Interactive Genetic Algorithms for User Interface Design

Juan C. Quiroz, Anil Shankar, Sushil J. Louis, Sergiu M. Dascalu

***Abstract*— We attack the problem of user fatigue in developing an interactive genetic algorithm that helps design user interfaces. The interactive genetic algorithm combines computable user interface design metrics with subjective user input to guide evolution. Individuals in our population represent interface specifications and we compute an individual's fitness from a weighted combination of user input and user-interface-design guidelines. Result from our preliminary study involving three users indicates that users are able to effectively bias evolution towards user interface designs that reflect both user preferences and computed guideline metrics. Furthermore, we can reduce fatigue, defined by the number of choices needing to be made by the human designer, by doing two things. First, asking the user to pick just two (the best and worst) user interfaces from among a subset of nine shown. Second, asking the user to make the choice once every t generations, instead of every single generation. Our goal is to provide interface designers with an interactive tool that can be used to explore innovation and creativity in the design space of user interfaces and make it easier for end-users to further customize their interface without programming knowledge.**

I. Introduction

User interface (UI) design is an expensive, complex, and time consuming process usually driven by documented style guidelines and design principles. However many of these guidelines and design principles are difficult to translate into code and good UI design is driven in large part by human aesthetics in their look and feel. Furthermore, Thimbleby points out that it is difficult to generalizes beyond specific case studies [15]. Thus UI designers tend to be guided both by objective measures gleaned from UI style guidelines and design principles, and by subjective measures such as the "look" and "feel" of an interface. Our interactive genetic algorithm combines both measures in its fitness function and allows the UI designer to simply and efficiently explore the space of UI designs.

Interactive genetic algorithms (IGAs) differ from GAs in that objective fitness evaluation is replaced with user evaluation. As such, they can incorporate intuition, emotion, and domain knowledge from the user. However, GAs usually rely on the use of large population sizes running for hundreds of generations to achieve satisfactory results [8]. Such computational dedication cannot be expected from the user due to psychological and physical fatigue. Thus, although the use of IGAs presents us with a powerful tool with which to incorporate subjective evaluation into the GA process, how best and effectively to incorporate user input remains a significant research challenge [14].

The authors are with the Evolutionary Computing Systems Lab, Department of Computer Science and Engineering, University of Nevada, Reno, NV 89557, USA. Email: {quiroz, anilk, sushil, dascalus}@cse.unr.edu

Our work differs in that we use both a computable fitness criterion and user evaluation to compose a combined fitness. We encode user interfaces as individuals in an IGA, and run over a number of generations to help explore the space of UI designs. Periodically, the UI designer sees the phenotypes (the UIs) corresponding to nine of the best individuals in the population and picks two - the best and worst looking interfaces. Empirical observations tell us that we should not display more than nine or ten items to be judged by a user [11]. In addition, users will tire and become erratic in their choices if we ask them to make hundreds of decisions. In this paper we reduce user fatigue by asking the user to pick two individuals from nine shown, every t generations of the IGA. Preliminary user studies show that users are still able to effectively bias evolution towards designs that reflect their preferences as well as the bias imparted by the computable interface design metrics.

The rest of the paper is organized as follows. We discuss IGA's, the challenges of UI design and related work in the next section. Section III presents the encoding used for individuals in the population. The next two sections discuss how we do subjective and objective evaluations and our experimental setup. Section VI describes our results with test subjects using our IGA to evolve a simple UI. Finally, in section VII we present our conclusions and directions for future work.

II. Background

In the following subsections, we describe IGAs, the inherent challenges of UI design, work related to incorporating user input into IGAs, and the use of evolutionary techniques for UI design. We also explain our choice of XUL as the target language for our UIs.

A. Interactive Genetic Algorithms (IGAs)

IGAs provide a mapping from a user's psychological space to a GA's parameter space and thereby combine the power of human subjective evaluation with evolutionary computation [14]. We can thus incorporate human preferences, knowledge, emotion, and intuition into GAs. A GA computes an individual's fitness using a mathematical equation, some computation, or a model [8]. However, we cannot model users trivially; users preferences are relative and context-dependent. In interactive genetic algorithms the user participates in fitness evaluation - in many cases the user **is** the fitness function. Generally a user assigns fitness to an individual in the following ways: (1) assigning a number on a subjective grading scale, (2) ranking the individual, or (3) choosing the best individual from a displayed subset [8], [14]. A wide variety of applications that depend on creative human input like editorial design, industrial design, image processing, database retrieval, graphic art, computer graphics animation, control and robotics amongst other are ideal candidates for an IGA-based approach due to the user-centric nature of IGAs [14].

Note that for an IGA-based user centric application, we do not have the luxury of relying on large populations sizes and hundreds of generations (like a canonical GA) because it is unrealistic to request a user to make hundreds or thousands of choices. If pressed for too much feedback, users are likely to lose interest and get tired. Llora, Sastry, Goldberg et al. have shown that the subjective nature of human input can lead to users changing their task goals through an IGA run [8]. Such user behavior results in noisy landscapes - which coupled with user fatigue can lead to suboptimal solutions.

