Interactive Evolution of XUL User Interfaces

ABSTRACT

We attack the problem of user fatigue in using an interactive genetic algorithm to evolve user interfaces in the XUL interface definition language. The genetic algorithm combines a set of computable user interface design metrics with subjective user input to guide the evolution of interfaces. Our goal is to provide UI designers with a 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 UI without programming knowledge. User interface specifications are encoded as individuals in a genetic algorithm's population and their fitness is computed from a weighted combination of user interface design guidelines and user input. This paper shows that we can reduce human fatigue in interactive genetic algorithms (the number of choices needing to be made by the designer), by 1) only asking the user to pick two UIs from among ten shown on the display and 2) by asking the user to make the choice once every t generations, where t depends on the selection algorithm used.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Theory and methods*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Design, Human Factors

Keywords

Interactive Genetic Algorithm, User Interfaces

1. 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, "very little knowledge in design generalizes beyond specific case studies" [13]. 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 [12].

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 a small subset of the population and picks two - the best and worst looking interfaces. Empirical observations tell us that we should not display more than ten items to be judged by a user [11] but the composition of the subset displayed for user evaluation creates rich dynamics that affect the convergence behavior of the population in the genetic algorithm. We address two issues that affect convergence behavior. First, should we show the user the top ten individuals in the population, a mixture of the ten best and worst individuals in the population, or display a random set of ten individuals in the population? Second, how often do we need to ask the user for feedback? Results show that displaying the top ten individuals results in faster convergence and better interfaces. In addition, we show that we can use genetic algorithm theory to choose the correct value for "t," the number of generations between displaying the chosen individuals to the user and asking for their evaluation.

The remainder of this paper is organized as follows: Sec-

tion 2 discusses background information in IGAs, the challenges of UI design, and related work. Section 3 presents the encoding and representation used for the UI individuals in the population while the next two sections discuss how we do subjective and objective evaluations and our experimental setup. The last two sections explain and discuss our results and present our conclusions and directions for future work.

2. BACKGROUND

In the following subsections we explore IGAs, the inherent challenges of UI design, and related work on the topic of UI design. We also explain our choice of XUL as the target language for our UIs.

2.1 Interactive Genetic Algorithms

Interactive genetic algorithms fuse the power of evolutionary computation and human subjective evaluation by providing a mapping from psychological space to parameter space [12]. By doing so, IGAs incorporate human knowledge, emotion, intuition, and preference into GAs. Usually we can compute the fitness of individuals in a GA based on a math equation, some computation, or a model [8]. However, a user cannot be trivially modeled; user preferences are relative and subject to change with time and context. IGAs incorporate user subjective evaluation by replacing the fitness evaluation with the user, where the user provides the fitness to individuals in the population by assigning a number on a subjective scale, ranking individuals, or choosing the best individual from a displayed subset [8, 12]. Because of the nature of IGAs, they have been used for a variety of applications which incorporate creative human input, including editorial design, industrial design, image processing, database retrieval, graphic art and CG animation, control and robotics, among others [12].

Effective IGAs have to overcome several issues. GAs usually rely on large population sizes running for many generations, but asking a user to make hundred or thousands of choices may be just a little unrealistic. A user would rapidly fatigue and/or lose interest. Furthermore, because of the subjective nature of human input, it can lead to users changing their goals through the IGA run, leading to noisy landscapes - which coupled with user fatigue can lead to suboptimal solutions [8].

UI design, discussed more in detail in the next subsection, is a process which combines objective and subjective heuristics. As such, an IGA is a suitable tool which uses evolutionary techniques, coupled with human input to help guide the evolution of the population.

2.2 User Interface Design

User interface design is a complex process critical to the success of a software system; designing interactive systems that are easy to use, engaging, and accessible is a challenging task. Consequently, the design of a user interface is a costly part of any software project.

