Combating User Fatigue in iGAs: Partial Ordering, Support Vector Machines, and Synthetic Fitness

Xavier Llorà, Kumara Sastry, David E. Goldberg, Abhimanyu Gupta, Lalitha Lakshmi

Illinois Genetic Algorithms Lab
University of Illinois at Urbana-Champaign
104 S. Mathews Ave., Urbana, IL 61801, USA

{xllora,kumara,deg,agupta21}@illigal.ge.uiuc.edu

ABSTRACT

One of the daunting challenges of interactive genetic algorithms (iGAs)—genetic algorithms in which fitness measure of a solution is provided by a human rather than by a fitness function, model, or computation—is user fatigue which leads to sub-optimal solutions. This paper proposes a method to combat user fatigue by augmenting user evaluations with a synthetic fitness function. The proposed method combines partial ordering concepts, notion of non-domination from multiobjective optimization, and support vector machines to synthesize a fitness model based on user evaluation. The proposed method is used in an iGA on a simple test problem and the results demonstrate that the method actively combats user fatigue by requiring 3–7 times less user evaluation when compared to a simple iGA.

Categories & Subject Descriptors

I.2.6 [Artificial Intelligence]: Learning–Parameter Learning, H.5.2 [Information Interfaces and Presentation]: User Interfaces–Theory and methods.

General Terms

Algorithms, Design, Human Factors, Theory.

Keywords

Interactive Evolutionary Computation. Human Evaluation, Synthetic Fitness, Support Vector Machines.

1. INTRODUCTION

An important concept used in genetic and evolutionary algorithms [16, 9] to implement the survival-of-the-fittest mechanism and to distinguish between good and bad solutions is the notion of *fitness*. Unlike traditional search methods, the fitness measure can be relative, and can be an *objective* measure consisting of a mathematical equation, model, or a computation. It could also be a *subjective* measure involving human evaluation, or even be *coevolved* in a co-operative or competitive environment. Evolutionary algorithms (EAs) that use human evaluations for determining quality of candidate solutions are called *interactive* evolutionary algorithms (iEAs) [26].

Unlike in evolutionary algorithms with objective fitness measures, one of the daunting challenges of iEAs is effective methods of combating user fatigue. Even for moderately-sized problems, iEAs may require a few hundred to a few thousand fitness evaluations, which is highly improbable—oftentimes even impossible—for users to evaluate. This places a premium on a variety of *efficiency-enhancement techniques* [10], particularly *evaluation relaxation* [23]. In evaluation relaxation schemes, the computationally costly, but accurate function evaluation is replaced by a cheap, but less accurate surrogate function. A serious stumbling block in developing surrogate fitness functions in iEAs is the absence of computable fitness function. Additionally, the user evaluation is relative and user preference might change over time. Hence, existing evaluation-relaxation methods fall short and cannot effectively model user fitness function.

Therefore, the purpose of this paper is to propose an evaluation-relaxation scheme for actively combating user fatigue. Specifically, we address two critical issues: (1) the lack of a computable fitness, and (2) lack of systematic way for modeling user-decision process. The proposed method consists of three major components: (1) *partial ordering:* The qualitative decisions made by the user about relative solution quality is used to generate partial ordering of solutions, (2) *induced complete order:* The concepts of non-domination and domination count from multi-objective evolutionary algorithms [9] to induce a complete order of the solutions in the population based on their partial ordering, and (3) *Surrogate function via support vector machine:* The induced order is used to assign ranks to the solutions and use them in a support vector machine (SVM) to create a surrogate fitness function that effectively models user fitness.

Inducing a synthetic fitness allows us to exploit different time scales involved in an iEA. Computing a synthetic fitness value may take several orders of magnitude less time than a typical user evaluation. This significant disparity in evaluation times allows us to evolve the population with surrogate fitness alone and to show the user potentially high-quality solutions. The proposed approach provides the following key benefits: First and foremost, it allows us to use larger

GECCO'05, June 25–29, 2005, Washington, DC, USA.

population sizes—which otherwise is at least improbable, if not impossible—required to obtain high-quality solutions [10]. It also speeds-up the iGA process and thereby avoids user fatigue. Additionally, the method presents potentially high-valued solutions for user evaluation, thus avoiding user frustration. Finally, the model can be regarded as a useful insight into the user preferences.

This paper is organized as follows. Section 2 reviews the research on iGAs. The surrogate model of the synthetic subjective fitness is presented in section 3. This section presents how a hypothetical fitness can be synthesized combining partial ordering concepts, multiobjective optimization ideas, and support vector machines. The proposed modeling is integrated and tested in real web application. Section 4 briefly reviews such application and summarizes the results obtained using it when compared to plain iGAs without reconstructed subjective fitness optimization. Finally, discussions and conclusions are presented in section 5.

