Learning Comparative User Models for Accelerating Human-Computer Collaborative Search

Gregory S. Hornby[1] and Josh Bongard[2]

[1] University of California Santa Cruz,
Mail Stop 269-3, Moffett Field, CA, USA
gregory.s.hornby@nasa.gov, http://idesign.ucsc.edu
[2] Morphology, Evolution and Cognition Lab
Department of Computer Science, University of Vermont
Burlington, VT, USA
jbongard@uvm.edu

Abstract. Interactive Evolutionary Algorithms (IEAs) are a powerful explorative search technique that utilizes human input to make subjective decisions on potential problem solutions. But humans are slow and get bored and tired easily, limiting the usefulness of IEAs. Here we describe our system which works toward overcoming these problems, The Approximate User (TAU), and also a simulated user as a means to test IEAs. With TAU, as the user interacts with the IEA a model of the user's preferences is constructed and continually refined and this model is what is used as the fitness function to drive evolutionary search. The resulting system is a step toward our longer term goal of building a human-computer collaborative search system. In comparing the TAU IEA against a basic IEA it is found that TAU is 2.5 times faster and 15 times more reliable at producing near optimal results.

Keywords: Evolutionary Design, Interactive Evolutionary Algorithm

1 Introduction

Interactive Evolutionary Algorithms (IEAs) are a powerful explorative search technique that utilizes human input to make subjective decisions on potential problem solutions [5, 9, 12, 13]. In traditional interactive evolution, a human user is presented with one or more candidate individuals being evolved for selection. The human user directly performs selection and then the favored individuals are selected for propagation of offspring into the next generation. Current examples of this work on the Web are Picbreeder [11] and EndlessForms [7], both of which are based on using neural networks to encode their designs.

Reliance on human input, however, induces a couple of major challenges. First, users suffer *user fatigue*: the quality and accuracy of human input greatly degrades with repeated prompts for input [13]. In addition, for typical non-interactive EAs, tens of thousands of evaluations are necessary to achieve interesting results, which is orders of magnitude more evaluations then can be

expected from a single user. Finally, humans are generally far slower at evaluating designs than computer software is. To make IEAs viable, some method must be developed for overcoming these limitations of human users.

Given the limited number of human interactions possible with an IEA, the IEA must make the most of what little data the user has provided. The approach we are analyzing is that of learning a model of the human user and using this model as the fitness function to drive the IEA. This idea came from the Estimation-Exploration Algorithm [2, 3], and was first applied to IEAs by Schmidt and Lipson [10]. The system we have developed is called **TAU**, for *The Approximate User*. Since the implementation used in this work is still in its preliminary stages, we deem it sufficient to show that a speed-up can be achieved on one domain and expect that this is likely to be extended by future improvements.

TAU is also a step toward a longer term goal of making a human-computer collaborative search system. Traditionally, either a person does all the work a computer search or optimization algorithm does all the work. With IEAs, a human is doing all the evaluations and the software is deciding what to be evaluated. We are working toward a system in which a human user is using their intelligence and experience to guide a search algorithm taking advantage of the computer's speed to perform most of the evaluations on its own.

The rest of this paper is organized as follows. In Section 2 we review related work in IEAs and user-modeling, followed by a description of the TAU algorithm in Section 3. To demonstrate that that TAU algorithm can accelerate IEA search we use a simulated human user, which is described in Section 4. The setup for the experiments is described in Section 5. Then in Section 6 we present experimental results in which we show that a TAU IEA is 15 times more likely and 2.5 times faster than a basic IEA at producing good results. Finally, we present our conclusions in Section 7.

2 Background

Quite recently there has been promising initial work toward addressing the user fatigue problem. One approach that has been used is to hardcode mathematical heuristics of aesthetics, and this has found some success for the interactive evolution of jewelry [14]. This system has several heuristics of beauty built in and reduces the amount of feedback needed from the user by two orders of magnitude. It can be thought of as a hybrid approach in which evaluations are partially done by an encoded fitness function – the heuristics of aesthetics – and partially done by the human user. Limitations of this approach are that it still requires a hard-coded fitness function and that results are somewhat dependent on it. Of interest are approaches in which there is no such dependence on a hard-coded fitness function.