Because of the subjective and creative nature of UI design, we believe that an interactive genetic algorithm is a viable tool for UI design. We briefly describe UI design and its challenges in the next section.

B. User Interface Design

Designing interactive systems that are easy to use, engaging, and accessible to all users is a challenging task. User interface design is a complex process critical to the success of any software system. The design of a user interface is a time-consuming, costly component of a majority of software projects.

Graphical User Interface (GUI) development toolkits and libraries provide UI designers basic widget elements (menus, buttons, textboxes) to develop GUIs faster. Kim and Foley have shown that GUI toolkits and libraries that facilitate design activities at too low a level might allow a UI designer to create a bad or poor design quickly [7]. Therefore, UI designers use style guidelines and design principles to design more usable interfaces. Style guidelines and design principles also help to evaluate a generated design. These guidelines define the look and feel of a user interface in addition to addressing the organization of widgets, the use of color, the use of font, the use of spacing and margins, among other properties. Current examples of prominent style guidelines include Apple's Human Interface Guidelines, Microsoft's User Interface Guidelines, Sun's Java Look and Feel Guidelines, and GNU's GNOME Human Interface Guidelines [1], [9], [13], [5]. Use of style guidelines and design principles leads to a couple of issues. The first issue is that *"interpreting the guidelines unambiguously and applying generic principles to a particular design problem is itself a major challenge"* [7]. Secondly, guidelines are either too specific or too vague, so they do not always apply to the problem at hand. Here is an excerpt from Apple's Human Interface Guidelines: *"use color to enhance the visual impact of your widgets"* [1]. This guideline is incomplete and confusing in that the guideline does not tell us which color to use for a given widget and on which context this principle should be applied. Such ambiguous guidelines force UI designers to make subjective decisions and evaluations to fill in omitted details.

C. Related Work

We address research challenges from two areas: incorporating user input into IGAs, and using evolutionary techniques for UI design.

1) Evolution of UIs: Oliver, Monmarché, and Venturini explored the evolution of the websites appearance and layout [10]. The user evolves either the style or the layout of a webpage; these two optimizations are separated in order to simplify the evaluation of individuals. The user guides evolution by picking the individuals the user likes, then replacing the rest of the individuals by mating and applying high mutation rates to the selected individuals. CSS parameters like font size, font color, font family, link color, text alignment were evolved in their experiments. We expand on this work in two ways: first, our research incorporates expert knowledge (in the form of style guidelines) in addition to incorporating the subjective evaluation by a user. Second, they used a population size of 12 individuals in order to display and fit all individuals on a screen. Instead we use large population sizes and display a small subset of the best nine (9) individuals, allowing us to sample the space of UIs more effectively and to present the user with potentially high fitness individuals.

2) User Fatigue in IGAs: Cho considers IGAs a suitable tool for problems where the fitness evaluation metric is in the human mind and difficult to compute [2]. UI evolution fits in this domain since UI designers create UIs based on guidelines as well as aesthetics. Reducing human fatigue is a major research problem in this domain [14]. We address this issue by first having the user evaluate a small subset of a large population, instead of having the user evaluate the entire population. Further, the users in our study guide a UI's evolution by only selecting the best and worst individuals from the subset displayed. Hence, our study subjects do not have to evaluate or rank all individuals in the subset.

Llora, Sastry, Goldberg et al. make the user pick the best, equality relations are also allowed, from a displayed small subset [8]. Their displayed subset is a tournament during selection, so they limit the population size to minimize the number of possible tournaments. 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 an SVM to learn the user's preferences [8]. In our current work we do not attempt to do any user modeling with machine learning techniques. Instead, we use a simple interpolation based on the user selection of the best and worst to determine the fitness of every other individual in the population. Thus we reduce the user input to two decisions every generation.

Kamalian, Zhang, Takagi et al. make the user evaluate a subset of the population every t^{th} generation [6]. We adopt a similar approach to put the user in a supervisory role and thus reduce the amount of user feedback. In our previous work we explored the effect of various values of t on the IGA performance with a simulated user, where we found that as the value of t increases, noise is introduced into the fitness landscape **??**. Section VI addresses results obtained by varying the value of t with our three subjects, instead of with a simulated user. Finally, in Kamalian's work, a user provides either a demote or promote reaction to individuals displayed for user evaluation. The algorithm uses a validity constraint to determine viable and meaningful designs displayed to the user and numerous individuals can match this constraint. In a similar approach, we incorporate an objective fitness component which evaluates compliance with UI design guidelines.

D. *XUL UIs*

We use the XML User-interface Language, a cross-platform markup language for user interfaces, as the target language for our evolved UIs [17]. XUL is a powerful and extensive language and it allows widget appearance definition through CSS style sheets [17]. XUL uses JavaScript to implement widget behaviors. We choose XUL as the target language because of its flexibility and the ease with which widgets can be manipulated. XUL is also suitable for evolving the structure of UI layouts. Using XUL syntax and structure, we can create a wide range of applications, from a simple layout consisting of two buttons, to a full fledged application consisting of a menubar, toolbar, and other common widget elements. XUL forms a subset of XML, and therefore we can use XML parsers and libraries to handle the manipulation of our XUL UIs.

