A Comparison of Three Fitness Prediction Strategies for Interactive Genetic Algorithms

Leuo-hong Wang[1] Jun-de Liao[2]

[1,2]Evolutionary Computation Laboratory, Dept. of Information Management, Aletheia University, Taiwan

Abstract

Human fatigue problem is one of the most important problems encountered by the interactive genetic algorithms (IGA). Different strategies, such as easing the way of evaluation, accelerating convergence of IGA via speedup algorithms and fitness prediction, have been proposed to address this problem. We will focus on studying the performance of fitness prediction strategies in this paper. Three prediction schemes, including a novel prediction method based on algorithmic probability (ALP) proposed by the authors, the neural network (NN), and the Bayesian learning algorithm (BLA) have been consequently examined. Numerical simulations were performed in order to compare the performance of three schemes.

Keywords: Interactive genetic algorithms, human fatigue problem, fitness prediction, algorithmic probability

1. Introduction

Interactive genetic algorithms (IGA), a variant of the genetic algorithms (GA), are the evolutionary computation framework whose selection process is guided by human but not computers. Since formulating the fitness functions is difficult to the optimization problems concerning subjectiveness, preferences, or cognition, humans are involved in the selection process of IGA to determine fitness value of each individual. IGA perform well in the cases of art design [1], industrial design [2] and product design [3], as indicated in previous works. However, IGA still suffer from the human fatigue problem even successfully applying to many applications. Reducing human fatigue therefore becomes one of the most important research issues in this field.

Takagi [4], in an excellent survey, stated that various alternatives have been proposed to address the human fatigue problem. The alternatives included easing the way of evaluation, accelerating convergence of IGA via speedup algorithms and fitness prediction. For more details, please refer to [4]. We will focus on the performance of different fitness prediction strategies in this paper.

Neural network (NN) is the most popular fitness prediction strategy in IGA research. Both Johanson et. al., [5] and Biles et. al., [6] in their IGA/IGP (interactive genetic programming) context of composing creative musical melodies, applied NN to learn the evaluation patterns of the user, and then to predict the fitness values of new individuals in succeeding generations. However, reducing the times of evaluation is the major concern no matter whether fitness prediction strategies are applied. The training data used by NN, in other words, must be restricted, or the human fatigue will be arisen. For such a case, the poor prediction accuracy is inevitable and consequently results in more evaluations being performed, which still cause human fatigue. In fact, the dilemma mentioned above is encountered by all of the statistical based learning methods, including NN. Learning schemes insensitive to the quantities of training data are therefore required for fatigue reduction of IGA. Algorithmic probability (ALP), a fundamental theory of incremental learning proposed by Solomoroff [7], is exactly a learning theory whose prediction errors are insensitive to the quantities of training data [8]. Hence, ALP seems suitable to serve as the fitness prediction strategy for IGA.

A GA-based incremental learning algorithm implementing the concept of ALP is presented in this paper. The algorithm will serve as the core of the prediction module in IGA. Two more learning schemes, including the Bayesian learning algorithm (BLA) and NN will also be implemented for the sake of performance comparison.

The rest of paper is organized as follows. Section 2 introduces the algorithmic probability and the ALP-based predictor which is actually a GA-based incremental learning algorithm. Section 3 shows the experimental results of three learning schemes mentioned above. The final section draws the conclusion and illustrates the future research direction.

2. The Prediction Algorithm

Algorithmic probability (ALP) is a universal probability theory which extends the probability axioms to apply to the cases with small observed samples. The prediction error of learning scheme based on ALP is therefore unconcerned with the quantities of training data, as described in [8]. The concept of ALP is searching many qualified predictors instead of learning only one best predictor from the training data. In order to find out qualified predictors as many as possible, the learning algorithm based on ALP must be capable of identifying a set of predictors whose summation of prediction errors is minimal. In such a case, the learning problem is equivalent to an optimization problem. The objective of the following equation, as proposed in [9], is to optimize the prediction accuracy which is in inverse ratio to the prediction error:

$$\textit{Maximize} \quad \sum_{j} a_0^j \prod_{i=1}^{N} O^j(A_i|Q_i) \qquad (1)$$

N question-answer pairs $(Q_1, A_1),(Q_2, A_2)...(Q_N, A_N)$ are given as the training data in equation 1. $O^j(A_i|Q_i)$ is the conditional probability from the view of the j[th] predictor. a_0^j is the *a priori* probability (algorithmic probability) of j[th] predictor, which is inversely proportionate to the length of predictor. More details please refer to [8].

