

# Actively Probing and Modeling Users in Interactive co-Evolution

Michael D. Schmidt

Computational Synthesis Laboratory  
School of Electrical and Computer Engineering  
Cornell University, Ithaca NY 14853, USA

mds47@cornell.edu

Hod Lipson

Computational Synthesis Laboratory  
School of Mechanical and Aerospace Engineering  
Cornell University, Ithaca NY 14853, USA

hod.lipson@cornell.edu

## ABSTRACT

A major challenge in interactive evolution is extracting user preferences with minimal probing. We introduce an interactive multi-objective co-evolutionary algorithm that actively selects the most informative probes: We simultaneously co-evolve a population of candidate models that explain users' selection so far, and a population of candidate probes that cause the most divergence among model predictions, thereby elucidating model uncertainties (divergence). As progress is made, we begin selecting for probes with the highest expected outcome averaged among different models, thereby exploiting model certainties (consensus). In the evolution of pen stroke drawings, we find this technique to be highly effective at extracting preference models from very limited human interaction. Using only pair-wise preference questions, strategy and preference in pen stroke drawings are extracted in fewer than ten user probes. Our results show that the optimal questions to probe the user need not include drawings similar to the target drawing. Instead, the user models converge on trends in the user responses, thereby extrapolating strong preference for target drawings which the models are never actually trained to prefer.

## Keywords

Interactive Evolution, Fitness Prediction, Estimation-Exploration Algorithm, Active Learning

## 1. INTRODUCTION

Interactive evolution is a powerful explorative search technique that utilizes human input to make subjective decisions on potential problem solutions [6, 19]. The fitness landscape in each domain is thereby determined explicitly by the human user. Reliance on human input however, induces two major challenges: First, the time cost to collect human input greatly prohibits the discovery of complex solutions. Second, the quality and accuracy of human input greatly degrades with repeated prompts for input.

In this paper, we introduce a co-evolutionary algorithm to maximize the information obtained from the user and minimize the necessary interaction. We co-evolve a population of individual

solutions with a population of models that predict the user's preference. Co-evolved solutions are used both to maximize user preference, and also to probe the user in order to refine uncertainty in the user models, two objectives that are not necessarily aligned.

Our primary hypothesis is that intelligently probing the user for input based on their co-evolutionary behavior can generate more accurate user models than conventional modeling from very limited user interaction. New user probes must challenge and refine uncertainty and ambiguity in the model population.. We claim that – like the game of *20 questions* – the co-evolved individual solutions provide invaluable information to select these new user probes and find optimal user questions base on answers to previous probes.

To further simplify the interaction with the user, all probes for input ask the user for a single preference decision between two individual solutions (e.g. drawings). Correspondingly, our user model encoding is a comparator which predicts preference between pairs of solutions. This is also a very natural design for a human's preference. Decisions about how preferable an individual is must be made in the context of another individual. Although this mechanism cannot assign fitness values, evolution can utilize it effectively in selection and ranking.

The key goal in this paper is to extract accurate user models through minimal user interaction. Human users can only answer a small number of questions before becoming “numb” to prompts [10]. Therefore, we perform experiments which evolve pen stroke drawings using ten or fewer user prompts. The accuracy is evaluated based on the user's preferred target drawing.

We then compare our results with the interactive costs of a random search and a perfect local search algorithm. The random search comparison shows how effectively the exponential domain can be narrowed through co-evolution. The perfect local search comparison, where the user essentially draws their exact preference explicitly, shows how the interactive co-evolution of user models and probes algorithm can extract a specific preferred drawing from the user with fewer user probes and much simpler binary preference questions.

## 2. RELATED WORK

In this work we utilize genetic programming where a population of potential problem solutions is evolved in a Darwinian fashion [12]. We also utilize co-evolution, where two or more populations are evolved which directly or indirectly impact the evolution of each other in order to improve final solutions [9], [20]. In

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*Genetic and Evolutionary Computation Conference (GECCO) '06,  
June 25-29, 2006, Washington DC, USA.*

Copyright 2006 ACM 1-58113-000-0/00/0004.\$5.00.

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 favored individuals are selected for propagation of offspring into the next generation [6, 17, 19].

A new area of research involves partial human interaction. In these algorithms, the user provides constraints on the problem to narrow traditional evolutionary search [18]. A very promising area in interactive evolution research is agent based modeling. In this research, many user models are learned through interaction in order to study their collective behavior [2, 7]. In this paper, we investigate how to discover the most optimal user models with the minimal amount of user interaction through co-evolution.

In user preference modeling research, a candidate critique agent (CCA) is trained to estimate the favor for single individual solutions. The CCA learns weights on individual parameters to give exact fitness values [1, 14]. The weights introduce constraints on individuals and hence the CCA is very similar to partial interaction techniques.