Graphical user interface development toolkits and libraries help user interface designers to develop graphical user interfaces (GUIs) faster by providing basic widget elements, such as menus, buttons, and textboxes. Because GUI toolkits and libraries facilitate the design activities at too low a level, they may allow the designer to create a bad or poor design quickly [7]. UI designers therefore also use style guidelines and design principles to guide their designs and this hopefully leads to more usable interfaces. In addition, such guidelines and design principles provide a means with which to evaluate a generated design. Style guidelines not only define the look and feel of a user interface, but they also address the organization of widgets, the use of color, the use of font, the use of spacing and margins, among other properties. Some prominent style guidelines are Apple's Human Interface Guidelines [1], Microsoft's User Interface Guidelines [3], Sun's Java Look and Feel Guidelines [9], and the GNOME Human Interface Guidelines [5]. The problem lies in that "interpreting the guidelines unambiguously and applying generic principles to a particular design problem is itself a major challenge" [7]. There is also the problem that guidelines are either too specific or too vague, so they do not always apply to the problem at hand. For example, an excerpt from Apple Human Interface Guidelines specifies: "use color to enhance the visual impact of your widgets," but no detail is given as to which color to use for a given widget and context [1]. Therefore, user interface designers are forced into making subjective decisions and evaluations to fill in the details that guidelines omit.

2.3 Related Work

Our work addresses research challenges from two fields: the incorporation of user input into IGAs, and the use of evolutionary techniques for UI design.

2.3.1 Evolution of UIs

The evolution of the appearance of websites was explored in [10]. The tool described allows the user to choose among the website styles he/she likes the best, and this guides the evolution of the population. It is assumed that the content of the page is fixed, and what is evolved is the cascading style sheet (CSS) for the website. Parameters evolved include font size, font color, font family, link color, text alignment, and other properties that can be configured through CSS. We seek to expand on the evolution of website appearances discussed in [10] by incorporating expert knowledge, in the form of style guidelines, into the evaluation function, instead of having a purely subjective evaluation by the user. Furthermore, we will not only be evolving style, but also widget layouts.

2.3.2 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" [2]. 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 [12]. We address this issue by first showing the relationship between individuals displayed and the population size of the GA. Specifically, we show that users can guide the evolution of user interfaces, and are able to evolve interfaces to their liking by only selecting the best and worst individuals from a small subset of the entire population, instead of having to evaluate or rank all individuals in the population.

In [8] the user picks the best from a small subset displayed to the user, but what is displayed to the user is a tournament during selection, where the user picks the best of the solutions displayed. Then, the user selected best input gets a copy in the mating pool; user picking is also used to infer an ordering of individuals based on winners and losers. In contrast, 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 reducing the user input to two decisions every generation. Furthermore, as in [6] 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. In [6] the user gives either a promote or demote reaction to individuals displayed for user evaluation. A validity constraint is used to determine viable and meaningful designs to be displayed to the user. Individuals matching the validity constraint can be numerous; instead 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 GA's performance. How to choose a good value for t is addressed in our methodology section.

2.4 XUL UIs

The target language used for the UIs being evolved is XUL, the XML User-interface Language, a cross-platform markup language for user interfaces [14]. XUL is a powerful and extensive language allowing the defining of widget appearance through CSS style sheets and the use of JavaScript to implement widget behaviors [14]. XUL was chosen as the target language because of its flexibility and the ease with which widgets can be manipulated. XUL is also suitable for the manipulation necessary to evolve the structure of UI layouts. The syntax and structure of XUL allow us to 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. Lastly, as a subset of XML, we can use XML parsers and libraries to handle the manipulation of our XUL UIs.