Solomonoff applied Levin's search, a sequential search method, to determine qualified predictors as many as possible within a limited time period [9]. However, a genetic algorithm based learning alternative is presented in this paper since the problem is obviously a multi-objective optimization problem.

Determining qualified predictors as many as possible is the objective of an ALP-based prediction scheme. In order to identify qualified predictors from the solution space, a GA-based approach is applied. However, niching scheme [10] is incorporated with the canonical GA in our approach since many *distinct* predictors rather than the best one are needed after the search has completed.

The system architecture, including the fitness prediction module implementing the concept of ALP, is sketched in Figure 1. In order to identify various predictors, fitness sharing, one of the methods to implement niching scheme, is applied during the selection phase of GA. Post-processing is inevitable to niching, since the final searching results are usually far away from the local optima. For this reason, a K-Mean clustering technique is firstly used to group the final population into several clusters. A steepest descent algorithm is then utilized for fitness improvement of the best individual in each cluster. Finally, the top M improved individuals construct the current prediction model. Once a prediction model is determined, the fitness values of new individuals produced by IGA can be predicted by this model. The details of our ALP-based IGA system have been described in [11].

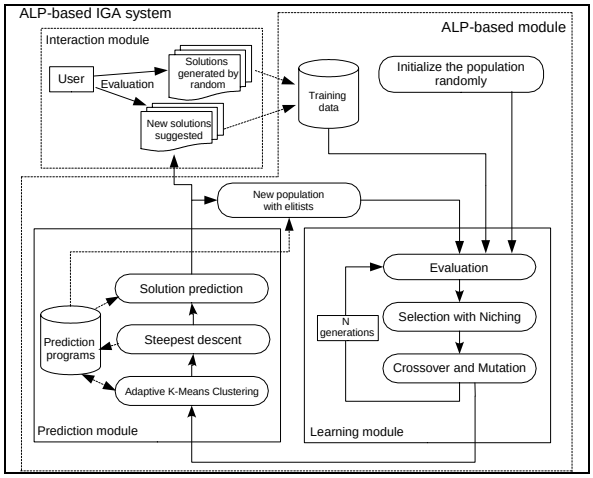

Fig. 1: The system architecture. ALP-based module, which is used to predict fitness values of new individuals, implements ALP concept by using a genetic algorithm with niching scheme.

3. Experimental Results

Three different prediction schemes, which are our ALP-based method, neural network and Bayesian learning algorithm (BLA), have been implemented for exploring the performance issue of the IGA with a fitness prediction module. The product design problem [12] was chosen for the experiments.

As shown in [12], the simplest form of the product design problem considers a product consisting of k relevant attributes. And further, each attribute a_i (i = 1, 2, …, k) has s_i different levels, $l_1^i, l_2^i, \ldots, l_{s_i}^i$, for instance. The product design problem thus becomes a problem of selecting a specific level for each attribute to satisfy the user's preference. Assume P is the set of all possible product profiles:

$$\mathrm{P} = \left\{ a_1 a_2 \cdots a_k \,\middle|\, a_i \in \{ l_1^i, l_2^i, \cdots, l_{s_i}^i \} \right\} \qquad (2)$$

If $U(\bullet)$ represents the part-worth utility function in user's mind, then searching the best product profile will be equivalent to determine $p^* \in P$ such that $U(p^*) \geq U(p), \forall p \in P, p \neq p^*$. $U(\bullet)$ is consequently the fitness function of IGA.

3.1. Implementation Details

The implementation details of ALP-based method, NN, and Bayesian learning algorithm are explained as follows.

Both ALP-based method and Bayesian learning algorithm were implemented by using GA. Assume k is the numbers of attributes and, with loss of generality, each attribute has s levels. The chromosome is then a real value array with k*s elements for both methods.