An alternative to hard-coding heuristics is to build a model of the user's preferences with ideas from statistical Machine Learning. One system is to treat user feedback as inputs to a parameter estimation system [1]. This leverages the speed of existing statistical machine learning systems but is limited to pa-

rameterized design spaces. To move beyond parameterized encodings, another approach is to learn weights on grammatical rules for constructing a design [6]. While allowing for search through a topological space of designs, this does not scale to systems with large sets of rules, or which require large derivation trees to produce a design, or in which multiple sets of rules can produce acceptable solutions.

Our approach is to build a model of what the user wants to drive search, and to continuously learn and refine this model of the user's preferences concurrent with the design process. The idea behind our approach comes from prior work with the Estimation-Exploration Algorithm [2, 3], in which a coevolutionary system is used to evolve an *estimation* population, which evolves improvements to models of the hidden system, given pairs of input/output data obtained from the physical model(s) being approximated; and an *exploration* population, which evolves intelligent tests to perform on the hidden target system using the best models so far. In this case, the "hidden system" is the human user, of whom the computer is trying to build a model.

By having a computer model of the human's desires, this model can be used tirelessly to perform countless evaluations and thereby circumvent the limitations of having human users act as the fitness function. Already this approach has been tried on IEAs and it seemed to work well [10]. We have implemented our own version, which we have called The Approximate User (TAU), and here we are performing a more rigorous comparison of this approach against a basic IEA.

3 Overview of TAU

The TAU algorithm differs from a Basic IEA in that it uses a model of the User to perform its fitness evaluations rather than having the User manually evaluate each candidate solution. A user-model is built from a relations graph, which is a directed graph which stores every preference provided as input by the User. From this relations graph, modern machine learning techniques are used to train a model which can accurately match the user's preferences stored in this graph. This model of the user is then used as the evaluation function for a traditional optimizer to create a new set of solutions. Once a new set has been produced, a subset of them are presented to the user and the process repeats until a satisfactory result is produced.

An initial version of The Approximate User (TAU) algorithm for user-modeling has been implemented and an IEA augmented with TAU operates as follows:

1. Use the existing User Model to generate a set of candidate designs to present to the User. If the User Model is empty, generate random designs. This set of designs should be both good and diverse.
2. After the User has indicated their preference, update the relations graph by inserting the designs which were presented along with the User's preferences.
3. Create a newer User Model by training a classifier to correctly predict each relation in the relations graph.
4. Quit if a satisfactory solution has been produced.

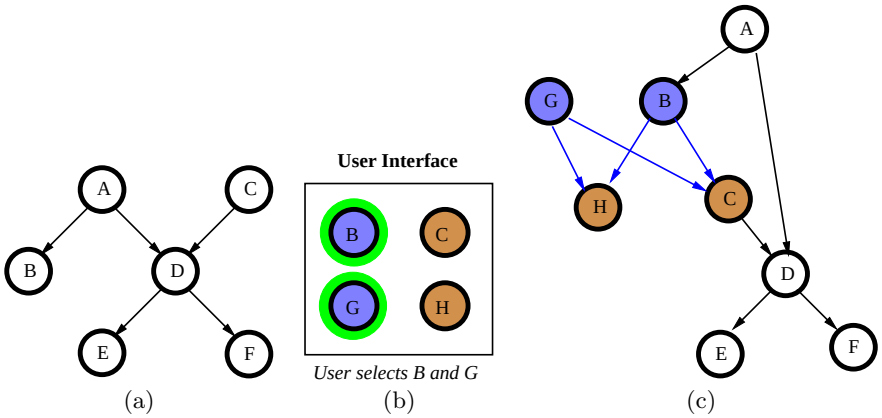


Fig. 1. An example of how an update of the relations graph works: (a) shows the current relations graph consists of candidate solutions A through F; (b) two of these "old" candidates are shown to the user as well as two new ones, G and H, of which the User selects B and G; (c) based on the User's selection, the relations graph is now updated.