2. INTERACTIVE GENETIC ALGORITHMS

Dawkin's Blind Watchmaker program [7] and the Faceprints system developed at New Mexico State University [4] are two early examples of iGAs. For example, in Faceprints, the system replaces the role of a human sketch artist in evolving the faces of criminal suspects from witness recollection. Faces are encoded as binary strings where subcodes represent different facial features (nose type, mouth type, hair type, etc.). Each full chromosome maps to a face and the population of chromosome is presented to the human critic who is asked to determine how close the face resembles that of the criminal. This subjective ten-point scale is used to drive the evolution of subsequent generations of faces, and in a relatively short time, the GA arrives at a reasonable facsimile of the correct face.

The use of interactive genetic algorithms allow the fusion of human and computer efforts for problem solving [26]. However, putting the evaluation process into the hands of a user sets up a different scenario when compared to normal optimization. Takagi [26] presented a review of research efforts related to the iGAs challenges. These research areas included: (1) discrete fitness value input method, (2) prediction of fitness values, (3) interface for dynamic tasks, (4) acceleration of iGAs convergence, (5) combination of evolutionary and non-evolutionary computation, (6) active intervention, and (7) theoretical research. These areas may be reorganized in five main elements that iGAs need to address on their road to effective solution:

Clear goal definition: A precise description of the goal is key element to help the user engage a successful innovation process. A clear definition helps evolve high-quality solutions. Moreover, if such definition is maintained along the run, the user's task gets greatly simplified.

Impact of problem visualization: The solutions presented to the user need to be understandable and comparable. If the visualization is too complex, the user will be overwhelmed with details. If there is no simple way to qualitatively compare solutions, the user may not be able to make a proper decision. If such qualitative comparison is not easy, the quality of the user evaluations will decrease and greatly penalize the performance of the iGA.

Lack of real fitness: iGAs lack a quantitative fitness function analogous to the one used in traditional GAs. The qualitative nature of the evaluation process usually leads to scenarios where the user is asked to provide solution rankings or relative evaluations among a selected subset of solutions.

Fatigue: User fatigue is a critical element to produce high-quality solutions. Long times until convergence lead to tedious and demanding attention periods on the user side. Fatigue becomes the main reason of an early stop of the iGA process and, hence, leads to low-quality solutions.

Persistence of user criteria: The user can change his evaluation criteria along an iGA leading to a noisy evaluation scenario. The user criteria may drift along the run, leading to a dynamic optimization scenario. Methodologies for helping the user to maintain the persistence of his evaluation criteria along the iGA run is a key element.

Our work focuses on the lack of a real fitness function and how we can take advantage of the relative evaluations provided by the user to reduce the user fatigue. Section 3 presents how such goal can be achieved by means of subjective fitness reconstruction. Section 4 presents how such reconstructed subjective fitness may be later combined with a traditional GA to combat the user fatigue.

3. SYNTHESIZING A SUBJECTIVE FITNESS

Simple GAs may require hundreds or thousands of function evaluations before finding good quality solutions, but most human evaluators providing feedback to an interactive GA will tire far short of best convergence. A number of techniques can be adopted to overcome this difficulty, but they largely fall in the area of GA efficiency enhancement [10]. Techniques such parallelization, time continuation, evaluation relaxation, and hybridization are useful approaches to combat evaluation fatigue by reducing the convergence time [10, 24]. However, even efficiency enhancements assume the existence of an objective fitness function that can be computed without any human intervention. Such requirement can not be fulfilled under the iGAs paradigm. Moreover, the qualitative nature of the evaluations in an iGA also fosters other issues that need to be addressed. For instance, if an iGA run involves two different parallel evaluators then the iGA needs to deal with two different subjective evaluation criteria. Thus, no assumption should be made about the coherence of such criteria across the parallel evaluators—multimodal and multiobjective approaches may be needed to deal with such situations.

3.1 The Lack of an Objective Fitness

Several techniques have been used to collect the user subjective evaluations. Ranking of solutions, quality measurement of the solution in a 0 to 100 scale, or selecting a subset of the promising solutions from a pool are a few alternatives proposed since iGAs origins [26]. Each of these methods

have their advantages and disadvantages, combining different evaluation and selection schemes.

Tournament selection [12] is one of the most widely used ordinal selection schemes. In tournament selection a specified number of individuals, s, is selected from the current population of size n. The best individual out of the s individuals gets a copy in the mating pool. The selection of the s individuals can be performed either *with replacement* or *without replacement*. In selection with replacement the individuals selected from the current tournament are candidates for other tournaments. On the other hand, if selected without replacement, the individual–once selected–are not candidates for other tournaments.

Such a selection scheme replaces the need for collecting a numeric evaluation from the user. Instead, an iGA under a tournament scheme with replacement displays a set of individuals, and the user pick the best solution out of the s candidates shown. The simplest situation for the user is when $s = 2$. For instance, given two solutions s_1 and s_2 in a given tournament, the user picks one–s_2 for instance. This decision can be safely casted into fitness fitness terms; s_2 is better than the one of s_1, hence, $f(s_2) > f(s_1)$. Such interpretation introduces a partial ordering among solutions.