3. UI REPRESENTATION

Widgets are laid out on a grid, with a layout being a permutation of widget identification numbers. Because we also encode aesthetic properties of widgets (such as widget color) which do not use a permutation encoding, we use two chromosomes to specify the UI. We assign integer identification numbers, ranging from 1 up to the maximum number of widgets in the layout, to every widget. Zero denotes blank spaces on the grid. A sample layout and its encoding is shown in Fig. 1. In order to preserve this permutation representation, we use PMX, partial mapped crossover, as the crossover operator for this chromosome. PMX keeps crossover from creating individuals with duplicate genes, which would violate the permutation property. We also use swap mutation, which randomly picks two individuals in the chromosome and swaps them. The permutation representation is effective in that the positions of the widgets in the layout are reduced to a shuffling of the permutation of widget identification numbers. The second chromosome encodes the characteristics of each widget in a layout; currently this consists of the color of each widget. The encoding for this chromosome is a bit string - 0s and 1s. We use single point crossover and bit flip mutation on this chromosome.

3.1 Widget Layout

We organize the layout of our widgets using a grid based system. The grid construct is provided by XUL and it allows us to organize our widgets in rows and columns.

Previously we 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 with each widget we encoded its position relative to the previous widget on 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 on a straight line, resulting on elongated UIs that wasted screen space. We also encountered the problem that with relative positioning we still obtained widgets placed on top of each other, since a widget placed to the left of a widget with a neighboring widget already on the left results on 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.

For the experiments conducted and discussed in this paper we have fixed the grid size to 10x2, 10 rows and 2 columns. We chose this grid size because it is similar in dimension to the available space on the Lagoon UI [4] discussed in Section 5.1. Widgets are placed on the layout in a row-major fashion, with remaining empty cells filled with blank spaces. Hence, the layout of widgets boils down to a permutation of widgets and blank spaces, and by shuffling the permutation we are able to explore different widget layouts.

3.2 Widget Color

We encode the color of widgets on a regular bit string. For the color we use the HTML hexadecimal color representation, where each color consists of three components: red (FF0000), green (00FF00), and blue (0000FF). The RGB components vary from 0 to 255 respectively, with 0 representing black and 255 representing white. Hence, we require 8 bits for each of the three main color components, with a

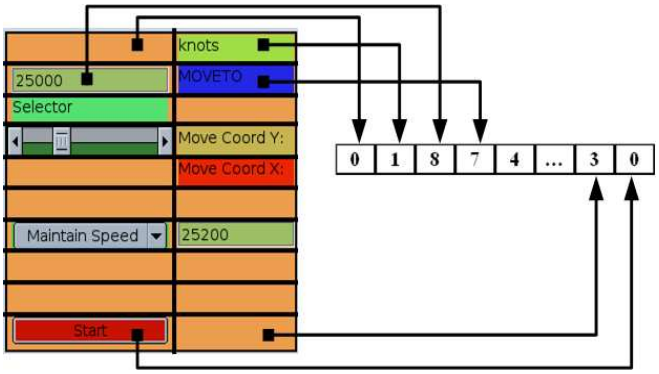

Figure 1: Encoding of the layout of widgets. Widgets are identified by integer IDs greater than 0 and empty cells in the grid are identified with 0s.

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 the widgets.

4. FITNESS EVALUATION

The fitness evaluation of the individuals in the IGA is a two step process. The user is first asked to provide input, which constitutes the subjective component of the fitness evaluation. After recording this user input, we check conformance to a set of objective heuristics taken from our style guidelines. Fitness is then computed as the weighted sum of objective and subjective components. The corresponding weights can be adjusted by the user and can cause GA behavior to range from pure objective evaluation to pure subjective evaluation. For the experiments and results discussed in this paper we used equal weights for the objective and subjective components.

4.1 Objective Evaluation

The objective fitness is obtained by checking the validity of style guidelines in the user interface layout represented by an individual. The two main guidelines currently incorporated on the objective evaluation are the use of a high contrast between the background and foreground color, and the use of a similar shade of color for widgets. The grid positioning of the widgets in the layout implicitly enforces a widget alignment guideline.