For Step 1, we have implemented an Evolutionary Algorithm to create a set of designs. For a fitness function, the User Model is used to compare pairs of individuals and return which one is better.

The relations graph (Step 2) is a directed graph which contains the history of user selections and which can be queried to infer preferences which were not explicitly tested. Consider a population of candidate solutions (called *individuals* by those in the field of Evolutionary Computation) ind_A, ind_B and ind_C. If ind_A is better than ind_B and ind_C is better than ind_A, it follows logically that ind_C is also better than ind_B.

The relations graph in Figure 1(a) contains six individuals (`A` through `F`) and represents a subset of the entire relations graph that might exist after a couple of prompts to the user. There are nine relations that can be derived from this graph – indicating the User's preference from past queries – with the first five relations being the arrows that are shown. The rest are: `A` is better than `E`; `A` is better than `F`; `C` is better than `E`; and `C` is better than `F`. To continue growing the relations graph, each prompt to the user contains some individuals already in the graph, and some new individuals from the current population, Figure 1(b). Once the User has submitted their preference, this is used to update the relations graph, Figure 1(c). This updated relations graph is then used to train an updated version of the User-Model which can correctly predict these preference relations.

To implement the comparator model (Step 3) we are using the Fast Artificial Neural Network (FANN) library. ANNs are used because they have robust regression power with excellent interpolation and extrapolation characteristics [8]. Their classification output also corresponds to their statistical confidence in their prediction [4]. The basic structure of a comparator neural net is shown in

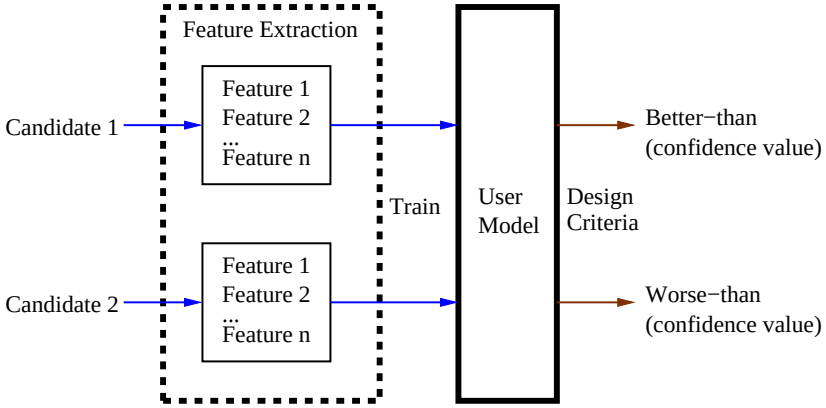


Fig. 2. The basic structure of how a User Model is used: feature are extracted from two candidate designs and are fed into the User Model, which then uses these features to predict which one is better. In the user-model training stage, existing relations between designs are used to train the User Model, in the design generation stage, the User Model is used as the fitness function to evaluate designs.

Figure 2. To improve performance, we are using an ensemble of five ANNs to create a User Model. Each ANN in the ensemble uses a randomly selected 75% of the available features to train on and to make its classification. Also, they each have a single hidden layer with a random number of hidden units (3 to 27) and are fully connected with weights randomly selected in the range of -0.1 to 0.1. Each ANN is trained using backpropagation for at most 50 iterations through the training data or until the training error is less than 0.001.

4 Simulating a Human User

Instead of performing comparison studies with real people, in this work a simulated person is used to drive the IEAs. The advantage of a simulated person is that experiments can be done quickly and as often as desired. The simulated person is implemented as a combination of a scoring function for how well candidate designs match a target shape and a method for using this to drive the IEA. This scoring function could be used as a fitness function in a regular EA but in this case it is used as the target function which the TAU algorithm is trying to learn. Since it is not directly used to drive the EA we choose not to call it a fitness function to avoid confusion.

