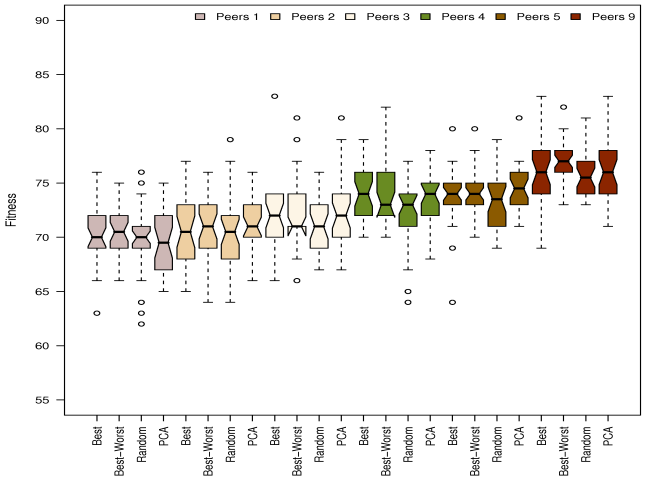


Fig. 7: Fitness performance comparison of different subset methods with varying number of peers

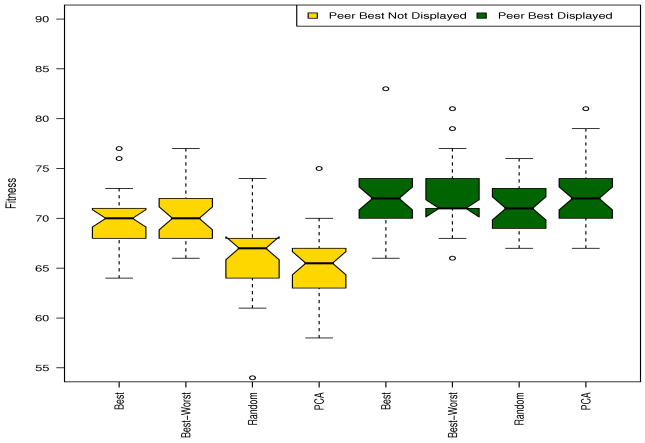


Fig. 8: Fitness performance comparison of the subset methods with and without displaying peer best solutions

contains a diverse number of high fitness individuals, whereas individual IGA sessions usually converge to a single solution.

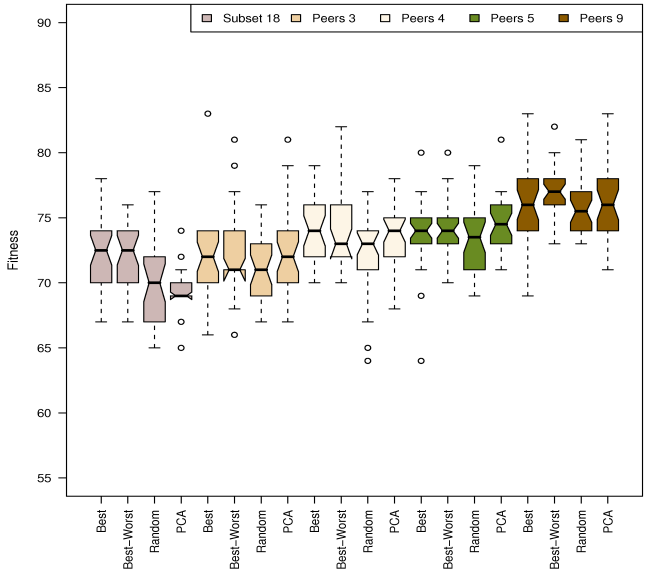


Fig. 9: Fitness performance comparison of large subset size made up of varying degree of collaborative solutions

VII. Conclusion

User fatigue in IGAs can be detrimental to the user experience. Traditional IGAs rely on the use of small population sizes and a small number of generations in order to mitigate user fatigue. Consequently, this leads to limited exploration of the search space. In addition, even if using a small population size, evaluating individuals from the population over several generations can result in the user losing interest and stopping before finding a satisfactory solution.

Our fitness interpolation technique demonstrated how to reduce user fatigue while effectively making use of the user feedback. In this technique, a small subset selected from a large population size is displayed to the user for evaluation every t generations. The user selects the solution the user likes best from the subset displayed for evaluation. By requiring only a single evaluation every t generations, we can reduce the amount of user evaluations. The use of a large population size reduces genetic drift, reduces the likelihood of premature convergence, and allows the user to explore a greater diversity of individuals. It was found using our technique we can reduce the amount of user feedback by tenfold compared to a standard IGA using a small population size. In addition, the evolutionary search can be effectively guided towards areas of interest while evaluating only subsets of the population and without an objective fitness component.

