

A Comparison of Three Fitness Prediction Strategies for Interactive Genetic Algorithms*

LEUO-HONG WANG

*Evolutionary Computation Laboratory
Department of Information Management
Aletheia University
Tamsui, 251 Taiwan*

The human fatigue problem is one of the most significant problems encountered by interactive genetic algorithms (IGA). Different strategies have been proposed to address this problem, such as easing evaluation methods, accelerating IGA convergence via speedup algorithms, and fitness prediction. This paper studies the performance of fitness prediction strategies. Three prediction schemes, the neural network (NN), the Bayesian learning algorithm (BLA), and a novel prediction method based on algorithmic probability (ALP), are examined. Numerical simulations are performed in order to compare the performances of these three schemes.

Keywords: interactive genetic algorithm, human fatigue problem, fitness prediction, algorithmic probability, product design

1. INTRODUCTION

Interactive genetic algorithms (IGA), a variant of genetic algorithms (GA), are an evolutionary computation framework whose selection process is guided by humans, not computers. Since formulating fitness functions is difficult for optimization problems concerning subjectivity, preferences, or cognition, humans are involved in the IGA selection process to determine the fitness value of each individual. IGA is effective in art design [1], industrial design [2], and product design [3], as indicated in previous work. IGA still suffers from the human fatigue problem, however, even when it has successfully been applied. The human fatigue problem, or the evaluation fatigue problem, arises from the fact that respondents must evaluate the fitness value of every single individual from generation to generation. Respondents quickly lose interest or attention as generations pass and inevitably provide biased evaluation results, either intentionally or unintentionally. Consequently, the evolutionary process will be significantly affected because of these defective fitness values. Therefore, reducing human fatigue has become one of the most important research issues in this field.

Takagi [4] stated that various alternatives have been proposed to address the human fatigue problem. These alternatives include easing evaluation methods, accelerating IGA convergence via speedup algorithms, and fitness prediction. For more details, please refer to [4]. This paper focuses on the *fitness prediction* alternative. The performances of three different fitness prediction strategies, including one novel prediction method im-

Received September 2, 2005; revised January 18 & March 9, 2006; accepted April 10, 2006.

Communicated by Pau-Choo Chung.

* The preliminary version of this paper was presented in 8th Joint Conference on Information Sciences, Salt Lake City Marriott-City Center, July 21-25th, 2005.

plementing the theory of algorithmic probability (ALP) [5], will be discussed in the following sections.

Neural network (NN) is the most popular fitness prediction strategy in IGA research. Both Johanson *et al.* [6] and Biles *et al.* [7] applied NN to learn the evaluation patterns of respondents, and then predicted the fitness values of new individuals in succeeding generations in their IGA/IGP (Interactive Genetic Programming) context of composing creative musical melodies. Reducing respondents' evaluation times, however, is the major concern of fitness prediction alternatives, no matter which strategy is applied. The training data used by NN must be restricted to some extent, or else human fatigue will occur. On the other hand, if the training data is so restricted as to avoid evaluation fatigue, NN will not predict effectively. In that case, more training data or more respondent evaluations will be required, which ultimately will also result in evaluation fatigue.

The dilemma mentioned above is encountered by all statistical-based learning methods, including NN. Therefore, learning schemes insensitive to training data quantity are required for IGA fatigue reduction. Algorithmic probability (ALP), a fundamental theory of incremental learning proposed by Solomonoff [5], is a learning theory whose prediction errors are insensitive to training data quantity [8]. Therefore, ALP seems more suitable as the IGA fitness prediction strategy.

A GA-based incremental learning algorithm that implements the concept of ALP is presented in this paper. The algorithm serves as the core of the IGA prediction module. Two more learning schemes, the Bayesian learning algorithm (BLA) and NN, are also implemented for performance comparison.

This paper introduces a new, ALP-based learning scheme insensitive to training data quantity in order to address the IGA human fatigue problem. This paper also introduces a new implementation framework by utilizing the power of niche GA for ALP-based systems. The current implementation for addressing the human fatigue problem is successful with small and medium test problem sizes.