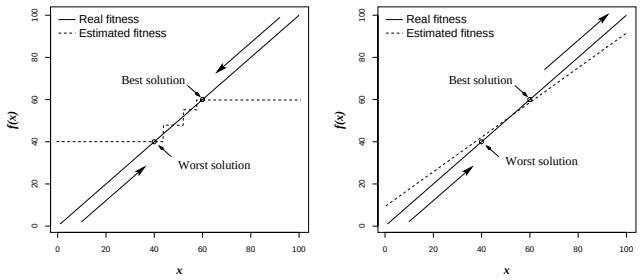
3.2 Properties of a Synthetic Fitness

Any attempt to synthesize the subjective fitness based on the partial order provided by the user requires, at least, two properties to be satisfied: (1) *fitness extrapolation*, and (2) *order maintenance*. The first property—*fitness extrapolation*—requires that the synthetic fitness provide meaningful inferences beyond the boundaries of the current partial order provided by the user. This property guarantees that any attempt to optimize such surrogate fitness will provide a useful guess of the user future evaluations. The second property—*order maintenance*—guarantees a synthetic fitness consistent under a tournament selection scheme. Thus, order maintenance guarantees that a synthetic fitness is accurate if it maintains the partial ordering given by the user decisions. Such assumptions allow us to use low-cost high-error models, as long as they maintain the proper ordering.

3.3 Surrogate Models and Subjective Fitness

Nearest-neighbor models have been commonly used as synthetic fitnesses[26]. However such models may not satisfy the properties mentioned above. Such models assume that given a set of user-evaluated solutions, the fitness of a new solution may be estimated as the fitness of the closest evaluated solution—using some distance metric. Other approaches average the fitness of the k nearest solutions. It is easy to prove that such schemes may easily violate both of the previous properties.

No fitness extrapolation may be provided by such nearest-neighbor surrogate model heuristic. Figure 1(a) illustrates such situation with a simple OneMax counterexample. Any finite randomly generated population will be bounded by two solutions—the ones with the best and worst evaluated fitness. The nearest-neighbor heuristic explained above produces a truncation of the synthetic fitness. If we optimize such a synthetic fitness, the final population may converge (in the best case) around the best solution in the nearest neighbor model. This happens because all the potential new good solutions (top-right portion of the line in figure 1(a)) are equally evaluated, not providing any useful insight on unseen good candidate solutions. Moreover, weighted nearest-neighbor heuristics may also violate the partial ordering property. A simple counter example may be given using OneMax again. Let's assume we are using a 3-nearest-neighbor equally-weighted heuristic, and the three user-evaluated closest solutions to a new unseen solution $x = 41$ are: $s_0 = (40, 40)$, $s_1 = (42, 42)$, and $s_2 = (45, 45)$. The estimated fitness $\hat{f}(x)$ is $\hat{f}(x) = \frac{1}{3}(40+42+45) = 42.33$. Using such result we may infer that $f(42) < \hat{f}(41)$, violating the inherent ordering of the problem.



(a) Fitness extrapolation and order maintenance can not be guaranteed under a nearest-neighbor surrogate model heuristic.

(b) Fitness extrapolation and order maintenance can be guaranteed under a regression surrogate model heuristic.

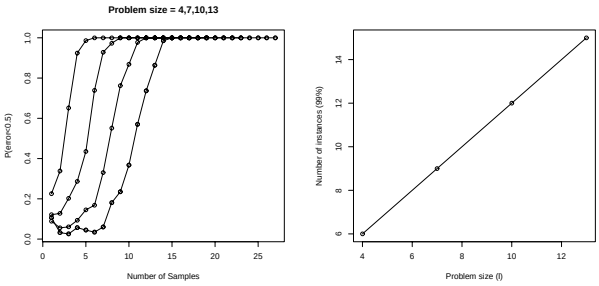
Figure 1: Any surrogate model of the subjective fitness need to guarantee two properties: fitness extrapolation and order maintenance.

Another surrogate model commonly used as a synthetic fitness function is based on regressors (for instance, neural networks [26]). In our work we use a generalized regression model, the ε-insensitive regression (ε-SVM). A detailed description of ε-SVM may be found in literature somewhere else [6, 25]. A synthetic fitness based on ε-SVM using a linear kernel easily satisfies the fitness extrapolation and order maintenance properties. Figure 1(b) illustrates how such fitness extrapolates beyond the boundaries provided by the user-evaluated solutions thanks to the hyper-plane adjustment done by ε-SVM [6, 25]. Moreover, even with a high-regression error, a ε-SVM guarantees the proper ordering of solutions under a tournament selection scheme. For instance, given the example of figure 1(b), any hyper-plane ($y = m \cdot x + c$) with a positive slope $m > 0$ guarantees the proper ordering of the synthetic fitness. Using the previous example, $s_0 = (40, 40)$ and $s_1 = (42, 42)$,

$$s_0 < s_1 \longrightarrow \hat{f}(s_0) < \hat{f}(s_1) \text{ iff } (m \cdot x_{s_0} + c) < (m \cdot x_{s_1} + c) \quad (1)$$

holding when $m > 0$. Hence, high-error surrogate models are synthetic fitness candidates, as long as the training examples are properly ordered.