We enforce a high contrast between the widgets' color and the panel color by taking the difference between the panel color and the color of each widget in the panel. Colors are treated as vectors, so taking the difference consists of finding the distance between two color vectors. We add the differences of color found, with a high difference attributed a high fitness and a low difference attributed a low fitness. Similarly, to enforce a low contrast between widget colors we compare the color of each widget with every other widget, summing up the difference of the colors. A low difference is attributed a high fitness, whereas a high difference is attributed a low fitness. The objective computation is therefore:

$$\text{objectiveFitness} = HCD + LCD$$

where HCD is the high contrast difference between each widget color and the panel background color, and LCD is the low contrast difference between widgets. The objective component value is then scaled between 0 and 1000 with the following formula:

$$\text{scaledObjectiveFitness} = OF/SOF * 1000$$

where OF is the objective fitness component value obtained through metric validation and SOF is the maximum possible objective fitness component value.

4.2 Subjective Evaluation

The subjective evaluation consists of taking the user choice of best and worst individuals displayed, and using this to assign a "fitness" to each individual in the population. We allow the user to choose only the best and worst in order to address user fatigue in evaluation. The user chosen worst individual, w, gets a subjective fitness of 0. The user selected best individual, b, gets the maximum subjective fitness which is relative to the length of its two chromosomes. Next, we interpolate the subjective fitness of every other individual, i, by 1) decoding the colors of individual i, 2) finding the distance between the colors used by i and b, 3) summing up the color distance values, 4) scaling the color distance values, giving a high value to small color distance values and a low value to high color distance values, and 5) compute hamming distance between widget layout permutation chromosomes. These steps are then repeated by comparing individual i with w, the user selected worst UI. The subjective component value is then scaled similarly to the objective fitness component, between 0 and 1000, with the following formula:

$$\text{scaledSubjectiveFitness} = SF/SFM * 1000$$

where SF is the subjective component value obtained through interpolation and SFM is the maximum possible subjective component value.

4.2.1 *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.

5. EXPERIMENTAL SETUP

For the experiments conducted we used a population size of 100 and we displayed 10 individuals for user evaluation. We compare two selection methods, roulette wheel selection and probabilistic tournament selection. For tournament selection we used a tournament size of 4, with 90% probability of choosing the best individual in the tournament. Four individuals from the population are randomly sampled to form a tournament for parent selection. The IGA was run over 30 independent runs.

5.1 The Lagoon UI

We used Lagoon, a real-time 3D naval combat simulation game developed at the Evolutionary Computing Systems Lab (ECSL) at UNR as a platform for AI research [4]. We tested our approach on one of the interaction panels from the complex Lagoon UI, the "MoveTo" panel that controls combat ships in the game. The widgets in the MoveTo panel, five text labels, a button, a drop-down menu, a slider, and two textboxes, were written in XUL and loaded into the IGA. The MoveTo panel was chosen because it has a variety of widgets, yet it is simple enough for our initial experiments. We conducted two experiments; the first to investigate which individuals to display and the second to investigate how often we need to pick. All results reported below are averages from 30 independent runs.

Instead of using real people we used a simulated human with a preference for the color blue. Given a set of UIs displayed for user evaluation, we used a greedy approach to simulate user picking, and the UI with the most blue widgets was chosen as the best, and the UI with the least blue widgets was chosen as the worst layout.

We chose to test three methods for selecting our $n = 10$ individuals in the population for user evaluation. The first method displayed the best n individuals in the population. The second method displayed both the best $n/2$ and the worst $n/2$ individuals in the population. The last method randomly selected n individuals in the population to be displayed for user evaluation.

6. RESULTS

As expected, we found that using tournament selection with a tournament size of 4 outperformed roulette wheel selection (see Fig. 2). The figure shows the best individuals in the population. Tournament selection's stronger selection pressure leads to much quicker convergence to better values.