The rest of the paper is organized as follows. Section 2 introduces algorithmic probability and the derived ALP-based prediction strategy, which is actually a GA-based incremental learning algorithm. Section 3 addresses the research problem, as well as the implementation details and the experimental results of the three learning schemes mentioned above. The final section draws conclusions and suggests future research directions.

2. THE PREDICTION ALGORITHM

2.1 Algorithmic Probability

Algorithmic probability (ALP) is a universal probability theory that extends probability axioms to cases with small observed samples. Therefore, the prediction errors of ALP-based learning schemes are unconcerned with training data quantity, as described in [8]. ALP searches for many qualified predictors, instead of learning only one optimal predictor from the training data. In order to discover as many qualified predictors as possible, the ALP-based learning algorithm must be capable of identifying a set of predictors so that the summation of their 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:

$$\text{Maximize } \sum_j a_0^j \prod_{i=1}^N O^j(A_i | Q_i), \quad (1)$$

N question-answer pairs $(Q_1, A_1), (Q_2, A_2), \dots, (Q_N, A_N)$ are given as the training data in Eq. (1). Note that N is not necessarily a large enough number for training, as mentioned above. $O^j(A_i | Q_i)$ is a conditional probability that the j th predictor can correctly predict the answer A_i under Q_i . a_0^j is the initial *a priori* probability, or *algorithmic probability*, of the j th predictor, which is inversely proportionate to the length of the predictor. For more details, please refer to [8].

Solomonoff applied Levin's search, a sequential search method, to determine as many qualified predictors as possible within a limited time period [9]. A genetic algorithm-based learning alternative is presented in this paper, however, since the problem listed in Eq. (1) is obviously a multi-objective optimization problem.

2.2 The ALP-based IGA System

The objective of an ALP-based prediction scheme is to determine as many qualified predictors as possible. Therefore, a GA-based approach is applied in order to identify qualified predictors from the solution space. The canonical GA is, however, unsuitable for implementing ALP-based prediction schemes because only the best solution is located. The niching technique [10], which can preserve the population as multiple niches during evolution, is therefore integrated with the canonical GA in this paper, since many *distinct* predictors, rather than just the best one, are needed after searching.

In this study, an IGA system is reformulated into a system consisting of three different modules: the interaction, learning, and prediction modules. If an ALP-based learning strategy is applied, these modules compose an ALP-based IGA system. The system architecture is sketched in Fig. 1.

The interaction module in Fig. 1 is responsible for collecting training data. The learning module is actually a genetic algorithm module. Therefore, data collected from the interaction module will be plugged into this module as the fitness function parameters. The learning module will locate the predictors that correspond with the given data. All predictors found will be stored in the prediction module for further processing. The prediction module will then use these predictors to forecast future events. If the user is not satisfied with the solutions presented in the interaction module, the prediction module will assist the interaction module in generating the next few solutions for further evaluation.

The ALP-based module, composed of learning and prediction modules, is implemented by integrating a genetic algorithm with niching, as well as a local optimizer. These parts are explained separately below.

1. A genetic algorithm with dynamic niching. This is the core of the learning module. As mentioned above, a niching scheme that disperses the population into subgroups