Our co-evolutionary design is based on an estimation-exploration algorithm (EEA) setup. Unlike classical evolution, an EEA consists of three components: A population of estimators, a population of exploratory solutions, and a target hidden system [13, 20]. In this case the evolving problem solutions comprise the exploratory population, the ensemble of comparator models are the estimation population, and the abstract fitness landscape of the user's preference is the target hidden system.

### 3. FITNESS BY COMPARISON

#### 3.1 Comparison vs. Fitness

In this paper, we utilize a pair-wise preference model as the basic element to define the complex fitness domains explored in interactive evolution domains. Subjective selection must be done in the context of one or more other individuals to have meaning. We claim that the decision between two individuals is a fundamentally easier and more natural decision than assigning precise preference values to single individuals.

Which drawing is more interesting?

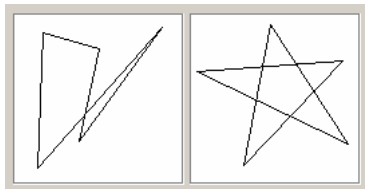


Figure 1. Example comparison between two pen drawings.

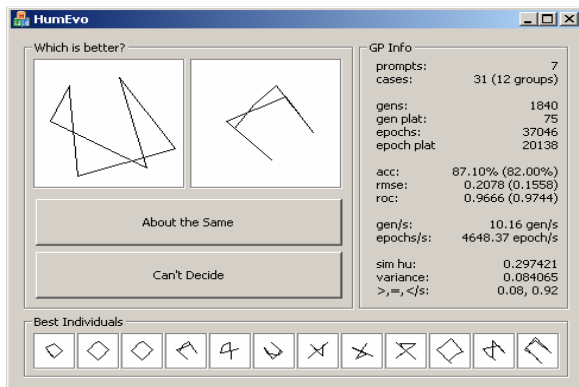


Figure 2. The user interface presented to the user.

Prompting the user to make individual comparison has proven very effective in other interactive evolution techniques such as incorporating human comparison into tournament selection [18]. In paper, we use the comparisons indirectly by co-evolving comparator models. The model population is used to evolve individuals and the user's input does not perform explicit selection.

### 3.2 User Interaction

#### 3.2.1 User Interface

The user interface presents two individuals to the user, and asks to click which one is more preferable, or optionally select them to be equally preferable, or for debugging purposes, request a new pair. An example of the screen shown to the user during runtime is shown in Figure 2.

The right-most panel displays performance and debugging information for expert users. The bottom-most panel shows the current best individuals in the population produced by the comparator model of the user.

### 3.3 Collecting Comparisons

#### 3.3.1 Relation Graphs

Comparator models define a fitness landscape through cascading many comparisons. For example, 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  $ind_B$ . Therefore, relations between each individual can be thought of as a directed graph.

It is important to note that comparison relations are prone to cyclical reasoning. For example,  $ind_A$  could be better than  $ind_B$ ,  $ind_B$  better than  $ind_C$ , and  $ind_C$  better than  $ind_A$ . We avoid this by storing all comparisons as an acyclic tree.

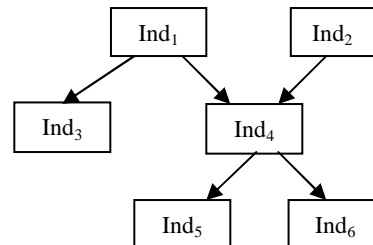


Figure 3. Example comparison relational graph of six individuals.

This tree in Figure 3 contains six individuals stored after five prompts to the user, where arrows indicate each better-than response. Note that there are nine relations that can be derived from this graph. The first five are the arrows shown. The next two are  $Ind_1$  better than  $Ind_5$  and  $Ind_1$  better than  $Ind_6$ . The last two are  $Ind_2$  better than  $Ind_3$  and  $Ind_2$  better than  $Ind_6$ .

To continue growing the relation tree, each prompt to the user contains one individual already in the tree, and one individual from the current population. The main advantage of this technique is it provides the potential for the number of known relations to grow at a binomial rate with user prompts in the ideal case, and a linear rate in the worse case.

#### 3.3.2 Minimizing User Prompts

A large problem in interactive evolution we address in this paper is the user becoming over worked. A user often becomes "numb"

to new prompts after a while, meaning they begin to put less thought into their selections and produce noisy data [10]. To reduce this effect, we take large strides to minimize the necessary interaction with the user. We apply active learning to maximize the potential information gained in each new prompt to the user. The goal when presenting a new prompt is for the response to refine the current comparator model and refine uncertainties and generality.