Here, designs are constructed from a contiguous sequence of connected line segments. The design-scoring function used by the simulated human user takes as input the end-points of these line-segments – P_0, P_1, P_2, ... – and computes how close these line segments are to the target shape. For these experiments, the test problem used is that of creating a square out of a sequence of four connected line segments. The four line segments are contiguous and are encoded as a real-

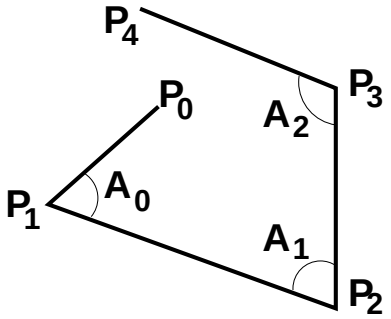


Fig. 3. An example drawing that is scored for its closeness to a square.

valued vector consisting of the five x and y coordinates for the 5 end points, P_0, P_1, ... P_4, Figure 3. This scoring function has three distinct components: a score on the gap between P_0 and P_4; a score on how similar in length each line segment is to each other; and a score on how similar each of the three angles, A_0, A_1 and A_2 is to a right angle. Each of these three scoring-components has a range of 0 to 1 and the overall score is a product of these three sub-functions.

When creating a square from the five points which specify four contiguous line segments the first and last points, P_0 and P_4, must be the same. To score for this result the gap score function, S_{gap}, divides the distance between points P_0 and P_4 by the sum of the lengths for each of the four line segments:

$$S_{gap} = 1 - \frac{\text{distance from } P_0P_4}{\text{sum of the length of all line segments}} \tag{1}$$

Another characteristic of a square is that all four sides have the same length. The score for this, the sub-function $S_{lengths}$ takes the length of each line segment and divides this by the average length of the four line segments. The ratio of the length of each side to the average length, L_i, is compared against the desired ratio, $L_{desired}$. For a square $L_{desired}$ is 1 for all line segments. To create a value between 0 and 1, the smaller of these two values is divided by the larger. This is done for each line segment and all four of these values are multiplied together:

$$S_{lengths}(L_{desired}) = \prod_{i=0}^{4} \begin{cases} \frac{|L_{desired}|}{|L_i|} \text{ if } L_i > L_{desired} \\ \frac{|L_i|}{|L_{desired}|} \text{ otherwise} \end{cases} \tag{2}$$

The third characteristic that is scored for is for the angles being right angles using the sub-function S_{angles}. Here, each angle A_i is compared against the desired angle, $A_{desired}$ and a value between 0 and 1 is computed similar to with $S_{lengths}$. For a square $A_{desired}$ is either $-\pi/2$ for all three angles or $+\pi/2$ for all three angles. The result for all three angles is multiplied together and returned:

$$S_{angles}(A_{desired}) = \prod_{i=0}^{2} \frac{|A_{desired}|}{|A_i - A_{desired}| + |A_{desired}|} \tag{3}$$

The overall score for how well a given shape matches the target shape is a product of the previous three sub-functions. For a square, the two options are for all line segments to have the same length and all three angles must be positive 90° turns ($\pi/2$) or they must all be negative 90° turns ($-\pi/2$):

$$S = S_{gap} \times S_{lengths}(1) \times max(S_{angles}(-\pi/2), S_{angles}(\pi/2)) \tag{4}$$

This approach to scoring how well a given shape matches a target shape is generic and can be used for scoring for different shapes by supplying the desired line-segment ratios and angles for the target shape. Because of symmetries, there are often multiple ways of producing a given shape from a sequence of line segments so the function returns the maximum of the different options.

In addition to the scoring function, the simulated user has a simple algorithm for driving the IEA to try and produce a desired drawing. The user interface for the IEA allows the user to: select and de-select designs; create a new set of candidate designs based on the current selections; discard all of the existing designs and create a new set of candidate designs based on the previous selections; and back-track to the previous set of designs. Roughly, the algorithm for the simulated human user requests a set of randomly generated designs until it finds one that scores higher than 0.25. It then iterates over selecting the top design, along with up to 2 other drawings which have a score within 10% of the top drawing. If the top drawing is not as good as the best drawing from the previous iteration, the algorithm requests an alternative set of candidates. If after three tries a new best drawing is not found, it backs up a level and tries again.