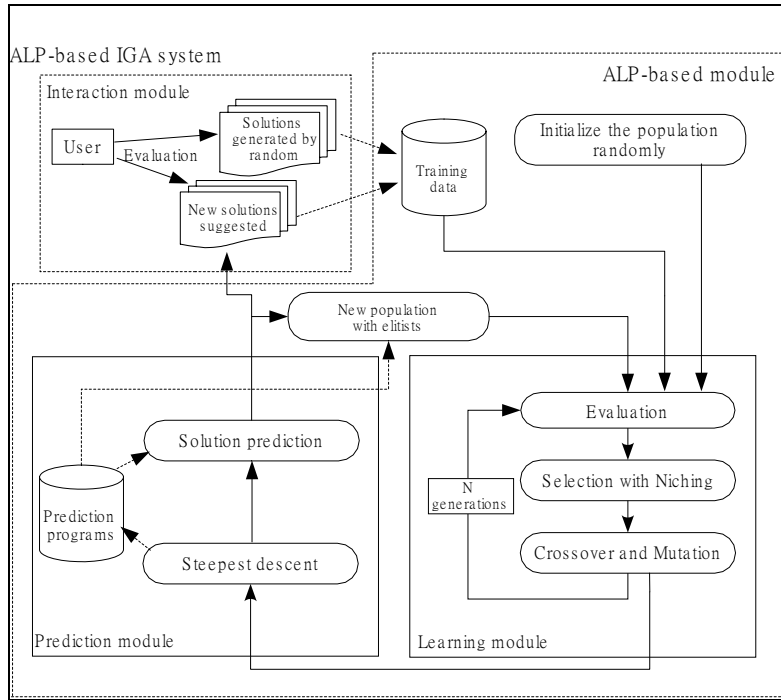


Fig. 1. The system architecture. An ALP-based module, which is used to predict fitness values of new individuals, implements ALP using a genetic algorithm with a niching scheme.

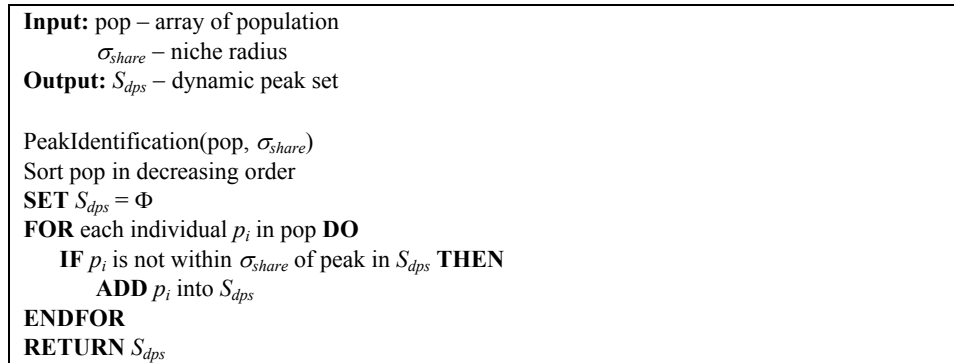


Fig. 2. Dynamic peak identification.

has been integrated with the canonical GA. A dynamic niching algorithm proposed by Gan and Warwick [10] was implemented. The algorithm adopts fitness sharing [11] to maintain population diversity. The shared fitness of individual i is expressed by $f_{sh,i} = f_i/m_i$, where f_i is the raw fitness of the individual and m_i is the niche count, which defines the amount of sharing between an individual and the rest of the population. Unlike the original fitness sharing algorithm, which has to define the niche count in advance, the dynamic niching algorithm automatically calculates the niche count using a greedy peak identification procedure, shown in Fig. 2. The niche radius σ_{share} is

determined by a sensitivity analysis procedure that compares the performance of many different values. This procedure is applied to each generation. The peaks identified by this procedure exactly correspond to the candidate predictors. Consequently, a local optimization procedure is applied to improve these candidates. Once the training data are updated by the interaction module, the learning module searches for distinct predictors corresponding to the updated data.