In the co-evolutionary setup, we train an ensemble of user models. The average of the ensemble then performs all selection in the individual population evolution [11]. We examine their separate predictions to measure their uncertainty and ambiguity. For example, different models in the ensemble may have strongly disagreeing predictions for different pairs. One may strongly predict a greater-than, while others may weakly predict a less-than. This is a case where feedback on a high variance relation will greatly improve the generality of the ensemble.

To select a new comparison prompt to the user, we consider two factors: the variance of the pair in the ensemble, and the strength of their predictions. In other words, we are interested pairs that have highly different predictions that are very strong and perhaps overfit in the model. These two parameters form a pareto front for prompt selection [8].

The prompt selection is done by generating all pairs of individuals with one from the current relation tree and one from the current population. These pairs are then considered on a bivariate graph defined by their variances and prediction strengths.

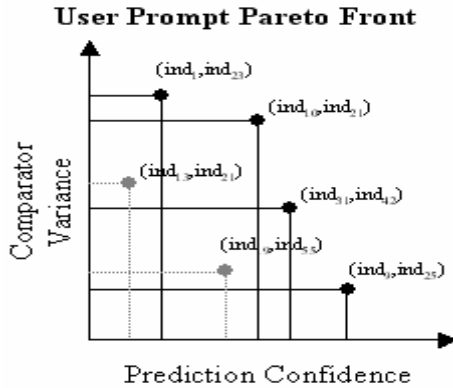


Figure 4. Example Pareto front plot of eight potential comparison pairs.

The pareto front consists of points that are non-dominated. This pareto front favors pairs that are both high variance and strong predictions. In other words, comparators in the ensemble have strong differing opinions on the predicted outcome. Points on the front with low confidence and high variance correspond to ambiguous pairs that are unexplored areas of the prediction domain. The high confidence and low variance predictions correspond to pairs that are well defined cases which can be refined to higher detail.

## 4. PREDICTING COMPARISONS

### 4.1 Basic Comparator

A simple comparator takes two individuals as input, and outputs three cases: better-than, equal, and worse-than. When comparing

individuals however, we are only interested in their better-than or worse-than outcome. Therefore, the interface of the comparison predictor takes two individuals, and outputs two confidence values for better-than and worse than. This structure is shown below in Figure 5.



Figure 5. The basic structure of an individual comparator user model.

The two outputs provide confidence values for each case. Their difference yields the final outcome in selection, and strength in user prompt calculation.

## 4.2 Neural Net Comparator

### 4.2.1 Network Structure

Neural nets are a natural fit for most comparison user modeling. They have robust regression power with excellent interpolation and extrapolation characteristics [5]. Their classification output also corresponds to their statistical confidence in their prediction. In other words, noisy samples or conflicting samples reduce prediction confidence but in general maintain prediction accuracy [3]. The basic structure of a comparator neural net is shown in Figure 6.

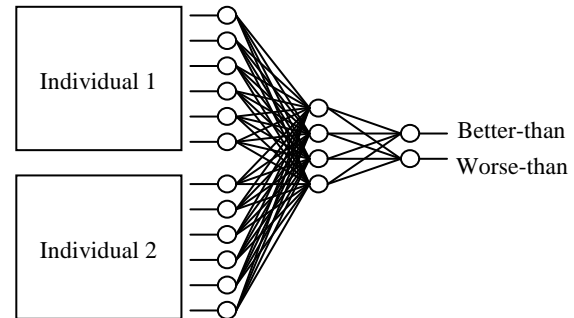


Figure 6. The structure of a neural network user model.

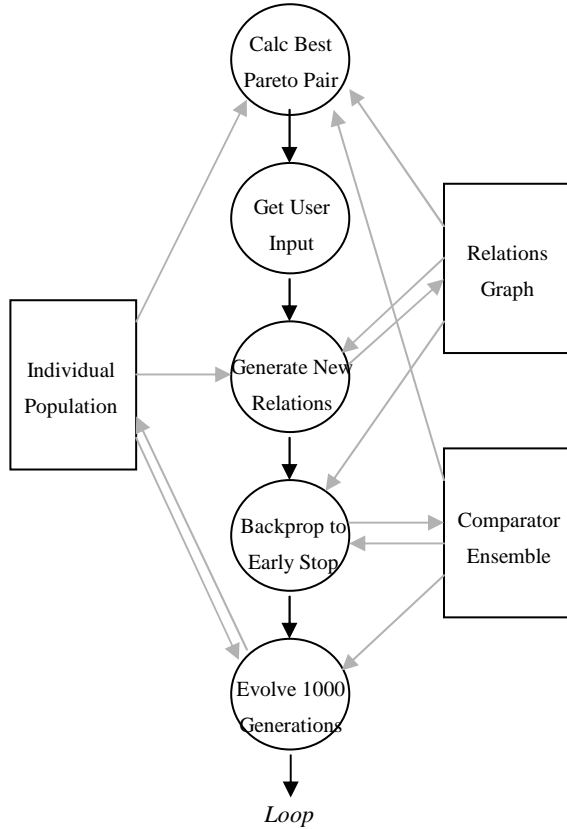
We use a network with a single hidden layer. We choose a number of hidden units sufficiently large enough for the domain. We then utilize early stopping on RMS with a eight-fold validation set to avoid over-fitting [4].