Additionally, each chromosome has an a priori probability corresponding with the current training data. The initial values of all the a priori probabilities equal to $1/|U|$, if $1/|U|$ represents the numbers of possible solutions. Moreover, given N pairs of training data, $(Q_1, A_1),(Q_2, A_2)...(Q_N, A_N)$ for example, the a priori probability of chromosome C_j is:

$$\prod_{i=1}^{N} (1 - \Pr(-|e_i| \le E \le |e_i|)) \quad (3)$$

where e_i equals to $(U(C_j) - A_i)$ which is the C_j's prediction error corresponding to (Q_i, A_i). Moreover, E is a random variable representing the prediction error of C_j. Each term in the product is therefore the probability that C_j can correctly predict by A_i given Q_i. This equation serves as the fitness functions of our ALP-based module and the Bayesian learning module.

A three-layer feed-forward network is used by the NN prediction module. The numbers of network inputs and the neurons of each hidden layer are both proportionate to the problem size, that is, the total the numbers of possible product profiles. In addition, a sigmoid transfer function is used in hidden layers. The output transfer function is a linear function, which generated a utility value for a given product profile. The steepest descent algorithm is used for training after the entire training set has been applied to the network. The training stops if the number of iterations exceeds 100.

3.2. Results and Discussion

Three different problem sizes were used in our experiments. All of them were 5 attributes, but for different sizes, there were 4, 8, and 16 levels respectively. In other words, the numbers of profiles are 2^{10}, 2^{15} and 2^{20} respectively.

The Monte Carlo method was used to generate the answers, that is, the utility value of each level. The training data, which were the product profiles and their corresponding scores, were generated by random. Moreover, once a new pair of profile and its score was generated, the prediction module would restart a new learning process. Updating the training data one by one and restarting the learning process is so as to observe the efficiency of prediction modules. However, updating generation by generation will be the case in an IGA context.

The population size used for all experiments was 10 because we found that larger population sizes did not effectively improve the accuracy of prediction. The crossover and mutation operators were linear crossover and random mutation respectively. It is common for real-valued encoding GA. The number of generation was fixed to 1000. Lastly, 30 runs of simulation were running for each experiment.

The average mean square errors (MSE) of all possible profiles for small, medium and large cases are sketched in Figure 2 and 3.

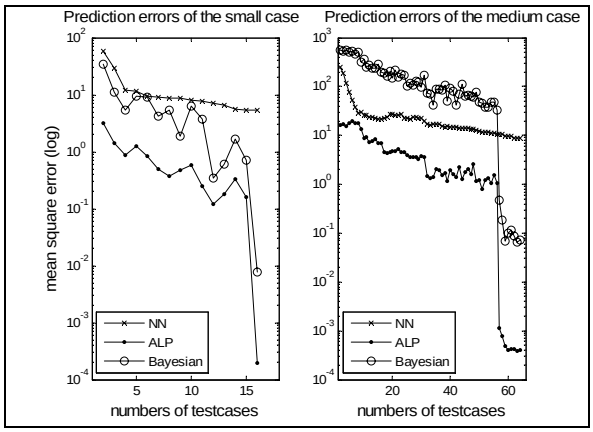

Fig. 2: The mean square errors of the small and medium cases for all three prediction modules.

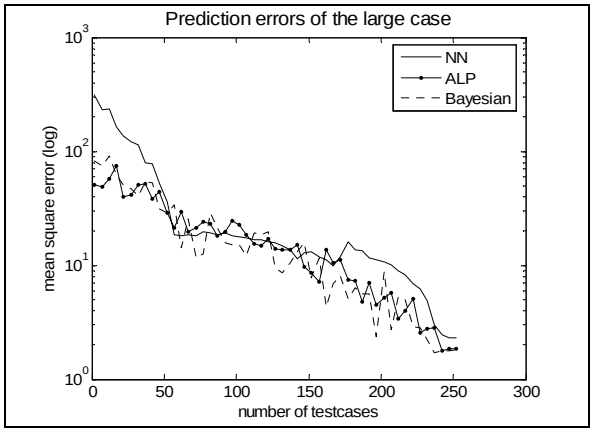

Fig. 3: The mean square errors of the large cases for all three prediction modules.