2. Local optimization. Since the final search results are usually far from local optima, further processing is inevitable. For this reason, a steepest descent algorithm is utilized to improve the fitness value of the best individual in each niche generated by the dynamic niching scheme. The algorithm is illustrated in Fig. 3. The search direction h_i is actually a vector whose components are the partial differential of each dimension. This algorithm's performance is strongly dependent on the definition of λ and the termination condition. As suggested by [12], possible termination conditions include when a maximum number of iterations is reached, when a solution with a fitness less than a predefined threshold value is found, or when the maximum number of iterations without improvements is reached. In this paper, the first two termination conditions are adopted. Finally, these 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 using this model.

```

Input:  $S_{dps}$  – dynamic peak set
          $\lambda$  – step size
Output:  $S_{opt}$  – peak set after optimization

SteepestDescent( $S_{dps}$ ,  $\lambda$ )
SET  $S_{opt} = \Phi$ 
FOR each element  $s$  in  $S_{dps}$ 
  SET  $s' = s$ 
  WHILE termination conditions not met DO
    Decide search direction  $h_i \in R^n$ 
    SET  $s' = s + \lambda \cdot h_i$ 
  ENDWHILE
  ADD  $s'$  into  $S_{opt}$ 
ENDFOR
RETURN  $S_{opt}$ 

```

Fig. 3. The steepest descent algorithm.

3. THE RESEARCH PROBLEM AND EXPERIMENTAL RESULTS

3.1 The Problem and Implementation Details

Three different prediction schemes, the ALP-based method, neural network, and Bayesian learning algorithm (BLA), were implemented for exploring the IGA performance issue with a fitness prediction module. The product design problem was chosen to investigate the performance of the proposed system and the other two alternatives. The product design problem has been proven to be NP-hard [13] and various heuristics to the

problem have been proposed. As shown in [14], the simplest form of the product design problem considers a product consisting of k relevant attributes. Each attribute a_i ($i = 1, 2, \dots, k$) has s_i different levels, $l_1^i, l_2^i, \dots, l_{s_i}^i$. The product design problem thus becomes a problem of selecting a specific level for each attribute that satisfies the user's preferences. Assume P is the set of all possible product profiles:

$$P = \{a_1 a_2 \dots a_k \mid a_i \in \{l_1^i, l_2^i, \dots, l_{s_i}^i\}\}. \quad (2)$$

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

The simplest product design problem has analytical solutions if we assume the attributes are linear independent. Unfortunately, in order to figure out the solutions, respondents need to evaluate full product profiles, or parts of them. Once the number of possible product profiles becomes too large and causes human fatigue, the solutions will be biased.

The ALP-based module has been simplified in several ways to fit the simplest form of the product design problem. The implementation details are explained as follows:

1. The chromosome. Theoretically, each chromosome must be a randomly generated predictor used to predict future events. In other words, the chromosome must be able to simulate all possible objectives in a user's mind. We assume the objective function of the simplest form of the product design problem is a linear equation, which is defined as:

$$U(p) = U(a_1) + U(a_2) + \dots + U(a_k) \quad (3)$$

where p is a profile consisting of levels $a_1 a_2 \dots a_k$. Therefore, there is only one kind of predictor. The chromosome consists of the prediction utility value of each level for all attributes. Without loss of generality, assume a product with k attributes, all of which have s levels. Thus, the chromosome will be a real value array with $k \times s$ elements.

2. The *a priori* probability and the fitness function. Additionally, each chromosome has an *a priori* probability that corresponds with the current training data. All initial values of the *a priori* probabilities equal $1/|U|$, if $|U|$ represents the number of possible solutions. Moreover, given N pairs of training data, $(Q_1, A_1), (Q_2, A_2), \dots, (Q_N, A_N)$, the *a priori* probability of chromosome C_j is:

$$\prod_{i=1}^N (1 - \Pr(-|e_i| \leq E \leq |e_i|)) \quad (4)$$

where e_i equals $(U(C_j) - A_i)$, which is C_j 's prediction error corresponding to (Q_i, A_i) . e is a random variable representing the prediction error of C_j . Therefore, each term in the product is the probability that C_j can correctly predict A_i given Q_i . Eq. (4) serves as the fitness functions of the ALP-based module.

The Bayesian learning algorithm was implemented identically to the ALP-based

module. The only difference was that the former used the canonical GA without incorporating niching. The Bayesian learning algorithm, in other words, searched for the best predictor.