The minimal number of user evaluations needed to obtain a surrogate ε-SVM subjective fitness model that guarantees both properties is show in figure 2. Given a problem with no dependencies among decision variables, OneMax, we sampled at random a set of evaluated solutions to synthesize a fitness by training a ε-SVM regressor. Figure 2(a) displays the probability of perfect global ordering by



(a) Probability of perfect ordering by a ε-SVM as a function of the number of sampled solutions.

(b) Number of samples required to build a perfect ordering using a ε-SVM as a function of the problem size.

Figure 2: A surrogate subjective fitness based on ε-SVM requires a number of training examples that grow linearly on the problem size ℓ when no linkage among decision variables are present.

the ε-SVM fitness as a function of the number of sampled instances. Results are averaged among 1,000 independent runs. Given a problem size ℓ, there is a minimal number of samples required to infer a property-complainant synthetic fitness. Moreover, such minimal number of solutions grow as $\ell + 2$, as figure 2(b) shows. Such results are totally consistent with the theoretical PAC bounds of ε-SVM which requires at least the number of training examples to grow linearly with the number of dimensions of the problem [6, 25].

These results hold for problems where no linkage [27, 13] among the decision variables is present. If linkage among variables is present, then a polynomial kernel may be required by the ε-SVM to satisfy the fitness extrapolation and order maintenance properties. The analysis of such model under linkage conditions is left as further work to be addressed.

3.4 A Simple Synthesis Exercise

The surrogate models reviewed in the previous section make a basic assumption: the partial order of user evaluations can be translated into a global numeric value. If such a value is available, then ε-SVM provides an efficient synthetic fitness. Thus, we need a method to turn the user-provided partial ordering among the different solutions into a numeric fitness value to synthesize the training examples of ε-SVM. In a tournament selection of size $s = 2$, a user is asked to provide an answer to the question of which of the two choices is better. The outcome of such question may be: the first shown, the second shown, or both are equal or the user was unable to decide. Let's assume the illustrative example where eight solutions need to be evaluated as presented in figure 3.

The tournament ordering presented in figure 3 guarantees that the partial order introduced by the user evaluations produces a connected graph G. Such graph $G =< V, E >$ represents the partial evaluation order representing the solutions as vertex in V, and the pair-wise comparison among

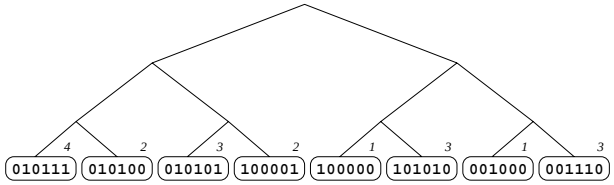
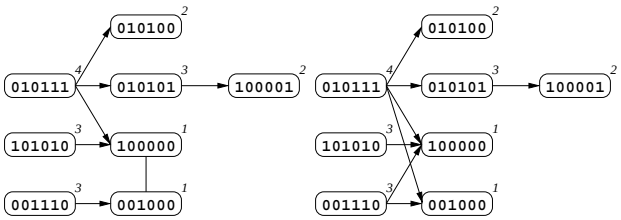


Figure 3: Eight randomly chosen individuals from a population, are grouped in seven different tournaments {(010111, 010100), (010101, 100001), (100000, 101010), (001000, 001110), (010111, 010101), (100000, 001000), (010111, 100000)}. The number beside each node simulates the objective function in the user mind.



(a) Partial ordering graph provided by the comparisons provided by the user based on the tournaments of figure 3

(b) Equivalent partial order graph where equality relations have been replaced by the proper greater than relations.

Figure 4: Partial ordering graph provided by the comparisons provided by a user. Direction on the arrows indicates *greater than* relations. When no direction is provided, equality is assumed.

individuals (greater, lesser, or equal) as edges in E. The partial ordering graph provided by the user may be undirected (equal evaluations are allowed), however, such graph can be easily turned into a directed graph as figure 4 shows. The directed graph is obtained by replacing the equal (undirected edges) by the proper greater or lesser relations (directed edges), as figure 4 shows.