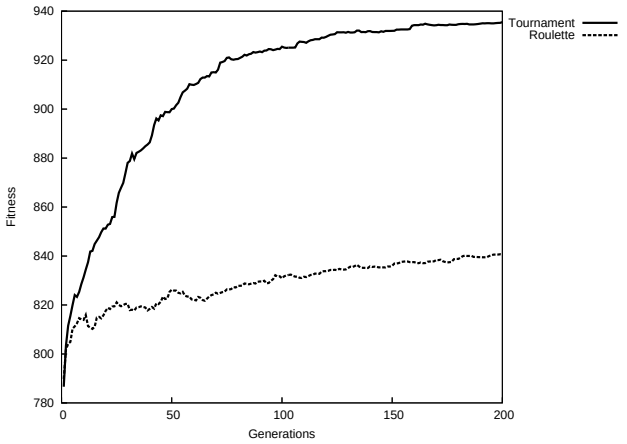

Figure 2: Tournament selection versus roulette wheel selection. The plot shows the best individuals in the population.

6.1 Subset Display Method

We compared three methods of selecting individuals to be displayed to the user. 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 UI designs to the user. Lastly, displaying both best and worst individuals from the population allows the user to see what the population is converging to and where it is coming from.

We ran the IGA with each of the three display methods using tournament selection and plotted the fitness of the best individuals in the population as shown in Fig. 3. We can see that displaying the best individuals in the population for user evaluation results in the best IGA performance when compared to displaying random individuals and displaying both the best and the worst individuals in the population. Fig. 4 also shows that our (simulated) user is able to bias IGA performance effectively by displaying the best individuals in the population for subjective evaluation. Remember, blue widgets was the user's assumed preference. Displaying the best and worst individuals in the population results in individuals with blue widgets, but which violate the style guideline metrics that we are trying to enforce through the objective evaluation.

6.2 The Power of t

We varied the value of t to explore the effects of user input every t^{th} generation on IGA performance. That is, the user was only asked to make a choice once every t generations and we used that choice for the next t generations to interpolate individuals' fitness. Figure 5 compares convergence behavior for $t = 1, 5, 10$, and 20, where we have plotted the average fitness over 30 runs of the best individuals in the population. We were encouraged to see that varying t, for small values of t, has little effect on the IGA's convergence behavior. Next, to look at the effect of changing t on the subjective fitness, we plotted the convergence to blue widgets in Fig. 6 (again this is average of the best individuals). Note that even a small change in t results in a drop in convergence to blue UIs as shown in the figure. With less frequent user input we get increasingly noisy subjective fitness evaluation.

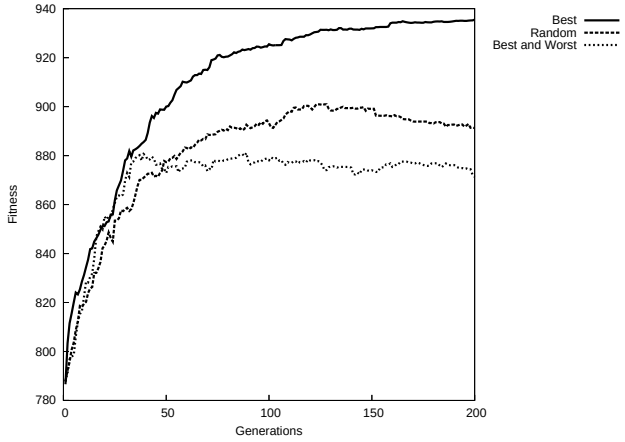

Figure 3: Subset display method comparison.

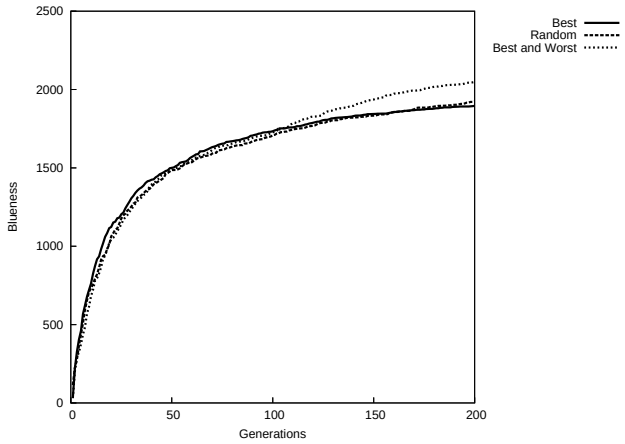

Figure 4: Subset display method comparison on convergence to blue widgets.