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

3.2 Results and Discussion

Three different problem sizes were used in the experiments. All of them had 5 attributes, but were for different sizes. There were 4, 8, and 16 levels. In other words, there were 2^{10} , 2^{15} , and 2^{20} profiles, respectively. Theoretically, 16, 64, and 256 interactions are needed for these three problems to obtain analytical solutions if we use experimental design techniques, such as the orthogonal array to design profiles. If there is any inconsistency during evaluation, however, more interactions are needed. Empirical studies indicate that three times the theoretical values (48, 192, and 768) are required.

The Monte Carlo method was used to generate answers, or the utility value of each level for all attributes. The training data, which were the product profiles and their corresponding scores, were generated randomly. Once a new 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 done to observe the efficiency of prediction modules. Updating generation by generation, however, is the procedure in an IGA context.

The population size used for all experiments was 10 because larger population sizes did not improve the prediction accuracy. The crossover and mutation operators were linear crossover and random mutation, respectively. This is common for real-valued encoding GA. The number of generations was fixed at 1,000. Finally, 30 runs of the simulation were running for each experiment.

The average mean square errors (MSE) of all possible profiles for small, medium, and large cases are shown in Figs. 4 and 5.

The MSE ratios of NN to ALP are almost 10 for both small and medium cases, as illustrated in Fig. 4. 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 outperforms NN in the small case, but its performance decreases in the medium case. The correctness percentages can be derived from the MSE data points in Fig. 4 by applying Eq. (4). The correctness percentages of both NN and BLA modules increase gradually from under 50% to over 90%. ALP correctness percentages, however, increase from 80% to over 95%. The ALP module, as indicated in Fig. 4, predicts correctly even when the training data set is very small, just as claimed previously.

The ALP prediction module is superior to statistical-based prediction methods for small training data sets, as proven by the experimental results. We next discuss the

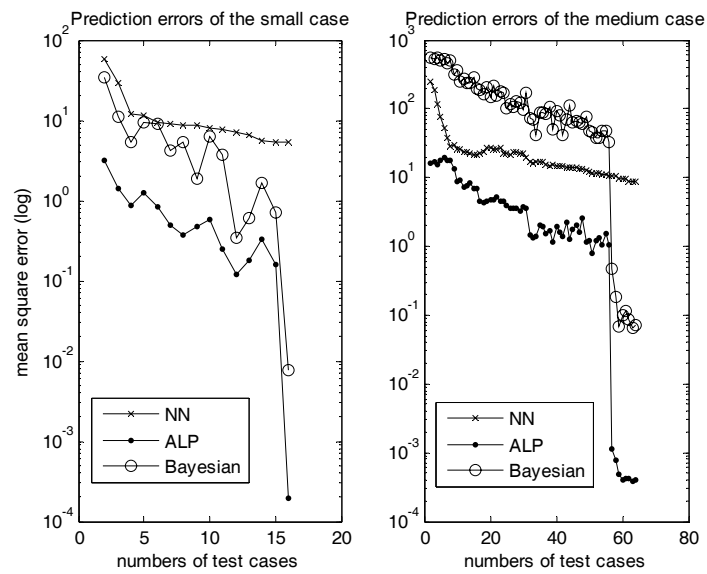


Fig. 4. The mean square errors of the small and medium cases for all three prediction modules.

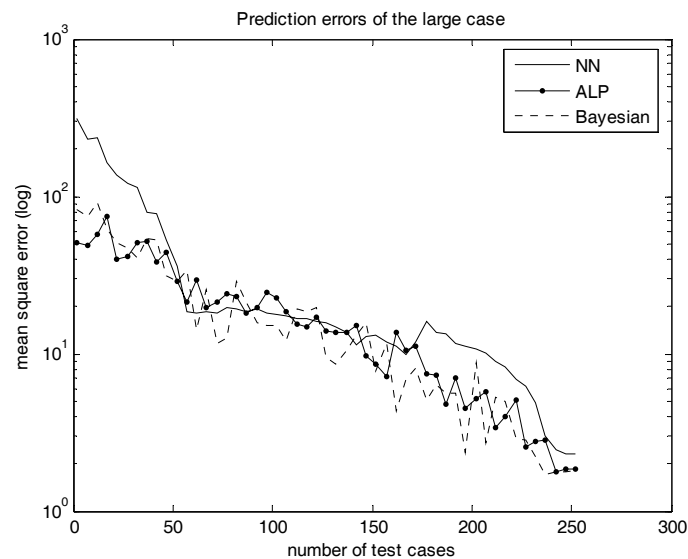


Fig. 5. The mean square errors of the large cases for all three prediction modules.