III. UI Representation

We encode the UI representation in two chromosomes (Fig. 1). One chromosome encodes widget layout organization, and the second chromosome encodes widget characteristics (such as widget color). We organize the widgets on a 10 rows by 2 columns grid. In user interface design a sizer usually manages widgets, and a grid sizer allows efficient widget organization in a layout. The grid layout also enforces alignment of widget, which is a style guideline in UI design. We avoided widget encoding as a bit string since standard genetic operators such as crossover or mutation could potentially destroy the representation by introducing duplicate widgets. To avoid this problem, we encode the widgets in an integer permutation string, of size 20 (10 rows by 2 columns), where each integer represents a unique identifier for each widget and 0s represent empty cells filled with spaces. The integer string maps to the 2D grid representation in a row major fashion. We chose the 10x2 grid because this results in UIs able to fit in the available space in our sample application: the Lagoon UI for the *MoveTo* panel explained in more detail in Section V.

To preserve the integer representation of the layout chromosome, we use PMX, partial mapped crossover. PMX prevents duplicate widget insertion during crossover. We use swap mutation, where we randomly pick two genes in the integer chromosome and swap their values. The integer permutation representation used for the layout of the widgets also saves us from having to compute whether widgets overlap, a computational save of l^2 for each individual (of length l) in the population (of size n), and a total save of l^2n computation every generation. Hence we can explore widget layouts by permuting widget identifiers.

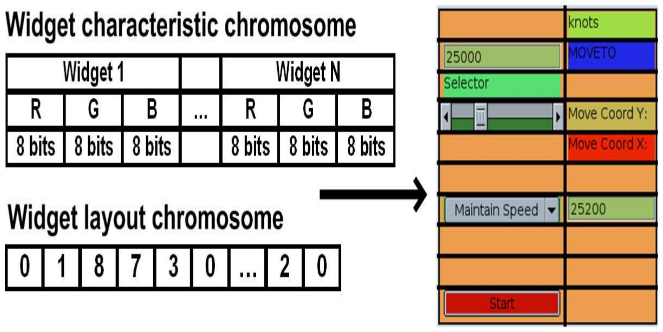

Fig. 1. UI encoding consists of two chromosomes. The widget characteristics chromosome encodes the color of each widget in a bit format. The widget layout chromosome encodes the position of the widgets in the grid. Widgets are identifi ed by integer IDs greater than 0 and empty cells in the grid are identifi ed with 0s.

The second chromosome encodes widget characteristics for each individual. This chromosome is a standard bit string and we use standard one point crossover and bit flip mutation on this part of an individual.

A. *Widget Layout*

We layout our widgets on a grid construct provided by XUL which allows us to organize our widgets in rows and columns.

We have tried using other layout organizations, including absolute positioning and positioning relative to other widgets. In absolute positioning we encoded the cardinal coordinates of our widgets, where the coordinates specified where in the panel the widgets were placed. While this was simple to implement, it resulted in widgets being placed on top of each other. This added another level of complexity to be resolved by the user by providing input into the system specifying that the UIs the user liked the best were the UIs with widgets not stacked on top of each other, instead of having the user concentrate on more useful characteristics, such as the actual widget organizations and the look and feel. We may return to this representation in the future.

Next we tried using relative positioning, where we encoded the relative positions of widgets with respect to the previous widget in the chromosome. The four positions allowed were left, right, up, and down. The first widget in the chromosome was placed on the middle of the panel, with each subsequent widget being placed relative to its predecessor in the chromosome. Without any bounds or overlap checking, we got cases where the widgets in the UI would almost line up in a straight line, resulting on elongated UIs that wasted screen space. Finally, the IGA still placed widgets on top of each other, since a widget placed to the left of a widget with a neighboring widget already on the left results in stacked widgets.

Although for the two previous representations we expect a GA to eventually untangle the layout, the permutation representation seems to be a more effective and elegant solution to the layout of the widgets. By laying out widgets in columns and rows we are able to avoid the overlapping of widgets without the n^2 computation involved to check the overlap of widgets. Most importantly, we restrict the layout of widgets to the dimensions of the grid and because of the nature of the grid layout, widgets are aligned with each other, which implicitly enforces a guideline of style.

B. Widget Color

We encode widget color on the widget characteristics chromosome. For the color we use the RGB color model, where each color is a combination of various degrees of red, green, and blue. The RGB components vary from 0 to 255 respectively. So red is $(255, 0, 0)$, green is $(0, 255, 0)$, and blue is $(0, 0, 255)$. Hence, we require 8 bits for each of the three main color components, with a total of 24 bits to represent the color of a single widget. This representation allows us to explore the 2^{24} space of colors for each widget.