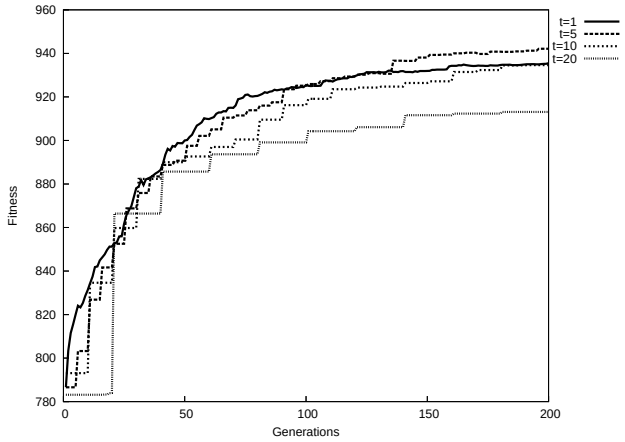

Figure 5: Effect of varying t on IGA performance.

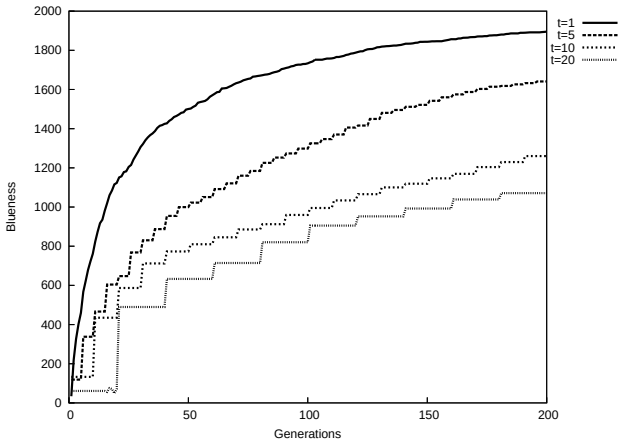

Figure 6: Effect of varying t on convergence to blue UIs.

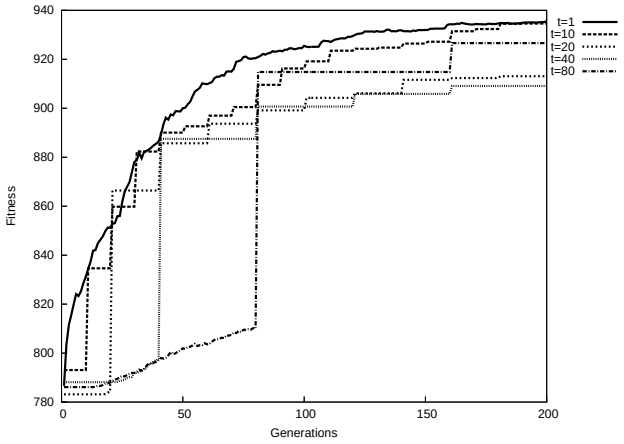

Figure 7: Degradation on the IGA performance (maximum) for high t values.

We increased the value of t to 20, 40, and 80 generations to assess the effect on IGA performance. Figure 7 shows the fitness plot of the best individuals in the population. We can see the step-like increase of fitness corresponding to the generation when our user makes a selection. Figure 8 shows the fitness plot of the average individuals in the population. The sharp decrease in fitness in early generations corresponds to the generation in which the user makes the second picking, since the first user picking is done upon population initialization. We then see a slow increase in fitness.

We also plotted the convergence to blue UIs, which was the user assumed preference. Figure 9 shows the "blueness" of the best individuals in the population. From the figure we see that increasing values of t leads to decreasingly blue UIs. Thus, as expected, less user input results in a less effective subjective bias on the population. Finally, Fig. 10 shows the average blueness of individuals in the population indicating that the average performance correlates well with best performance.