More precisely, the algorithm for the simulated user is as follows:

```
Level = 0
Set existing best to 0.
repeat                          ▷ Starting with randomly generated designs.
    Request new random designs.
    Score each design.
until Best score is < 0.25
repeat
    if New best is better than the previous best. then
        Level = Level + 1
        Tries[Level] = 0
        Select the best design.
        Select the next 2 best with a score within 10% of the best.
        Submit selections and request new designs.
        Score each design.
        continue.
    else
        Tries[Level] = Tries[Level] + 1.
        if Tries[Level] > 3 then
            Go back to previous generation.
            Level = Level - 1
            if Level = 0 then
```

```
22:                 Goto line 1.                    ▷ Move to initial state.
23:             end if
24:         end if
25:         Request an alternative set of design.
26:         Score each design.
27:     end if
28: until Best score is 1.
```

5 Experimental Setup

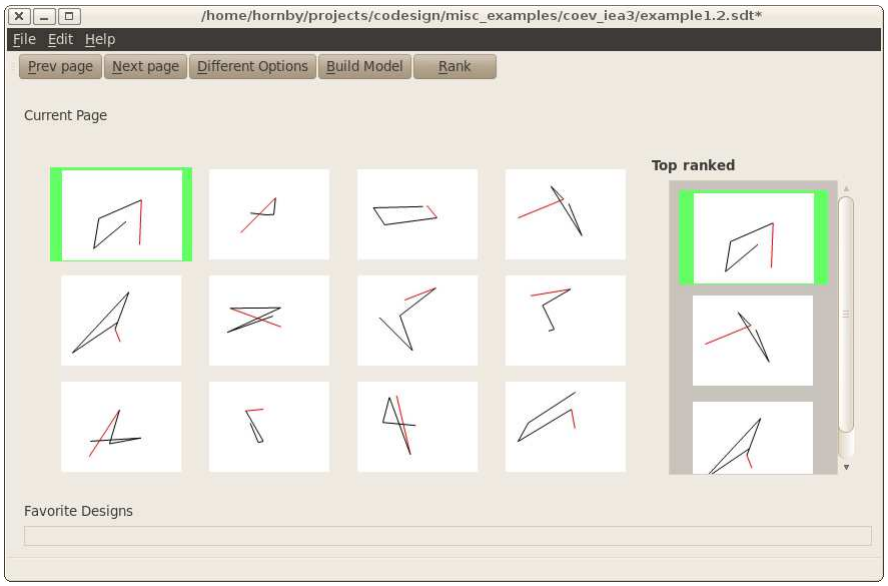
To demonstrate that concurrent construction of a user-model can accelerate search we compare the TAU IEA against a Basic IEA. The Basic IEA consists of having the designs selected by the user being the "parent" designs for creating the next set of designs to present to the user. The TAU IEA was described in Section 3. We configured the TAU IEA to use a population of five times the number of candidates shown to the user and to perform ten generations of evolution with the user model. Instead of using real people for our user-testing, both of these IEAs are driven by the simulated human user described in Section 4.

For these experiments, the test problem used is that of creating a square out of a sequence of four connected lines. An example of the application with a 3x5 grid of designs is shown in Figure 4. The top image (Figure 4(a)) shows an initial set of randomly generated line drawings and the bottom image (Figure 4(b)) shows the results after a few selection rounds using the TAU IEA.