The RGB model was chosen because of its support in CSS, which is how the characteristics of widgets are specified in XUL, the target language for our UIs. We could have used the HSV color model [**?**], but its gamut is the same as RGB, and experiments have shown that there is no significant efficiency difference in the RGB and HSV color models [3], [16]. Therefore, we decided to stick to RGB, however we treat RGB colors as vectors in a 3D color-space.

IV. Fitness Evaluation

Our IGA's fitness evaluation consists of two steps: (1) user input evaluation, and (2) objective metric conformance checking. In the first step we have the user make two selections, the UI the user likes the best and the UI the user likes the least. We use these two selected UIs to evaluate the subjective fitness component of all other individuals in the population through interpolation. In the second step the GA looks through the UIs in the population and checks to see how well they adhere to or violate coded guideline metrics. We then add the subjective and objective fitness components in a linear weighted sum. For this experiment we used equal weights for the subjective and objective fitness components.

A. Subjective Evaluation

Our earlier work discusses our decision to choose a subset consisting of best individuals in the population [12]. We compute the similarity between two individuals in two steps. In the first step, we calculate color similarity of the two UIs, in terms of the widgets and the panel background. To determine color similarity, we calculate the euclidean distance between two colors. We reward a small distance between the widget color in individual i and the user selected best individual b. On the other hand, a large distance between the widget color in individual i and the user selected worst individual w is rewarded. Next, we compute widget layout similarity. Here we compute the hamming distance between the permutation layout chromosomes of the two individuals. This fitness is inversely proportional to the hamming distance between individual i and the user selected best b and directly proportional to hamming distance between i and the user selected worst. Finally, we scale the subjective component to make it comparable to the objective component.

We compute similarity between the best individual b and individual i in the population as follows:

$$b_s = \sum_{k=1}^{m} \frac{M - dist(e_{b,k}, e_{i,k})}{M} * 100 \\ +(MH - hamming(b, i))$$

$$w_s = \sum_{k=1}^{n} \frac{dist(e_{w,k}, e_{i,k})}{M} * 100 \\ +hamming(w, i)$$

$$subjective = b_s + w_s$$

The term within the summation computes color similarity and the second line, the layout similarity. b_s is the subjective fitness component computed with reference to the user-selected best individual while w_s computes the subjective fitness component with reference to the user-selected worst individual. In the formulas above, M is the maximum distance between any two colors, $\sqrt{255^2 \times 3} = 441.68$ and $dist(e_{b,k}, e_{i,k})$ is the euclidean distance between the k^{th} widget of the best individuals and the k^{th} widget of individual i. MH is the maximum hamming distance ($l = 20$). We finally scale the subjective fitness to lie between 0 and and 1000.

B. Objective Evaluation

We compute the objective fitness component by checking how well UI individuals in the population adhere to and respect coded style guidelines. Our first coded color style guideline checks whether a UI has a high contrast between background panel color and widget foreground colors. Maintaining a low contrast between widget colors is our second coded color style guideline. We prefer the high contrast between background and widget colors to ensure legibility. The low contrast between widget colors ensures that widgets have a similar shade of color, instead of having each widget in a UI with an independent color. The use of the grid positioning to layout widgets enforces their alignment, which is a style guideline too.

We iterate through the widgets of each UI layout and compute the euclidean distance from each widget color to background panel color to check high contrast between the background panel color and widget colors. We consider a large distance between widget j and the panel background color as a high contrast value. We sum all the euclidean distances, rewarding individuals that have a high euclidean sum. Next, we compare each widget j in a UI layout to every other widget (an n^2 computation) in the layout, taking their euclidean distances and adding them up. Large euclidean distance values between two widgets means that the widgets do not have a similar shade of color. We do this to cluster the colors in 3D space into a center of gravity which defines the color shade that all these colors should share in common. A large sum of the euclidean distances means that all widgets have very different colors, and hence they are spread out far from each other thereby violating our style guidelines. We therefore assign a low reward to such an individual. A small sum of the euclidean distances means that the widgets are clustered together and share a similar shade of color. This individual fulfills our style guideline and we therefore assign a high reward. We sum the rewards from the high contrast between widget colors and background color and low contrast between widget colors. Finally, as with the subjective fitness, we scale this objective value to also lie between 0 and 1000.

$$obj_1 = \sum_{k=1}^{m-1} \sum_{j=k+1}^{m} \frac{dist(e_{i,k}, e_{i,j})}{M} * 100$$

$$obj_2 = \sum_{k=1}^{m} \frac{M - dist(e_{i,k}, window\_bg_i)}{M} * 100$$

$$objective = obj_1 + obj_2$$

After we compute the subjective and objective fitness components, we take a linear weighted sum of the two to determine the fitness of each individual:

$$fitness = w_1 * objective + w_2 * subjective$$

where w_1 is the objective component weight, w_2 is the subjective component weight, $objective$ is the fitness objective component and $subjective$ is the subjective fitness component. The weights w_1 and w_2 are complements of each other, with values between 0 and 1. We used values of 0.5 and 0.5 for w_1 and w_2 respectively for the experiments discussed in section V.