6.3 UIs Generated

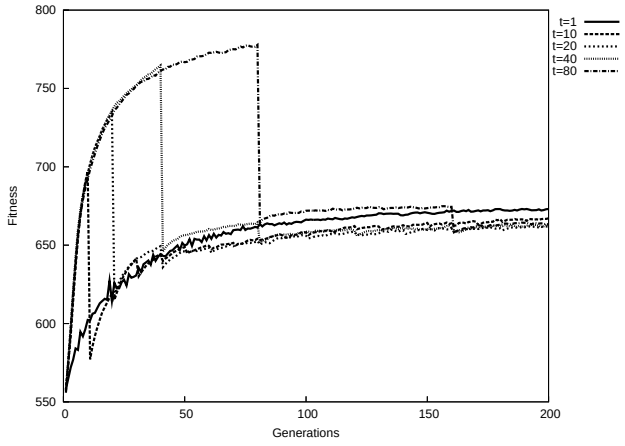

Figure 8: Degradation on the IGA performance (average) when using high t values.

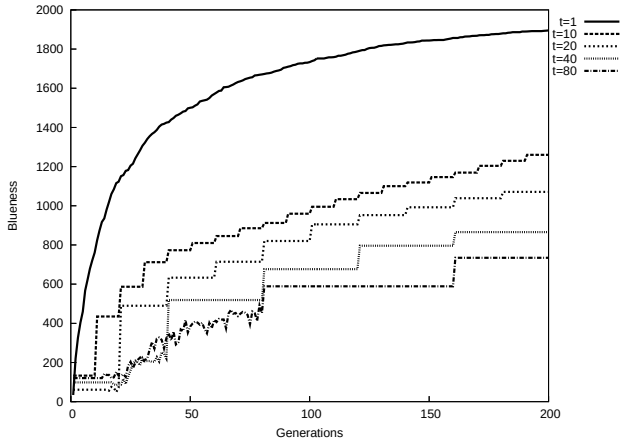

Figure 9: Degradation in the convergence (maximum) to blue UIs when using high t values.

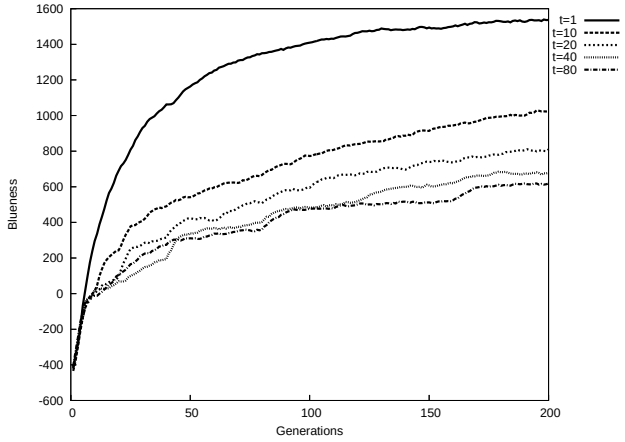

Figure 10: Degradation in the convergence (average) to blue UIs when using high t values.

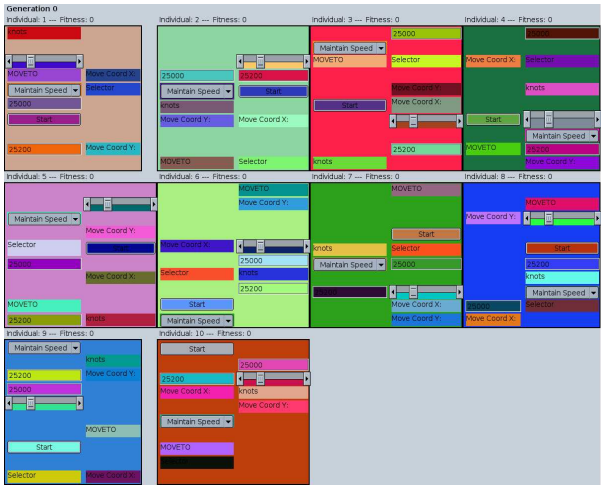

Figure 11: Displaying the best 10 individuals for user evaluation at generation 0.