To create a synthetic fitness we propose a heuristic based on the partial ordering provided by user evaluations and the Pareto dominance concept [21] of multiobjective optimization [5, 8]. A global ordering measure may be computed using a heuristic based on two dominance measures, δ and ϕ. Let's define $\delta(v)$ as the number of different nodes present on the paths departing from vertex v. Analogously, $\phi(v)$ is defined as the number of different nodes present on the paths arriving to v. Since the partial order is a directed graph, such mapping has an interesting property. If v appears more than once in a path (trial) of $\delta(v)$ or $\phi(v)$, then a cycle in such graph exists. Thus, due the greater and lesser relations, a contradiction on the user evaluations is identified. We will not discuss this issue further, leaving it as part of the further research. The work presented breaks such cycles by removing the oldest evaluation (edge) in the path thus breaking the cycle. Table 1 computes $\delta(v)$ and $\phi(v)$ given the graph presented in figure 4(b).

Table 1: Estimation of the global ranking based on the dominance measure. This data presented on the table uses the partial order presented on figure 4(b). Given a vertex v, the number of dominated vertexes $\delta(v)$ and dominating vertexes is computed. Using this measures, the estimated fitness may be computed as $\hat{f}(v) = \delta(v) - \phi(v)$. The estimated ranking $\hat{r}(v)$ is obtained by sorting based on $\hat{f}(v)$.

v	$f(v)$	$r(v)$	$\delta(v)$	$\phi(v)$	$\hat{f}(v)$	$\hat{r}(v)$
010111	4	1	5	0	5	1
010100	2	3	0	1	-1	4
010101	3	2	1	1	0	3
100001	2	3	0	2	-2	5
100000	1	4	0	3	-3	6
101010	3	2	2	0	2	2
001000	1	4	0	3	-3	6
001110	3	2	2	0	2	2

The estimated fitness of a given solution (vertex) v may be computed as $\hat{f}(v) = \delta(v) - \phi(v)$. Intuitively, the more solutions a solution v dominates (is greater than), the greater the fitness. Otherwise, the more solutions dominate (are greater than) a solution v, the smaller the fitness. The final global estimated ordering $\hat{r}(v)$ is obtained sorting by $\hat{f}(v)$, as shown in table 1. The estimated ranking introduces some spurious relations inside common ranks. Once the global ordering is computed (estimated ranking $\hat{r}(v)$), such ordering may be used to train a ε-SVM. By optimizing such a synthetic fitness we may obtain a look-a-head on candidate solutions to be evaluated by the user.

4. USER FATIGUE AND SYNTHETIC FITNESS FUNCTIONS

The next step toward fighting the user's fatigue requires to combine the synthetic fitness function with an iGA. This section presents how we integrated both elements. The proposed method was tested by an independent user with no experience neither in the problem to be solved nor the usage of iGAs. The results are analyzed and compared to the theoretical simple GA models of population sizing, convergence time, number of function evaluations, and speed up.

4.1 Active Interactive Genetic Algorithms

The existence of a synthetic fitness allow us to actively use such model to combat user fatigue. The proposed iGA (we call it *active* iGA) may be summarized as shown in table 2. Given the theoretical framework used so far (OneMax and ε-SVM using a linear kernel), the compact GA is a suitable option to optimize the synthetic fitness. The compact genetic algorithm (cGA) [15], is one of the simplest estimation of distribution algorithms [22, 17]. Similar to other EDAs, cGA replaces traditional variation operators of genetic algorithms by building a probabilistic model of promising solutions and sampling the model to generate new candidate solutions. The probabilistic model used to represent the population is a vector of probabilities, and therefore implicitly assumes each gene (or variable) to be independent of the other. Specifically, each element in the vector represents the proportion of ones (and consequently zeros) in each gene

Table 2: Algorithmic description of the active iGA.

1.	Create an empty directed graph G.
2.	Create 2^h random initial solutions ($\mathcal{S}$ set).
3.	Create the hierarchical tournament set $\mathcal{T}$ using the available solutions in $\mathcal{S}$.
4.	Present the tournaments in $\mathcal{T}$ to the user and update the partial ordering in G.
5.	Estimate $\hat{r}(v)$ for each $v \in \mathcal{S}$.
6.	Train the surrogate ε-SVM surrogate synthetic fitness based on $\mathcal{S}$ and $\hat{r}(v)$.
7.	Optimize the ε-SVM synthetic fitness using the cGA.
8.	Create a $\mathcal{S}'$ set with 2^{h-1} new different solutions, where $\mathcal{S} \bigcap \mathcal{S}' = \emptyset$, sampling out of the probabilistic model evolved by cGA.
9.	Create hierarchical tournament set $\mathcal{T}'$ with $2^h - 1$ tournaments using 2^{h-1} solutions in $\mathcal{S}$ and 2^{h-1} solutions in $\mathcal{S}'$.
10.	$\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}'$
11.	$\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}'$
12.	Go to 4 while not converged.

position. The probability vectors are used to guide further search by generating new candidate solutions variable by variable according to the frequency values.

The compact genetic algorithm consists of the following steps:

1. *Initialization:* As in simple GAs, where the population is usually initialized with random individuals, in cGA we start with a probability vector where the probabilities are initially set to 0.5. However, other initialization procedures can also be used in a straightforward manner.