C. Parasitism

We are evolving and trying to optimize the layout and the look of the widgets in a panel. Consequently, we have multiple criteria that we are trying to optimize. This has led to parasitic behavior on the evolution of UIs. The user picks the UI the user likes the best and the UI the user likes the least. However, the user does not specify these in terms of what exactly the selection is being made on. When the user picks a UI as the best, this leads to the GA attributing a high fitness to both the look and the layout of the widgets. For example, if the user picks a UI because of the vibrant blue colors the widgets have, then a high fitness will be attributed to whatever layout the widgets have. Thus, on our simulated user picking, it is assumed that the user picks the best and worst based on color alone, ignoring the layout of the widgets. Widgets associated with the most blue UIs, regardless of orderliness, will be given a high fitness as well.

In the current implementation we have not incorporated a means with which to prevent the emergence of this parasitic behavior. This could be suppressed by fixing either the layout or the look of the widgets, and evolving the other non fixed parameter. Alternatively, the user could be asked to select the best UI based on widget layout and the best UI based on widget look. However, this adds to the number of selections that have to be made by the user, thus increasing user fatigue.

V. Experimental Setup

For this paper, we collected data from 3 users. Our IGA's parameter settings are as follows: (1) population size of 100, (2) we displayed 9 individuals for user evaluation, and (3) we used probabilistic tournament selection with a tournament size of 4, 90 percent probability of choosing the tournament best individual (otherwise we choose a random individual from the tournament losers).

Three users participated in five IGA sessions, each session lasting 30 generations. For these five sessions, we used t values of 1, 3,5, 10, and 15, allowing the user to bias the evolution of the UIs 30, 10, 6, 3, and 2 times respectively. We instructed the users to choose the UI they liked the best and the UI they liked the least, based on whatever criteria they desired.

Users participating in our IGA sessions guided the evolution of the *MoveTo* interaction panel that controls combat ships in *Lagoon*, a real-time 3D naval combat simulation game developed in our lab [4]. The MoveTo panel consists of five text labels, a button, a drop-down menu, a slider, and two textboxes, all written in XUL and loaded into the IGA. We chose the MoveTo panel because it has a variety of widely used widgets, yet it is simple enough for our initial experiments.

Our experiment investigated the effects of delayed user input. In our previous work we had a simulated user do the IGA runs [12]. The simulated user liked blue widgets and greedily picked UIs with the most blue widgets as the best UI and the UI with the least blue widgets as the worst UI. Our simulated user results showed that asking for less user input leads to increased noise in fitness evaluations. In this paper, we explore how varying t affects convergence behavior and performance with real users.

VI. Results

We plotted the fitness convergence for our three study subjects: $user1$, $user2$, and $user3$. Figure 2 shows user1's session fitness convergence of the best individuals in the population for $t = 1, 3, 5, 10,$ and 15. We can step like increases for $t = 1$ and $t = 3$ as the user varies their selection of the UI they like the best. Sharp increases in fitness reflect the user choosing an individual that also conforms to the objective metrics. Note that in our IGA, the population will constantly evolve towards UIs that reflect the objective design metrics, hence the fitness increases over time. Through the generations the user sees individuals that increasingly reflect

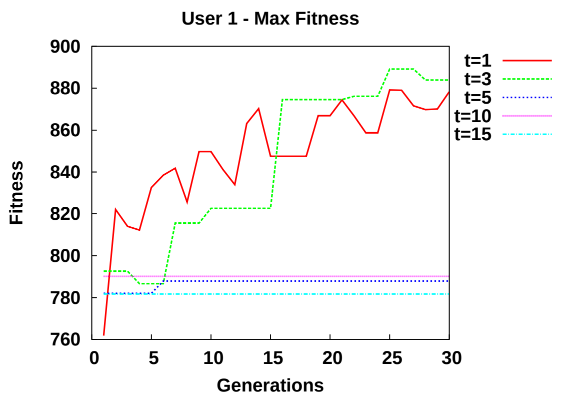

Fig. 2. Fitness performance of user 2. The plot shows the best individuals in the population.

conformance to the objective metrics, yet which resemble individuals the user liked. The fitness increase shows the successful fusion of computable objective metrics and user subjective input guiding the evolution of the UIs. Lastly, notice that for a value of $t = 3$ user1 is able to achieve a higher fitness than with $t = 1$. We did not expected this behavior since our previous results with a simulated user showed that giving the simulated user complete control over the UI evolution by allowing them to participate in every generation resulted in the highest fitness performance [12]. Also, we noticed that the maximum fitness for values of $t = 5, 10$, and 15 remain constant. We attribute this behavior to user1 not changing their selection of the best UI during the entire session. With low values of t a user has more opportunities to change the selection of the best and worst UIs, (30 chances with $t = 1$ and 10 chances with $t = 3$).