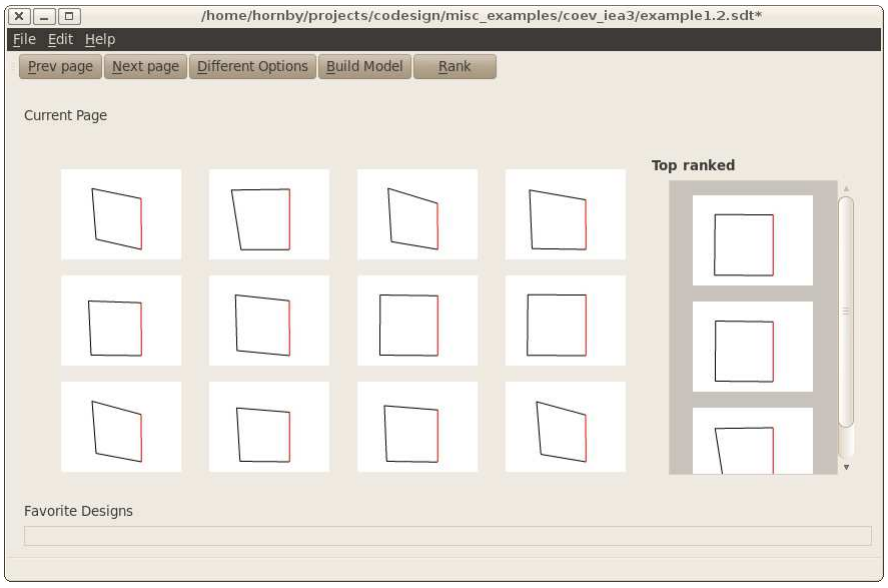
For the TAU IEA, the features for a given design consist of the ratio of the length of each line segment to the average length of line segments in that design and the angle between each line segment. Also, the angle of the first line segment to the horizon is included to indicate the overall orientation of the design and the size of the average line segment is included to provide an indicator for the overall size of the drawing.

6 Experimental Results

For these experiments 100 trials were run with three grid sizes using both the Basic IEA and the TAU IEA: 3x3, 3x4 and 3x5. Results of our experiments for all three grid sizes are shown graphically in Figure 5 and also in tabular form for the 3x5 grid in Table 1. The TAU IEA was about 15 times more reliable, achieving 98% of optimal solutions in 99% of its runs whereas the Basic IEA achieved this level in only 6% of its runs. Also, only the TAU IEA found optimal solutions (26% of the time), whereas the Basic IEA never found an optimal solution. The TAU IEA also achieved near-optimal results about 2.5 times faster: on average it achieved 98% of optimal in 30 selection iterations whereas it took the Basic IEA 84 iterations on the few runs in which it achieved this level of performance. In the best case, some trials with the TAU IEA achieved an optimal square design in 5-10 selection iterations.



(a)



(b)

Fig. 4. An example of using the system to interactively design a square. On the top are a set of initial, randomly-generated designs and on the bottom are candidates after a few selection iterations. Fitness values for the squares in Figure 4(a) are (from top row to bottom row): 0:0.257, 1:0.076, 2:0.144, 3:0.088, 4:0.044, 5:0.135, 6:0.114, 7:0.047, 8:0.110, 9:0.086, 10:0.116, and 11:0.046. Fitness values for the squares in Figure 4(b) are: 0:0.816, 1:0.853, 2:0.636, 3:0.681, 4:0.617, 5:0.716, 6:0.673, 7:0.922, 8:0.545, 9:0.806, 10:0.586, and 11:0.525.