While the focus of this work is on the onemax problem, the similarity and mapping between the onemax problem and to subjective domain problems has been established. Thus, the use of the onemax problem is a promising direction of research which can significantly contribute to the empirical testing of user fatigue mitigation techniques in interactive evolutionary computation.

VIII. Acknowledgments

This work was supported in part by contract number N00014-05-1-0709 from the Office of Naval Research and the National Science Foundation under Grant no. 0447416.

References

[1] A. Banerjee, J. C. Quiroz, and S. J. Louis. *A Model of Creative Design Using Collaborative Interactive Genetic Algorithms*, pages 397–416. Springer, 2008.

[2] C. D. Cheng and A. Kosorukoff. Interactive One-Max problem allows to compare the performance of interactive and Human-Based genetic algorithms. In *Genetic and Evolutionary Computation GECCO 2004*, pages 983–993. 2004.

[3] Cho. Towards creative evolutionary systems with interactive genetic algorithm. *Applied Intelligence*, 16:129–138, Mar. 2002.

[4] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley and Sons, 2001.

[5] D. Gong, J. Yuan, and X. Ma. Interactive genetic algorithms with large population size. In *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 1678–1685, 2008.

[6] Z. Gu, M. X. Tang, and J. H. Frazer. Capturing aesthetic intention during interactive evolution. *Computer-Aided Design*, 38(3):224–237, Mar. 2006.

[7] S. Henmi, S. Iwashita, and H. Takagi. Interactive evolutionary computation with evaluation characteristics of Multi-IEC users. In *Systems, Man and Cybernetics, 2006. SMC '06. IEEE International Conference on*, volume 4, pages 3475–3480, 2006.

[8] R. Kamalian, Y. Zhang, H. Takagi, and A. Agogino. Reduced human fatigue interactive evolutionary computation for micromachine design. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 9, pages 5666–5671 Vol. 9, 2005.

[9] J. Lee and S. Cho. Sparse fitness evaluation for reducing user burden in interactive genetic algorithm. In *Fuzzy Systems Conference Proceedings, 1999. FUZZ-IEEE '99. 1999 IEEE International*, volume 2, pages 998–1003, 1999.

[10] X. Llorà, K. Sastry, D. E. Goldberg, A. Gupta, and L. Lakshmi. Combating user fatigue in igas: partial ordering, support vector machines, and synthetic fitness. pages 1363–1370, Washington DC, USA, 2005. ACM.

[11] S. Louis and C. Miles. Playing to learn: case-injected genetic algorithms for learning to play computer games. *Evolutionary Computation, IEEE Transactions on*, 9:669–681, 2005.

[12] R. McGill, J. W. Tukey, and W. A. Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, Feb. 1978.

[13] M. Najafi and H. Beigy. Using PCA to improve evolutionary cellular automata algorithms. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1129–1130, Atlanta, GA, USA, 2008. ACM.

[14] J. Quiroz, S. Louis, A. Banerjee, and S. Dascalu. Towards creative design using collaborative interactive genetic algorithms. In *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, pages 1849–1856, 2009.

[15] J. Quiroz, S. Louis, A. Shankar, and S. Dascalu. Interactive genetic algorithms for user interface design. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 1366–1373, 2007.

[16] J. C. Quiroz, S. M. Dascalu, and S. J. Louis. Human guided evolution of xul user interfaces. pages 2621–2626, San Jose, CA, USA, 2007. ACM Press.

[17] J. C. Quiroz, S. J. Louis, and S. M. Dascalu. Interactive evolution of xul user interfaces. pages 2151–2158, London, England, 2007. ACM Press.

[18] J. Secretan, N. Beato, D. B. D. Ambrosio, A. Rodriguez, A. Campbell, and K. O. Stanley. Picbreeder: evolving pictures collaboratively online. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1759–1768, Florence, Italy, 2008. ACM.

[19] H. Takagi. Interactive evolutionary computation: fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 89:1275–1296, 2001.

[20] Y. Watanabe, T. Yoshikawa, and T. Furuhashi. A study on application of fitness inference method to PC-IGA. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 1450–1455, 2007.