Figure 3 shows user2's session fitness convergence of the best individuals for the same values of t. We see that for

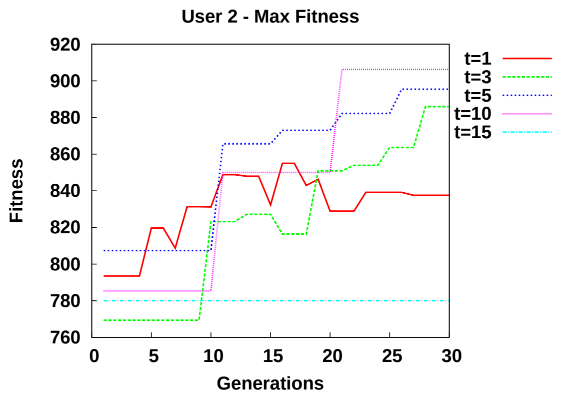

Fig. 3. Fitness performance of user 2. The plot shows the best individuals in the population.

user2, $t = 10$ achieved the highest fitness, and for $t = 15$ user2 did not change their selection of the best UI during the entire session. User2 was also able to successfully bias the evolution of the UIs by fusing objective and subjective criteria. Figure 4 shows the fitness plot for user3. User3 presents interesting results, since his varied selections of the best UI helped in finding high fitness values for all t. Notice that for all three users using a value of $t = 1$ did not result in the highest fitness convergence. Figures 5, 6, and 7 show the fitness plot of the population average for the three users. The steep drops in average fitness performance correspond

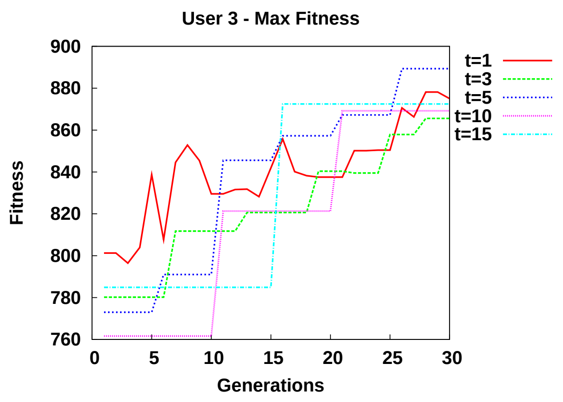

Fig. 4. Fitness performance of user 3. The plot shows the best individuals in the population.

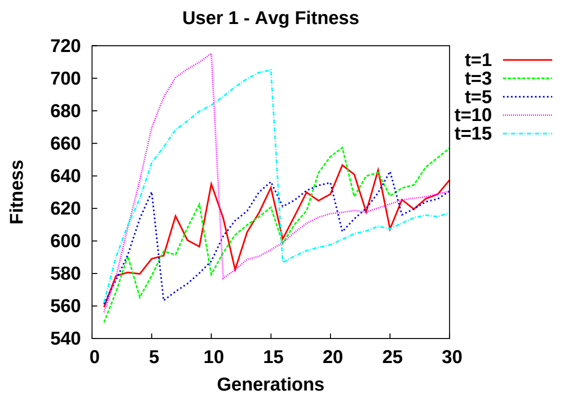

Fig. 5. Fitness performance of user 1. The plot shows the average individuals in the population.

to the time steps where the user makes a selection of the best and worst UIs. These average fitness performance results are similar to our previous results with a simulated user.

We expected to see a drop in average performance associated with the user changing the selection of the best and worst UIs, but we can see a drop in performance even when the user does not change his selection through the session as was seen with user1 (Figure 2) and user2 (Figure 3). The drop in fitness performance associated with user input can be a result of a user changing the least-preferred UI while not changing the best-preferred UI. Our comparison with the user selected worst individual is what causes the drop in fitness performance associated with user input. To test this hypothesis, we conducted another session run with a user, where the user was instructed to pick a UI at the beginning of the session and to continually pick that individual throughout the rest of the run. We had the user do this on two sessions, where in one of the sessions we turned off the comparison to the user selected worst UI. Finally, we used $t = 3$, since none

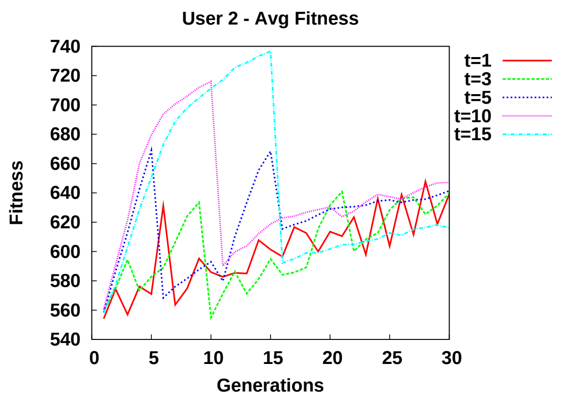

Fig. 6. Fitness performance of user 2. The plot shows the average individuals in the population.