2. *Model sampling:* We generate two candidate solutions by sampling the probability vector. The model sampling procedure is equivalent to uniform crossover in simple GAs.

3. *Evaluation:* The fitness or the quality-measure of the individuals are computed.

4. *Selection:* Like traditional genetic algorithms, cGA is a selectionist scheme, because only the better individual is permitted to influence the subsequent generation of candidate solutions. The key idea is that a "survival-of-the-fittest" mechanism is used to *bias* the generation of new individuals. We usually use tournament selection [12] in cGA.

5. *Probabilistic model updating:* After selection, the proportion of winning alleles is increased by $1/n$. Note that only the probabilities of those genes that are different between the two competitors are updated. That is,

$$p_i^{t+1} = \begin{cases} p_i^t + 1/n & \text{If } x_{w,i} \neq x_{c,i} \text{ and } x_{w,i} = 1, \\ p_i^t - 1/n & \text{If } x_{w,i} \neq x_{c,i} \text{ and } x_{w,i} = 0, \\ p_i^t & \text{Otherwise.} \end{cases} \quad (2)$$

Where, $\mathbf{x}_{w,i}$ is the i^{th} gene of the winning chromosome, $\mathbf{x}_{c,i}$ is the i^{th} gene of the competing chromosome,

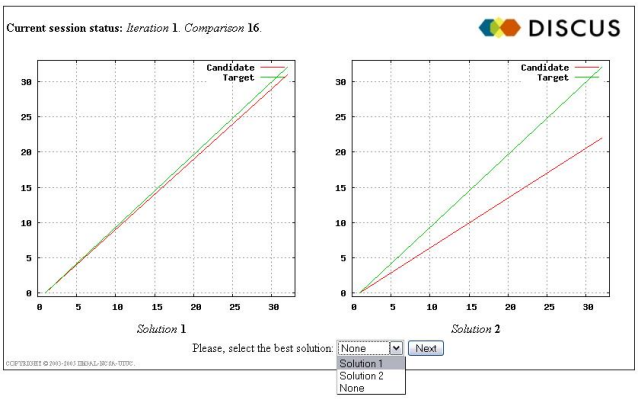


Figure 5: A simple interface for the active IGA test.

and p_i^t is the i^{th} element of the probability vector—representing the proportion of i^{th} gene being one—at generation t. This updating procedure of cGA is equivalent to the behavior of a GA with a population size of n and steady-state binary tournament selection.

6. Repeat steps 2–5 until one or more termination criteria are met.

More details are available elsewhere [15, 14]. However it is important to note that the cGA is operationally equivalent to the order-one behavior of simple genetic algorithm with steady state selection and uniform crossover [15]. Therefore, the theory of simple genetic algorithms can be directly used in order to estimate the parameters and behavior of the cGA.

4.2 Some Experimental Results and Analysis

We create a simple web interface to test the performance of the *active* iGA proposed in table 2. The interface was designed to minimize the interface bias providing a clear goal definition, a simple problem visualization, and a clear relative comparison method to help maintain the user criteria. Figure 5 shows a snapshot of a real iGA session conducted using the proposed *active* iGA.

In order to collect unbiased results, a user with no experience on evolutionary methods or interactive optimization was selected to perform the experimentation. The underlying problem to be solved was OneMax given different problem sizes $\ell = \{4, 8, 12, 16, 20, 24, 28, 32\}$. The user did not know the underlying problem to be solved. For each problem size, 10 independent runs were conducted by the user. Such results were collected, averaged, and later analyzed and compared against the the usage of a theoretical iGA based on the simple GA.

We begin our analysis with the scalability of selectorecombinative genetic algorithms followed by the scalability of the active iGA. Two key factors for predicting the scalability and estimating the computational costs of a genetic algorithm are the convergence time and population sizing. Therefore, in the following subsections we present facet-wise models of convergence time and population sizing.

4.2.1 Population-Sizing Model

Goldberg, Deb, & Clark [11] proposed population-sizing models for correctly deciding between competing BBs. They incorporated noise arising from other partitions into their model. However, they assumed that if wrong BBs were chosen in the first generation, the GAs would be unable to recover from the error. Harik, Cantú-Paz, Goldberg, and Miller [14] refined the above model by incorporating cumulative effects of decision making over time rather than in first generation only. Harik et al. [14] modeled the decision making between competing BBs as a gambler's ruin problem. Here we use an approximate form of the gambler's ruin population-sizing model [14]:

$$n = \frac{\sqrt{\pi}}{2}\frac{\sigma_{BB}}{d}2^k\sqrt{m}\log m\sqrt{\left(1+\frac{\sigma_N^2}{\sigma_f^2}\right)}, \qquad (3)$$

where k is the BB size, m is the number of BBs, d is the size signal between the competing BBs, σ_{BB} is the fitness variance of a building block, and σ_N^2 is the variance of the noise, and σ_f^2 is the fitness variance. The above equation assumes a failure probability, $\alpha = 1/m$.