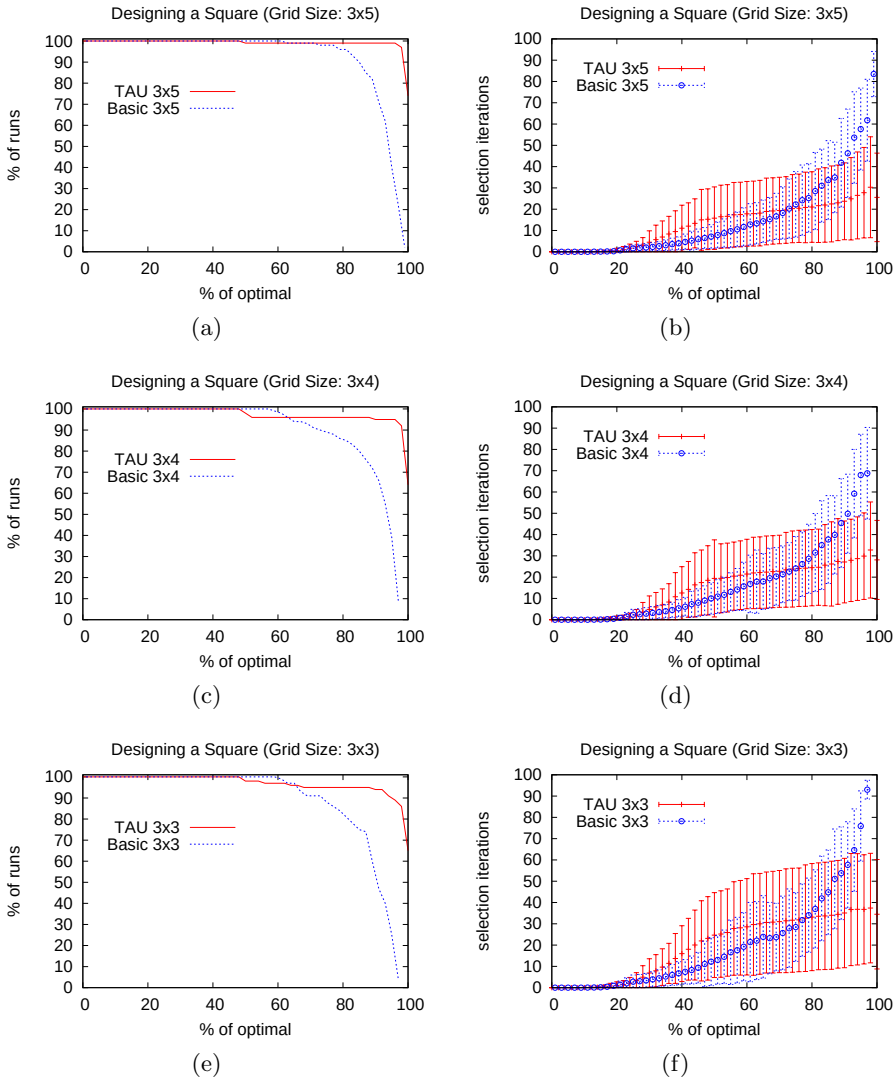


Fig. 5. This compares an IEA using the Basic IEA against the TAU (*The Approximate User*) algorithm on a simple design problem. Each row shows results for different grid sizes of options (3x3, 3x4 and 3x5). The left column shows the percentage of runs (out of 100) which achieve a given percentage of optimality on the design problem, and the right column shows the average number of selection iterations to reach a given percentage of optimality. Whereas the TAU IEA takes a few iterations to identify what the user wants and then quickly goes from there to optimal, it appears that the Basic IEA has a polynomial growth rate in the number of iterations needed to reach a given percentage of optimality.

	Basic IEA		User-Modeling IEA	
% of Optimal	Prob Success	Avg Iterations	Prob Success	Avg Iterations
80	0.96	26.8 ± 16.6	0.99	21.0 ± 16.6
90	0.74	43.3 ± 20.2	0.99	23.7 ± 18.0
95	0.4	57.6 ± 19.3	0.99	27.0 ± 20.5
98	0.08	66.2 ± 14.1	0.97	30.3 ± 23.7
99	0.02	83.5 ± 10.6	0.92	31.0 ± 25.5
100	0	–	0.74	25.5 ± 21.0

Table 1. Summary of results for evolving a square on the 3x5 grid. This shows that adding User Modeling increases the probability of success such that optimal designs can be reached almost three-quarters of the time whereas they were not achievable with a basic IEA. In addition, User Modeling is about 2 to 2.5 times faster at achieving near-optimal results.

Examining the graphs in Figure 5 shows an important difference between the algorithms in achieving a given level of design optimality. With the Basic IEA, the number of selection iterations needed to achieve a given level of optimality seems to be growing at a polynomial rate. Combine this with the Basic IEA's rapidly decreasing success rate and it suggests that the Basic IEA will not be able to scale to achieving good results on more challenging problems. In contrast, with the TAU IEA it seems that a few selection rounds are needed for the algorithm to learn a reasonable model of what the user wants and, after this, it has a fairly flat, linear growth in the number of selection rounds needed to achieve a given level of design optimality. Combine this with its much higher success rate at finding good solutions and it shows that the TAU algorithm has considerable promise as a way to accelerate search with an IEA.