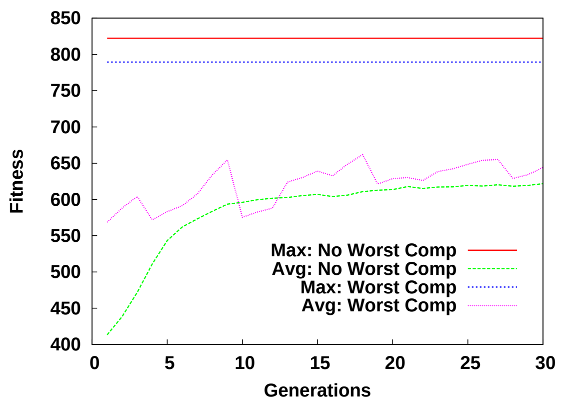

Fig. 8. Fitness performance of comparing to user selected worst and without the comparison.

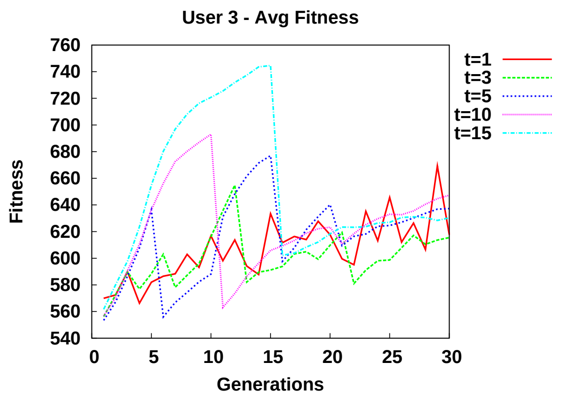

Fig. 7. Fitness performance of user 3. The plot shows the average individuals in the population.

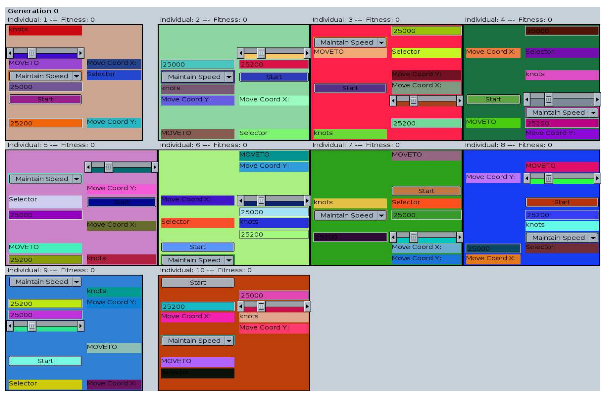

Fig. 9. The best 9 individuals in the initial population.

of the users picked the same UI as the best for $t = 1$ and $t = 3$, hence we wanted to confirm the conjecture that such behavior was less common with low values of t since the user has more opportunities to change his selection. Figure 8 shows these results. The plot shows the fitness convergence of the best and average individuals in the population with and without comparison to the UI the user likes the least. We can see that having the user pick the same UI as the best at every time step results in constant maximum fitness as we saw in Figures **??**. Notice that comparing individuals in the population to the UI the user likes the least results in steep drops in fitness performance associated with the time step (every 3 generations) in which the user makes a selection. We also see from the plot that removing the comparison to the user selected worst individual results in a monotonic increase in fitness performance. This supports our hypothesis that the comparison to the UI the user likes the least accounts for the sharp fitness drops, even when the user selected best UI remains constant. It also supports the conjecture that with low values of t the user has more opportunities to change the selection of the best and worst UIs.

A. *User Experience*

Doing all 5 IGA runs (for values of $t = 1, 3, 5, 10,$ and 15) took about 30 minutes to complete, with the session using a value of $t = 1$ (user input every generation for a maximum of 30 generations), taking over half the time (20 minutes) to complete. We found that using a value of $t = 1$ results in slow changes from generation to generation, forcing the user to pay more attention to detail. One of the users commented that using high values of t usually converged to likable UI colors, without having to spend a lot of time making a selection every generation. Even though 30 generations is not a big number, having to make a selection every generation ($t = 1$) still results in user fatigue. Higher values of t seem to significantly reduce user fatigue and lessen the time spent on each session [8].

B. *UIs Generated*

Figures 9 and **??** show a subset consisting of the 9 best individuals in the population at generations 0 and 30 respectively for user3. The figures show the session using $t = 15$. In generation 0, widgets start with random positions and random colors. In generation 30, we can see the best UIs which reflect both the user3's preferences and which best follow coded guideline metrics.

VII. Conclusion and Future Work

We presented an IGA that combines both computable metrics, taken from style guidelines, and human subjective input to guide the evolution of UIs. The design process