4.2.2 Convergence-Time Model

Mühlenbein and Schlierkamp-Voosen [20] derived a convergence-time model for the breeder GA using the notion of *selection intensity* [3] from population genetics. Thierens and Goldberg [28] derived convergence-time models for different selections schemes including binary tournament selection. Bäck [1] derived estimates of selection intensity for s-wise tournament and (μ, λ) selection. Miller and Goldberg [19] developed convergence-time models for s-wise tournament selection and incorporated the effects of external noise. Bäck [2] developed convergence-time models for (μ, λ) selection. Even though the selection-intensity-based convergence-time models were developed for the OneMax problem, Miller and Goldberg [18] observed that they are generally applicable to additively decomposable problems of bounded order.

Here, we use an approximate form of Miller and Goldberg's [19, 10] convergence-time model:

$$t_c = \frac{\pi}{2I}\sqrt{m}\sqrt{1+\frac{\sigma_N^2}{\sigma_f^2}}. \qquad (4)$$

where I is the selection intensity, and $\ell = mk$ is the string length. For binary tournament selection, $I = 1/\sqrt{\pi}$. A detailed derivation of the above equation and other approximations are given elsewhere [10, 23].

Using equations 3 and 4, we can now predict the scalability, or the number of function evaluations required for successful convergence, of GAs as follows:

$$n_{\text{fe,GA}} = \frac{\pi^2}{4}\frac{\sigma_{BB}}{d}\sqrt{k}\log m\cdot\left(1+\frac{\sigma_N^2}{\sigma_f^2}\right)\cdot 2^k\cdot m. \qquad (5)$$

4.2.3 Active iGA analysis

The first difference of that using a synthetic fitness function is the population size requirements. Figure 6(a) presents the population sizing of a simple GA and the one of the iGA. As it can be seen, the iGA requires a population size that, at least, grows linearly. Such requirement is the result of using a ε-SVM with a polynomial kernel which require at least as many training examples as dimensions (ℓ in the iGA case) as figure 2(b) showed. Moreover, the active iGA population is also constrained by the three tournament

structure, given a problem size ℓ the population size is forced to grow $2^{\lceil \log_2(\ell) \rceil}$.

Figure 6(b) compares the convergence time of a active iGA to the a simple iGA. Based on the results presented in figure 2(b), the theoretical convergence time of the active iGA with the proper population sizing, should be constant. The empirical results show in figure 6(b) support such assumption. Combining the population sizing and the convergence time, the number of functions evaluations of the active iGA should grow linear. However, due to the three structure of the tournament evaluation used, a staircase effect may be appreciated in figure 6(c).

Finally, figure 6(d) shows the speedup achieved using the active iGA respect to a simple iGA. The results show how with the active use of a simple low-cost high-error synthetic fitness function we were able to achieve speedups ranging from 3 up to 7 times. The instability of the speedup is the result of the constrains on the population sizing (fee figure 6(a)). However, being able to cut down the total number of evaluations on such ratios proved to be an effective method for combating the user fatigue.

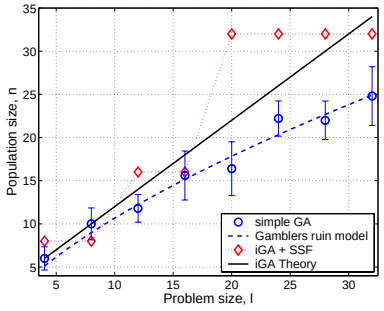
5. DISCUSSION AND CONCLUSIONS

This paper has focused on two critical elements for combating the user fatigue problem of iGAs: (1) the lack of a computable fitness, and (2) how synthetic fitness models based on user evaluation may be built. We proposed a heuristic to synthesize a model of the user fitness combining partial ordering concepts, multiobjective optimization ideas, and support vector machines. The proper trained model provided by a ε-SVM is able to satisfy the two properties a synthetic fitness need to satisfy—*fitness extrapolation*, and *order maintenance*.

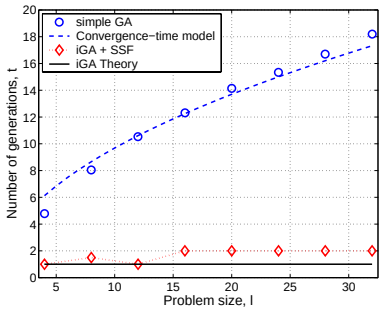
The existence of a synthetic fitness allow us to actively use such model to combat user fatigue. The actively optimizing the synthetic fitness using the compact GA, produce a population of candidate solutions. The injection of such candidate solutions into the user evaluation process effectively reduce the number of evaluations required on the user side till convergence. The empirical results obtained by an unexperienced proved such improvements with speedups ranging from 3 up to 7 times.