## 5. ALGORITHM SUMMARY

The interactive co-evolution of user models and probes algorithm presented in this paper maintains three essential components: the individual population, the relations graph, and the comparator model ensemble. The algorithm operations on these components in five stages: calculating the best comparison pair, requesting the user's input, generating new relations based on the feedback received, training the comparator model ensemble, and evolving the individual population using the comparator model.

The algorithm consists of five stages that operate on the individual population, the relations graph, and the comparator

model ensemble. The first stage chooses the best pareto pair of individuals to present to the user as in section 3.3.2. The algorithm then pauses for the user to respond. Next, all possible comparisons derived from the response when added to the relations graph are calculated. Then the comparator models are randomized and re-trained to their early stopping point. Finally, the individual population is evolved for one thousand generations before returning to stage 1.



**Figure 7. The comparator model based interactive evolution algorithm basic outline.**

The algorithm loops until the user is satisfied with the top ranked individual co-evolved by the comparator ensemble. As long as the user has a consistent preference, further iterations will stabilize and simply fine tune the preferred result.

## 6. EVOLVING DRAWINGS

### 6.1 Drawing Encoding

In this experiment we evolve drawings produced by a series of closed pen strokes. Each individual encodes each coordinate drawn to on a 32 by 32 pixel image. In our experiments we predefine the number of strokes for each drawing, but this could easily be evolved as well in future work.

The search space for these types of drawings increases exponentially with the number of pen strokes. The number of possible individuals for  $N$  pen strokes is calculated below.

$$\# \text{ of pixels} = 32 \cdot 32 = 1024$$

$$\# \text{ of possible individuals} = 1024^N$$

The large search space makes the discovery of preferable individuals an excellent application for an evolutionary algorithm.

### 6.2 Comparator Encoding

The comparison user model is encoded as a neural network as described in section 4.2. We choose to compare images based upon their image moments. Specifically, the zero and first order moments plus the first three Hu Invariant Moments, yielding a total of twelve inputs to each neural net.

**Table I. Neural Network Training Parameters.**

Parameter	Value
Hidden Layers	1
Hidden Units	32
Learning Rate	0.001
Momentum	0.5
Cross Validation Folds	8

The parameters for training the neural nets are summarized in Table I. These parameters are chosen empirically and they perform well for this experiment. Although they can influence comparison performance in extreme cases, they generally only impact the training time required.

### 6.3 Evolution Setup

We use standard genetic programming to evolve individuals in this experiment with the exception that all selection and fitness comparisons are performed using the trained comparator model ensemble. Individuals are not given explicit fitness values but are instead ranked using the averaged comparator ensemble. A summary of the evolution parameters is shown in Table II.

**Table II. Evolution Parameters.**

Parameter	Value
Population Size	64
Selection	Deterministic Crowding using the comparator ensemble
Mutation Probability	0.05
Crossover Probability	0.75

We use Deterministic Crowding for selection because it is a natural fit for a pair-wise comparator. The Deterministic Crowding method maintains population diversity through child-parent elitism and tends to follow multiple divergent pathways to the final solution [15]. This results in more variety in user prompt selection and better generalization of the comparator fitness landscape.

### 6.4 Discovering Square Shape Preference

#### 6.4.1 Square Drawings

The first experiment conducted tests the ability of algorithm to identify and infer a user’s preference for “square-like” drawings from initially random pen drawings. Each individual is encoded as

a series of four pen strokes. In this experiment we make two basic algorithm comparisons with random search and local search techniques. These comparisons gauge how effectively our algorithm reduces the interactive cost with the user to discover the target drawing.

### 6.4.2 Random Search Comparison

To quantify the difficulty of this problem we calculate the probability of finding a “square-like” drawing through random search. Note that the square is uniquely determined by two of its vertices. Given that a “square-like” drawing can have any orientation or size, and we allow the 3<sup>rd</sup> and 4<sup>th</sup> vertices to have some noise of four pixels, the probability and expected random individuals observed,  $T$ , to find such a drawing is calculated below.

$$P_{square} = \frac{(\#loc_1) \cdot (\#loc_2) \cdot (\#loc_3) \cdot (\#loc_4)}{(\#pixels) \cdot (\#pixels) \cdot (\#pixels) \cdot (\#pixels)}$$

$$P_{square} \approx \frac{(32^2) \cdot (32^2) \cdot (4^2) \cdot (4^2)}{(32^2) \cdot (32^2) \cdot (32^2) \cdot (32^2)}$$

$$P_{square} \approx \frac{4^4}{32^4} = \frac{1}{4096} \approx 0.024\%$$

$$\therefore E(T_{square}) \approx 4096$$

Therefore, the probability randomly generate a square is 0.39% and the expected iterations to encounter a square in a random search is 256.