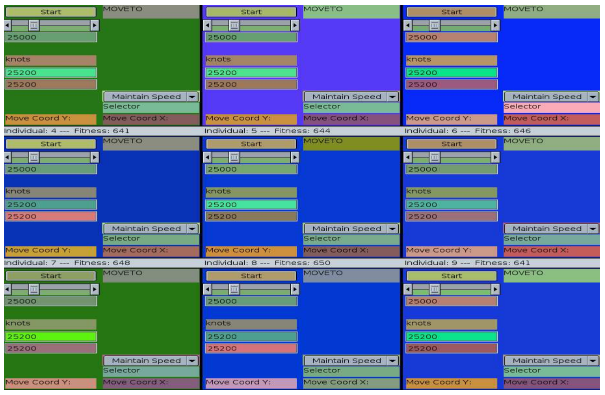
Start
MOVETO
25000
knots
25200
25200
Maintain Speed
Selector
Move Coord Y:
Move Coord X:
Individual: 4 --- Fitness: 641
Individual: 5 --- Fitness: 644
Individual: 6 --- Fitness: 646
Individual: 7 --- Fitness: 648
Individual: 8 --- Fitness: 650
Individual: 9 --- Fitness: 641

Fig. 10. The best 9 individuals at session end.

is driven by both formalized style guidelines and by a human sense of aesthetics, making our approach a suitable and promising application that can change the way UIs are designed and maintained.

Within this context, we investigated three methods to reduce user fatigue in IGAs by 1) displaying a subset of the best 9 individuals from the population, 2) asking the user to select the best and worst UIs from the subset displayed for user evaluation, and 3) assessing the effects of limiting user input by having the user pick every t generations.

We had 3 users evolve UIs with our tool to explore how often to ask for user input. High values of t can reduce human fatigue and reduce the time spent by the user on a session. Users are able to effectively bias the evolution of the UIs by making the selections of the UIs they like the best and the UIs they like the least. Through interpolation we were able to reduce the number of selections to two every generation.

We believe that the work presented in this paper lays a good foundation for future research and development. We would like to conduct further experiments by varying the value of t over the course of a single run, exploring how asking for user input often early in an IGA session (when there is a high degree of diversity in the population), and asking for less user input in later generations (when the population approaches convergence) affects performance. Further, we would like to expand widget encoding to support coupling between widgets and high level spatial relationships with other widgets and the parent panel.

We also plan to incorporate more heuristics from the various style guidelines into the objective evaluation component. Further user studies need to be conducted, with a larger sample, to evaluate the utility of the tool and the effectiveness with which users can guide and bias the evolution of UIs. We also plan to investigate using the longest common subsequence metric as our similarity measure for the layout chromosome. Last, we plan to further explore representations and genetic operators that can yield higher fitness individuals in less generations and with higher confidence intervals. We need a representation which yields smooth gradients during crossover and mutation. A color model that better correlates to how we define similarity among colors will help.

The long-term goal of this evolutionary approach to UI design is to streamline and help reduce the complexity associated with the generation and the fine-tuning of UIs. We believe that the research reported in this paper shows the viability of an interactive evolutionary approach to UI design.

VIII. Acknowledgments

This material is based upon work supported by the Office of Naval Research under contract number N00014-03-1-0104 and upon work supported by the National Science Foundation under Grant No. 0447416.

References

[1] Apple. Apple human interface design guidelines: Introduction to apple human interface guidelines, 2006.

[2] S.-B. Cho. Towards creative evolutionary systems with interactive genetic algorithm. *Applied Intelligence*, 16(2):129–138, 2002.

[3] S. A. Douglas and A. E. Kirkpatrick. Model and representation: the effect of visual feedback on human performance in a color picker interface. *ACM Trans. Graph.*, 18(2):96–127, 1999.

[4] ECSL. Lagoon, 2006.

[5] GNOME. Gnome human interface guidelines 2.0, 2004.

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

[7] W. C. Kim and J. D. Foley. Providing high-level control and expert assistance in the user interface presentation design. In *CHI '93: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 430–437, New York, NY, USA, 1993. ACM Press.

[8] X. Llora, K. Sastry, D. E. Goldberg, A. Gupta, and L. Lakshmi. Combating user fatigue in igas: partial ordering, support vector machines, and synthetic fi tness. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1363–1370, New York, NY, USA, 2005. ACM Press.

[9] Microsoft Corporation. Windows xp - guidelines for applications, 2006.

[10] A. Oliver, N. Monmarché, and G. Venturini. Interactive design of web sites with a genetic algorithm. In *Proceedings of the IADIS International Conference WWW/Internet*, pages 355–362, Lisbon, Portugal, november 13-15 2002.

[11] J. Preece, Y. Rogers, and H. Sharp. *Interaction Design: Beyond Human Computer Interaction*. Wiley, 2002.

[12] J. C. Quiroz, S. M. Dascalu, and S. J. Louis. Human guided evolution of xul user interfaces. In *CHI '07: CHI '07 extended abstracts on Human factors in computing systems*, New York, NY, USA, 2007. ACM Press.

[13] Sun Microsystems. Java look and feel design guidelines, 2001.

[14] H. Takagi. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, Sept. 2001. Invited Paper.

[15] H. Thimbleby. User interface design with matrix algebra. *ACM Trans. Comput.-Hum. Interact.*, 11(2):181–236, 2004.

[16] Y. Wu and M. Takatsuka. Three dimensional colour pickers. In *APVis '05: proceedings of the 2005 Asia-Pacific symposium on Information visualisation*, pages 107–114, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[17] XULPlanet. Xulplanet.com, 2006.