Figures 11 and 12 show a subset consisting of the 10 best individuals in the population at generations 0 and 200, respectively. In generation 0, widgets start with random positions and random colors. In generation 200, the UIs shown all have blue widgets, which was the user assumed preference. The UIs at generation 200 both respect the metrics enforced on the objective evaluation: 1) Widgets should all have a similar shade of color, and 2) There should be a high contrast between foreground and background colors.

7. CONCLUSION AND FUTURE WORK

We have presented an IGA that combines both computable metrics, taken from guidelines of style, as objective heuristics and human subjective input to guide the evolution of UIs. Guidelines of style and a human sense of aesthetics are part of the UI design process, making our approach a suitable and promising application. We have also explored methods by which to reduce user fatigue in IGAs by 1) displaying a subset from the population that yields the best IGA performance, 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 first compared selection methods in order to find the one that yielded the best IGA performance. We found tournament selection's high selective pressure to yield better and more robust IGA results. Next we compared three subset display methods, where each method resulted in a different composition of the subset displayed for user evaluation. Our results indicate that displaying the best individuals for user evaluation results in better and faster convergence of the population, when compared to displaying a random subset and a subset consisting of the best and the worst individuals in the population. We also found that we can get reasonable convergence to well laid out blue (our preferred color) interfaces for small values of t. High values of t can reduce human fatigue considerably, but at the cost of increased noise in the subjective fitness landscape.

Clearly much work remains to be done. For example, we are

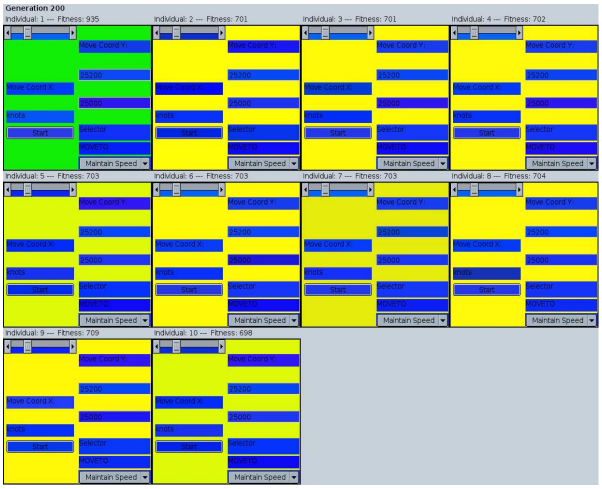

Figure 12: Displaying the best 10 individuals for user evaluation at generation 200.

considering changing the value of t over the course of evolution. One idea is to let the user to pick more often in early generations than in later generations. We may also want to give users more control. However, we believe that the work presented in this paper lays a good foundation for future research and development. First, widget encoding can be expanded 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. Furthermore, we would like to conduct user studies to assess the type of UIs that can be evolved, and to evaluate how well a human user can guide and bias the evolution of UIs. Last, we plan to further explore representations and genetic operators that can yield higher fitness individuals in less generations and with higher confidence intervals.

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.

8. 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.

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

[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 fitness. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1363–1370, New York, NY, USA, 2005. ACM Press.

[9] S. Microsystems. Java look and feel design guidelines, 2001.

[10] N. Monmarche, G. Nocent, M. Slimane, G. Venturini, and P. Santini. Imagine: a tool for generating html style sheets with an interactive genetic algorithm based on genes frequencies. In *Proceedings of the International Conference on Systems, Man, and Cybernetics*, pages 640–645. IEEE Computer Society, 1999.

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

[12] 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.

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

[14] XULPlanet. Xulplanet.com, 2006.