The MSE ratios of NN to ALP are almost on the order of magnitude of 10 for both small and medium cases, as demonstrated in Figure 2. If the last few values are omitted, the ratios actually range from 9.45 to 59.73 in the small case and from 1.68 to 15.62 in

the medium case. BLA performs better than NN in the small case, but gets worse in the medium case. Moreover, the correctness percentages can be derived from the MSE data points in Figure 2 by applying equation 3. The correctness percentages of both NN and BLA modules consequently grow gradually from under 50% to above 90%. However, the ones of ALP grow from 80% to above 95%. The ALP module, as indicated in Figure 2, predicts correctly even the training data set is very small, just as being claimed previously.

The ALP prediction module being superior to statistical based prediction methods for the cases of small training data set is evidenced by the experimental results. In addition, the capabilities of statistical based methods emerging after training data being increased are also explained. The prediction strategy based on ALP, in sum, is more probably than NN or BLA for human fatigue reduction.

Scalability is one of the problems for our current ALP implementation. When the solution space becomes larger, the MSE of ALP is close to NN and BLA all the time, as shown in Figure 3. However, ALP should be theoretically better than the other two if the predictors found by the module are good enough.

Efficiency is the other problem of ALP since the execution time of both NN and BLA modules are far faster than ALP. The ALP, in fact, gets instant response only in the small case. When the size of training data increases gradually, the response time dramatically decelerates. Applying local optimization to improve the correctness of predictors is the reason to make ALP slowing down. Hence, implementing ALP by using multi-objective GA without any local optimization is a feasible alternative for efficiency improvement.

4. Conclusion

The preliminary comparison of three different fitness prediction schemes has been made in this study. We have found that ALP indeed performs better than NN and BLA in small or medium problem sizes. Fewer training data, in other words, are required for learning user's preference by ALP. In such a case, ALP is more probably than NN or BLA to reduce human fatigue in IGA. However, our current ALP implementation still suffers from the poor scalability and efficiency. We will implement ALP by using multi-objective genetic algorithms to improve the efficiency in near future.

5. References

[1] N. Tokui, H. Iba, "Music Composition with Interactive Evolutionary Computation," *Proceedings of 3rd International Conference on Generative Art*, 2000.

[2] H. S. Kim, S.B. Cho, "Application of Interactive Genetic Algorithm to Fashion Design," *Engineering Application of Artificial Intelligence*, vol. 13 , pp. 635-644, 2000.

[3] F. C. Hsu, P. Huang, "Providing an appropriate search space to solve the fatigue problem in interactive evolutionary computation," *New Generation Computing*, vol. 23, 2005.

[4] H. Takagi, "Interactive Evolutionary Computation: Fusion of the Capacities of EC Optimization and Human Evaluation," *Proceedings of the IEEE*, vol. 89, pp. 1275-1296, 2001.

[5] B. Johanson, R. Poli, "GP-Music: An Interactive Genetic Programming System for Music Generation with Automated Fitness Raters," *Genetic Programming 1998: Proceedings of the Third Annual Conferenc*e, pp. 181-186, 1998.

[6] J. A. Biles, P. G. Anderson, L. W. Loggi, "Neural Network Fitness Functions for a Musical IGA," *International ICSC Symposium on Intelligent Industrial Automation and Soft Computing*, 1996.

[7] R. J. Solomonoff, "Two Kinds of Probabilistic Induction," *The Computer Journal*, vol. 42, pp. 256-259, 1999.

[8] R. J. Solomonoff, "The Application of Algorithmic Probability to Problems in Artificial Intelligence", *Uncertainty in Artificial Intelligence*, pp. 473-491, Elsevier Science Publishers B.V., 1986.

[9] R. J. Solomonoff, "Progress in incremental machine learning," TR-IDSIA-16-03 revision 2.0, 2003.

[10] J. Gan, K. Warwick, "A Genetic Algorithm with Dynamic Niche Clustering for Multimodal Function Optimisation," *Proc. Int. Conference on Artificial Neural Nets and Genetic Algorithms*, 1999.

[11] L. H. Wang, P. Y. Wei, Y. T. Chang, "Reducing Evaluation Fatigue in Interactive Evolutionary Algorithms by Using an Incremental Learning Approach," *Soft Computing as Transdisciplinary Science and Technology*, Springer-Verlag, 2005.

[12] R. Kohli, R. Krishnamurti, "A heuristic approach to product design," *Management Science*, vol. 33 , pp.1523-1533, 1987.