6. ACKNOWLEDGMENTS

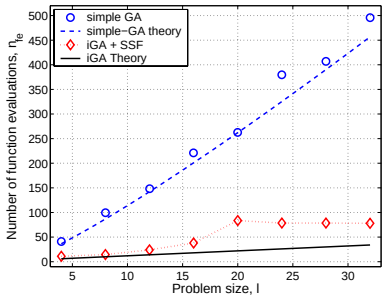
This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF (F49620-03-1-0129), and by the Technology Research, Education, and Commercialization Center (TRECC), at University of Illinois at Urbana-Champaign, administered by the National Center for Supercomputing Applications (NCSA) and funded by the Office of Naval Research (N00014-01-1-0175). The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the Technology Research, Education, and Commercialization Center, the Office of Naval Research, or the U.S. Government.



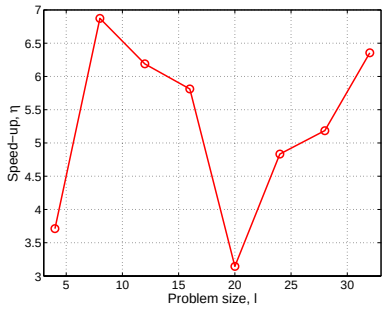
(a) Population Size



(b) Convergence Time



(c) Function Evaluations



(d) Speed up

Figure 6: Analysis of the results obtained using the active iGA propose when compare to a simple iGA.

7. REFERENCES

[1] T. Bäck. Selective pressure in evolutionary algorithms: A characterization of selection mechanisms. *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 57–62, 1994.

[2] T. Bäck. Generalized convergence models for tournament—and (μ, λ)—selection. *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 2–8, 1995.

[3] M. G. Bulmer. *The Mathematical Theory of Quantitative Genetics*. Oxford University Press, Oxford, 1985.

[4] C. Caldwell and V. S. Johnston. Tracking a criminal suspect through face-space with a genetic algorithm. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 416–421. Morgan Kaufmann, 1991.

[5] C. A. Coello-Coello. An updated survey of GA-Based Multiobjective Optimization Techniques. Technical report lania-rd-09-08, Laboratorio Nacional de Informática Avanzada (LANIA), Xalapa, Veracruz, México, December, 1998.

[6] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge Press, 2000.

[7] R. Dawkins. *The blind watchmaker*. New York: W. W. Norton, 1986.

[8] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, 2000.

[9] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley, 1989.

[10] D. E. Goldberg. *The design of innovation: Lessons from and for competent genetic algorithms*. Kluwer Academic Publisher, 2002.

[11] D. E. Goldberg, K. Deb, and J. H. Clark. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6:333–362, 1992. (Also IlliGAL Report No. 91010).

[12] D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5):493–530, 1989.

[13] G. Harik. Linkage Learning via Probabilistic Modeling in the ECGA. IlliGAL Report No. 99010, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 1999.

[14] G. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, 7(3):231–253, 1999. (Also IlliGAL Report No. 96004).

[15] G. Harik, F. Lobo, and D. E. Goldberg. The compact genetic algorithm. *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 523–528, 1998. (Also IlliGAL Report No. 97006).

[16] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press/ Bradford Books edition, 1975.

[17] P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms*. Kluwer Academic Publishers, Boston, MA, 2002.

[18] B. L. Miller. *Noise, Sampling, and Efficient Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, General Engineering Department, Urbana, IL, May 1997. (Also IlliGAL Report No. 97001).

[19] B. L. Miller and D. E. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9(3):193–212, 1995. (Also IlliGAL Report No. 95006).

[20] H. Mühlenbein and D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm: I. continous parameter optimization. *Evolutionary Computation*, 1(1):25–49, 1993.

[21] V. Pareto. *Cours d'Economie Politique, volume I and II*. F. Rouge, Lausanne, 1896.

[22] M. Pelikan, F. Lobo, and D. E. Goldberg. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21:5–20, 2002. (Also IlliGAL Report No. 99018).

[23] K. Sastry. Evaluation-relaxation schemes for genetic and evolutionary algorithms. Master's thesis, University of Illinois at Urbana-Champaign, General Engineering Department, Urbana, IL, 2001. (Also IlliGAL Report No. 2002004).

[24] K. Sastry, D. E. Goldberg, and M. Pelikan. Efficiency Enhancement of Probabilistic Model Building Genetic Algorithms. IlliGAL Report No. 2004020, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 2004.

[25] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Patter Analysis*. Cambridge Press, 2004.

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

[27] D. Thierens and D. E. Goldberg. Mixing in Genetic Algorithms. *Proceedings of the Fifth International Conference in Genetic Algorithms*, pages 38–45, 1993.

[28] D. Thierens and D. E. Goldberg. Convergence models of genetic algorithm selection schemes. *Parallel Problem Solving from Nature*, 3:116–121, 1994.