### 6.4.3 Local Search Comparison

An alternative to interactive evolution is an interactive local search algorithm. In this technique, the user is given a random individual and asked to fine tune parameters individually. In pen stroke drawing this corresponds to adjusting the x and y coordinates of each pen stroke vertex to desired locations. This technique can be viewed as very simplistic partial interactive evolution where the user effectively constrains each parameter individually.

The local search algorithm we compare with can be thought of as a perfect algorithm for transforming a random drawing into the target drawing. The user is assumed to be an oracle that makes perfect choices to optimally tweak parameters with the minimum number of prompts.

Here we calculate a lower bound on the expected number of local adjustment steps necessary to shape a random pen stroke drawing to a square shape. Recall that a square is determined by two vertices. So we begin by calculating the expected diagonal of the square from two random points on the 32 by 32 pixel grid.

$$\langle \Delta x \rangle \langle \Delta y \rangle = \frac{\sum_{x_1=0}^{31} \sum_{x_2=0}^{31} |x_1 - x_2|}{32^2} = \frac{10912}{32^2} \approx 10$$

$$\langle d \rangle = \sqrt{\langle \Delta x \rangle^2 + \langle \Delta y \rangle^2} \approx 15$$

Next, the expected mean of these two points, and all random vertices is the center point of the grid ( $x=15, y=15$ ). This means that the two remaining points are expected to be at the center of

the final square. Hence, they must each move half a diagonal,  $\langle d \rangle / 2$ .

In the easiest case, these points move only along an individual axis. In the worst case, they move diagonally, stepwise on the grid. The expected number of steps per vertex is calculated below.

$$\langle steps \rangle = \frac{1}{16} \cdot \sum_{x=0}^{15} x + \left[ \sqrt{15^2 - x^2} \right] \approx 18$$

Therefore, the lower bound on the total number of expected local updates to the x and y coordinates of the two remaining vertices is approximately 36.

Note that this is a lower bound approximation on the expected minimum number of user prompts required for a perfect local search algorithm to form a pen stroke drawing of a square at any orientation. An expert user is required to identify a desired diagonal and then isolate the remaining vertices accordingly.

### 6.4.4 Results

Figure 8 shows a standard run for a square drawing using the comparator user model interactive evolution algorithm. The user has a specific strategy to prefer shapes with parallel sides and

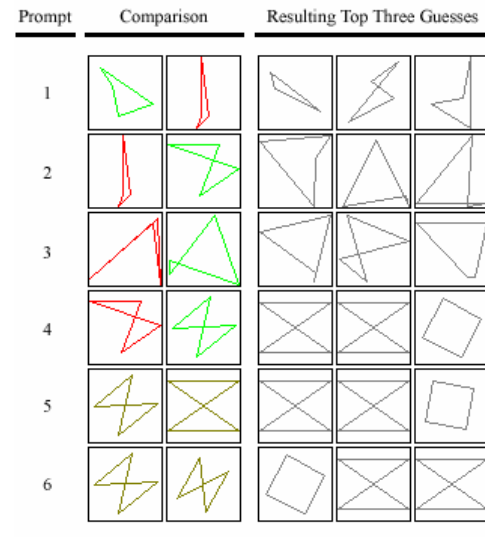
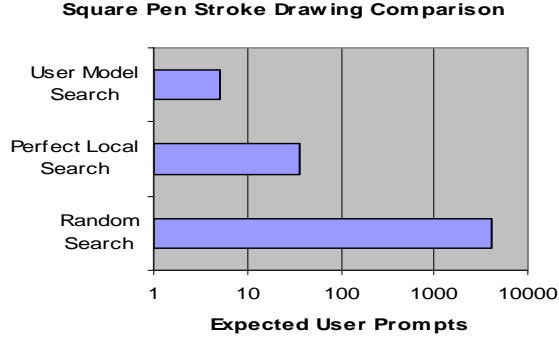


Figure 8. The prompts given to the user and the resulting top three guesses over six iterations.

right angles consistent with a square. The user’s preferred drawing for each comparison is shown in green, non-preferred drawings are shown in red, and drawing deemed to be equivalent are shown in dark yellow. This is a very representative run since the initial random prompts do not include any box-like drawings. If by chance they do, the runs tend to converge on a box shape immediately.