7 Conclusion

While IEAs have shown much promise in enabling human users to direct evolutionary search, one of the main problems has been overcoming the limitations of human users: they are slow and the quality and accuracy of their input degrades rapidly over time. To overcome these limitations, we describe The Approximate User (TAU) algorithm. With TAU, the user's input is used to build a model of their preferences and this preference model is used to drive evolutionary search. With each user input, the user model is refined and is better able to guide evolutionary search to a desired result.

To validate the effectiveness of TAU, we developed an artificial human user to drive search using a basic IEA and with the TAU IEA. Experimental results show that search with TAU's user modeling is up to 15 times more likely to achieve a good result than without user modeling. In addition, TAU is 2.5 times faster on average and in the best case can be more than 10 times faster than the basic IEA.

We expect that by developing better approaches to modeling a human user's preferences that the TAU IEA can be made faster and able to scale to more difficult problems. In addition, the resulting system is a step toward our longer term goal of building a human-computer collaborative search system. That is, one in which both the human and the computer are simultaneously working on the problem and helping each other: the computer is fast and the human is good at getting past local optima.

Acknowledgments. This research was supported in part by the NSF Creative-IT grant 0757532 and DARPA M3 grant W911NF-1-11-0076. In addition, thanks to Grace Lin for her work on an early prototype.

References

1. Barnum, G.J., Mattson, C.A.: A computationally assisted methodology for preference-guided conceptual design. Journal of Mechanical Design 132 (2010)
2. Bongard, J., Lipson, H.: Nonlinear system identification using coevolution of models and tests. IEEE Transactions on Evolutionary Computation 9, 361–384 (2005)
3. Bongard, J., Lipson, H.: Automated reverse engineering of nonlinear dynamical systems. Proceedings of the National Academy of Sciences 104, 9943–9948 (2007)
4. Bridle, J.: Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In: Fogelman-Soulie, Herault (eds.) Neurocomputing: Algorithms, Architectures and Applications. NATA ASI Series, Springer (1990)
5. Caldwell, C., Johnston, V.S.: Tracking a criminal suspect through 'face-space' with a genetic algorithm. In: Booker, R.K.B.L.B. (ed.) Proc. of the Fourth Intl. Conf. on Genetic Algorithms. pp. 416–421. Morgan Kaufmann, San Mateo, CA (1991)
6. Campbell, M.I., Rai, R., Kurtoglu, T.: A stochastic graph grammar algorithm for interactive search. In: 14th Design for Manufacturing and the Life Cycle Conference. pp. 829–840. ASME (2009)
7. Clune, J., Lipson, H.: Evolving three-dimensional objects with a generative encoding inspired by developmental biology. Lecture Notes in Computer Science (2011)
8. Cybenko, G.: Approximations by superpositions of a sigmoidal function. Math. Contrl., Signals, Syst. 2, 303–314 (1989)
9. Dawkins, R.: The Blind Watchmaker. Harlow Longman (1986)
10. Schmidt, M., Lipson, H.: Actively probing and modeling users in interactive coevolution. In: et al., M.K. (ed.) Proc. of the Genetic and Evolutionary Computation Conference, GECCO-2006. pp. 385–386. ACM Press, Seattle, WA (2006)
11. Secretan, J., Beato, N., Ambrosio, D.B.D., Rodriguez, A., Campbell, A., Folsom-Kovarik, J.T., Stanley, K.O.: Picbreeder: A case study in collaborative evolutionary exploration of design space. Evolutionary Computation (2011)
12. Sims, K.: Artificial Evolution for Computer Graphics. In: SIGGRAPH 91 Conference Proceedings. pp. 319–328. Annual Conference Series (1991)
13. Takagi, H.: Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. In: Proceedings of the IEEE. pp. 1275–1296 (2001)
14. Wannarumon, S., Bohez, E.L.J., Annanon, K.: Aesthetic evolutionary algorithm for fractal-based user-centered jewelry design. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 22, 19–39 (2008)