First, notice that none of the comparisons shown to the user involve a square. Based upon the several non-square comparisons, the comparator model is able to infer that the user is likely to prefer a square shape. If in fact the user was not knowingly seeking squares but giving feedback on some unknown preference, the algorithm would predict and identify box-like solutions they are likely to favor.

The algorithm successfully found a square in its top three guesses after four user prompts, and ranked a square as its top guess after six prompts. This is a vast improvement over the random search which is expected to require 4096 user interactions before finding an approximate square.



**Figure 9. The number of user prompts expected between the compared algorithms to find a square shape.**

Figure 9 shows the comparison with the user comparator model algorithm with the perfect local search and random search algorithms. The logarithmic scale shows that the user comparator model makes significant improvement over the perfectly performing local search and vastly reduces the search cost over random search.

## 6.5 Discovering Star Shape Preference

### 6.5.1 Star Drawings

In this experiment, the algorithm evolves drawings with six pen strokes. Here we evaluate the algorithm ability to identify a preference for star-shaped drawings. In this experiment we make two basic algorithm comparisons with random search and local search techniques. These comparisons gauge how effectively our algorithm reduces the interactive cost with the user to discover the target drawing.

### 6.5.2 Random Search Comparison

To quantify the difficulty of this problem we calculate the probability to find an approximate star shape randomly. To approximate the number of star shapes possible, we split the drawing space into six regions: a center where no vertices can be, and fix surrounding regions for each vertex, all of equal area. Note that a five point star is uniquely determined, by region in this case, but three of its vertices. The probability and expected random individuals observed,  $T$ , to find an approximate star shaped drawing is calculated below.

$$P_{star} = \frac{(\#loc_1) \cdot (\#loc_2) \cdot (\#loc_3) \cdot (\#loc_4) \cdot (\#loc_5)}{(\#pixels) \cdot (\#pixels) \cdot (\#pixels) \cdot (\#pixels)}$$

$$P_{star} \approx \frac{\left(\frac{5}{6} \cdot 32^2\right) \cdot \left(\frac{2}{6} \cdot 32^2\right) \cdot \left(\frac{1}{6} \cdot 32^2\right)^3}{(32^2) \cdot (32^2) \cdot (32^2) \cdot (32^2) \cdot (32^2)}$$

$$P_{star} \approx \frac{5}{3888} \approx 0.128\%$$

$$\therefore E(T_{star}) \approx 777.6$$

Therefore, the probability to randomly generate a star drawing individual is 0.128%, and the expected number of random generations to encounter one is 777.6. Note that this is an easier search than finding a square since we are not requiring specific angles, so the acceptable star shapes could be quite deformed.

### 6.5.3 Local Search Comparison

Following the same logic as in section 6.4.3, we now approximate a lower bound on the expected user necessary to form a star pen stroke drawing.

As found earlier, the expected distance between two random vertices on the 32 by 32 pixel grid is approximately 15. To simplify this calculation we make an approximation that the expected locations of the other three vertices lay at the center of the star. The radius of this star with line length 15 is easily calculated to be approximately 8. The expected number of steps, moving both in x and y is calculated below.






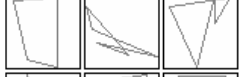








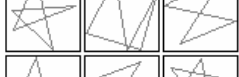


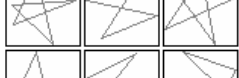



$$\langle steps \rangle = \frac{1}{9} \cdot \sum_{x=0}^8 x + \left[ \sqrt{8^2 - x^2} \right] \approx 9$$

Therefore, the lower bound on the total number of expected local updates to the x and y coordinates of the three remaining vertices is approximately 27.

Note that this is a lower bound approximation on the expected minimum number of user prompts required for a perfect local search algorithm to form a pen stroke drawing of a square at any orientation. An expert user is required to identify an optimal starting edge and then isolate the remaining vertices accordingly.

### 6.5.4 Results

Figure 10 shows a standard run to evolve an approximate star shaped drawing. The user has a general strategy when answering

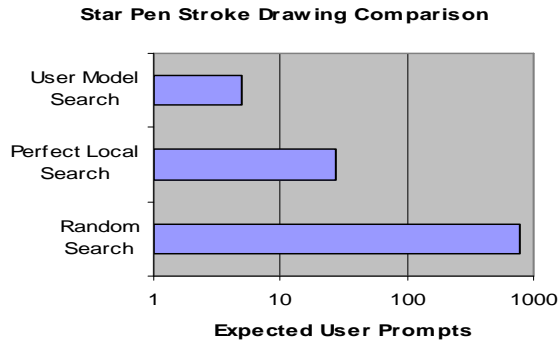
Prompt	Comparison	Resulting Top Three Guesses	
1			
2			
3			
4			
5			
6			
7			

**Figure 10. The prompts given to the user and the resulting top three guesses over seven iterations.**

comparison prompts to prefer shapes with multiple sharp pointed corners such as a star shape has. The user’s preferred drawing for each comparison is shown in green, non-preferred drawings are shown in red, and drawings deemed to be equivalent are shown in dark yellow. This is a very representative run since the initial random prompts are very dissimilar to the target star shape. In the unusual case, early prompt may resemble the target shape by random chance, resulting in nearly immediate convergence to the star shape.

Notice in Figure 10 that a very well formed star is derived as the top predicted shape after five user prompts. No prior prompts required a star shape. Instead the comparator model inferred the star shape as an optimum solution given the user responses favoring shapes with multiple sharp edges.

The next three prompts answered by the user are shown to demonstrate the comparator model is stable and continues to favor the consistent star-like shapes with further input. Therefore, the algorithm has likely converged on the star shape preference.



**Figure 11. The number of user prompts expected between the compared algorithms to find the target star shape.**

Figure 11 shows the comparison with the user comparator model algorithm with the perfect local search and random search algorithms. The logarithmic scale shows that the user comparator model makes significant improvement over the perfectly performing local search and vastly reduces the search cost over random search.

## 7. INFERRING A FITNESS LANDSCAPE

### 7.1 Discovering Clock Drawing Preference

In this experiment we want to visualize the fitness landscape being learned by the comparator model. To do this, we modify our pen stroke drawing individual to have only a single pen stroke originating from the center of the drawing area. The search space of the individuals is now only a single coordinate,  $x$  and  $y$ . We can then display relative fitnesses for all possible individuals in a 3D surface.

For this experiment, the single pen stroke encoding is considered to be a clock hour hand. The user is then asked to prefer clocks drawn where the time is closer to some time of day that they prefer. The user for the experiment chooses to prefer clock drawings where the hand points to either 3:00 or 7:00. Therefore, this experiment has two equally favour global maxima solutions.

### 7.2 The Fitness Landscape of a Comparator

To determine if the comparator accuracy learns the dual solutions in this experiment we need some way to calculate a fitness landscape from a comparator model. Therefore, we need to define a fitness calculation for a single individual using the binary comparator.

For this experiment, the search space for all clock drawings is the total number of pixels,  $32^2$ . For this small search space, it is feasible to list all possible drawings and compare them with a particular drawing. We calculate an effective fitness by taking the average confidence of better-than and worse-than outcomes when compared with all possible  $32^2$  clock drawings.

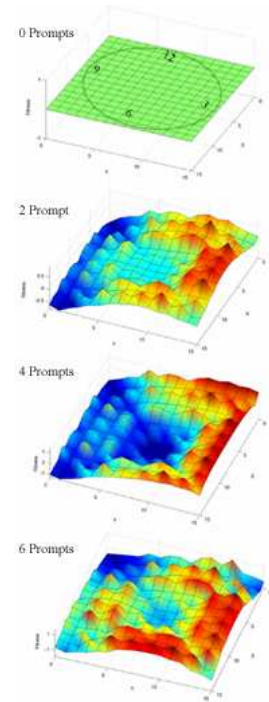
$$Fitness = \frac{1}{32^2} \sum_x \sum_y Better(Ind(x, y)) - Worse(Ind(x, y))$$

In this expression, Better() and Worse() are the two confidence outputs of the basic comparator model described in section 4.1.

### 7.3 Landscape Results

Figure 12 shows four fitness landscapes of the comparator model over six user prompts. At zero prompts, the fitness landscape is entirely flat.

After two prompts, the comparator model has identified a single preferred clock time region, 3:00. It also strongly disfavors clock times between 7:00 and 11:00. After two further prompts, the comparator now favors times between 1:00 and 5:00. This appears to be an intermediary stage where the comparator has learned to favor clock hands of a minimum length. This is clear by noticing the fitness landscape shows very low fitness for short clock hands near the center, and high fitness for hands at the extremities between 1:00 and 5:00. Finally, after six prompts the comparator has successfully identified the two preferred regions near 3:00 and 7:00.



**Figure 12. The clock time fitness landscapes calculated over six user prompts.**

Notice that the fitnesses on the  $xz$ -plan resemble an arch which peaks at approximately the clock hand 3:00 position. Correspondingly, the  $yz$ -plane fitnesses exhibit the same phenomenon although slightly less accurately near the 7:00 clock hand position. This final resulting landscape shows the comparator has inferred a very friendly fitness landscape for the evolutionary search. Very gradual gradients exist near the target clock hand locations that should be easily descended.

It’s interesting to note that the comparator landscape also exhibits a few other medium fitness clock hand areas such as near the 9:00 position. This is a weakly favored region that the algorithm shows

some probability for preference in. Since these are local maxima, evolution is likely to focus on these areas. The pareto criteria hence is likely to prompt the user to refine these regions in additional iterations.

## 8. CONCLUSION

Experiments in this paper show the interactive co-evolution of user models and probes algorithm to be highly effective at extracting preference models and resulting solutions from very limited human interaction. Using only pair-wise preference questions, strategy and preference in pen stroke drawings are extracted in fewer than ten user probes.

In comparison to the perfect local search algorithm, where the user essentially draws their exact preference explicitly, the interactive co-evolution of user models and probes algorithm requires roughly ten times fewer total user input. Furthermore, the prompts to the user are presented in much simpler binary preference questions.

Finally, our results show that the optimal questions to probe the user need not include drawings similar to the target drawing. Instead, the user models converge on trends in the user responses, thereby extrapolating strong preference for target drawings which the models are never actually trained to prefer.

## 9. REFERENCES

- [1] Bishop, I. "Comparing regression and neural net based approaches to modeling of scenic beauty". *Lands. Urban Plann.* 34 (1996), pp. 125–134, 1996.
- [2] Bonabeau, E. "Agent-based modeling: methods and techniques for simulating human systems". *Proc. Nat. Acad. Sci. USA* 99, 7280-7287, 2002.
- [3] Bridle, J. "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition." In *Neurocomputing: Algorithms, Architectures and Applications*, Fogelman-Soulie, and Hérault (eds.). NATO ASI Series, Springer, 1990.
- [4] Caruana, R., S. Lawrence, and L. Giles. "Overfitting in Artificial Neural Nets Trained with Backpropagation, Conjugate Gradient, and Early Stopping," *Neural and Information Processing Systems*, Vol. 13 MIT Press, 2001.
- [5] Cybenko, G. "Approximations by Superpositions of a Sigmoidal Function," *Math. Contrl., Signals, Syst.*, Vol. 2, pp. 303-314, 1989.
- [6] Dawkins, R. "The Blind Watchmaker," W. W. Norton, New York, 1987.
- [7] Ecemis, I, E. Bonabeau, and T. Ashburn. "Interactive estimation of agent-based financial markets models: modularity and learning". p.1897-1904, GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation, 2005.
- [8] Ficici, S., and J. Pollack. "Pareto optimality in coevolutionary learning." In *Advances in Artificial Life: 6th European Conference (ECAL 2001)*, 2001: 316-327, 2001.
- [9] Hillis, W. "Co-evolving improves simulated evolution as an optimization procedure." In Langton, C. et al. (Eds.), *Artificial Life II*. Addison Wesley, 1992.
- [10] Hollnagel, E. "Human Reliability Analysis—Context and Control." Academic Press, Inc., New York, NY, 1993.
- [11] Kohavi, R. "A study of cross-validation and bootstrap for accuracy estimation and model selection. In: C.S. Mellish, Editor, *Proceedings IJCAI-95 Montreal, Que.*, Morgan Kaufmann, Los Altos, CA, pp. 1137–1143, 1995.
- [12] Koza, J. "Genetic Programming: On the Programming of Computers by Means of Natural Selection." Cambridge, MA: The MIT Press, 1992.
- [13] Larrañaga, P., and J. Lozano. "Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation." Norwell: Kluwer Academic Publishers, 200.
- [14] Linden, G., S. Hanks, and N. Lesh. "Interactive assessment of user preference models: The Automated Travel Assistant" *Proceedings of the 6th International Conference on User Modeling*, 1997.
- [15] Mahfoud, S. "Niching Methods for Genetic Algorithms." Ph.D. Thesis, University of Illinois at Urbana-Champaign, 1995.
- [16] Parmee, I. "Improving problem definition through interactive evolutionary computation," *Journal of Artificial Intelligence in Engineering Design, Analysis and Manufacture-Special Issue: Human-computer Interaction in Engineering Contexts* 16(3), 2002.
- [17] Poli, R. "Genetic programming for image analysis." In Koza, J. R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L., editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 363–368, Stanford University, CA, USA. MIT Press, 1996.
- [18] Poli, R., S. Cagnoni. "Genetic programming with user-driven selection: Experiments on the evolution of algorithms for image enhancement" presented at the 2nd Annu. Conf. Genetic Programming, J. R. Koza et al., Eds., San Francisco, CA, 1997.
- [19] Takagi, H. "Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation." *Proc. IEEE* 89: 1275-1296, 2001.
- [20] Zykov, V., J. Bongard, and H. Lipson "Co-evolutionary variance guides physical experimentation in evolutionary system identification." In *The 2005 NASA/DoD Conference on Evolvable Hardware*, 2005.