capabilities of statistical-based methods as training data increases. The ALP-based prediction strategy, in sum, is more effective than NN or BLA for human fatigue reduction.

When the solution space increases, the MSE of ALP is still better than NN and BLA during the early stages of learning, but gradually grows closer to NN and BLA as training data increase, as shown in Fig. 5. As stated previously, the ALP prediction errors are unconcerned with the training data quantity. It is therefore expected for an ALP-based

system to outperform NN and BLA when there is a small amount of training data, and for it to perform as well as NN and BLA when training data are large. In Fig. 5, ALP indeed outperforms the other two schemes when training test cases are lower than 20, however, the performance deteriorates rapidly if training test cases increase above 20.

Evidence indicates that the current ALP implementation can be improved in order to solve problems with a large solution space. As shown in Fig. 6, the prediction correctness of the large problem size (5 attributes, 16 levels for each attribute) is worse than the other two cases. In addition, if 95% is set as the stop criterion of interaction, 11 and 56 interactions are needed for the small and medium cases, respectively. In the case of the large problem size, however, more than 256 interactions, which is the optimal number of interactions required by an orthogonal array design to obtain an analytical solution, are needed to predict with greater accuracy. If the IGA generations are limited to 20, and the population size is no larger than 8, then the user burden cannot be reduced if more than 256 interactions are performed. For small and medium problem sizes, however, the user burden is reduced easily.

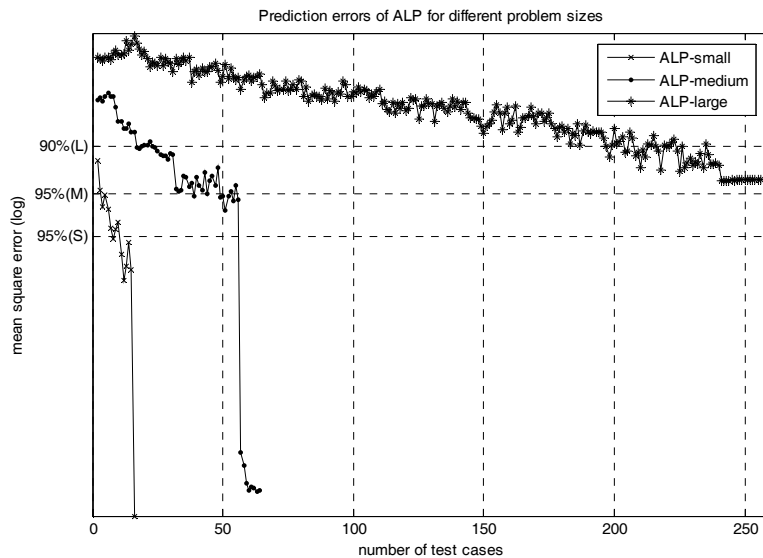


Fig. 6. The prediction errors of the ALP module in all cases.

ALP's problem is efficiency, since the execution times of both NN and BLA modules are far lower than ALP. ALP, however, receives an instant response for small cases. When training data size increases gradually, the response time dramatically decreases. Applying local optimization to improve predictor correctness is the main reason ALP slows down. As shown in Fig. 3, the performance of the local optimization algorithm is strongly dependent on the definition of λ (step size) and the termination condition. If m is the number of levels, n is the current number of training data, and s is the number of steps executed by the algorithm, then the time complexity of the algorithm is of the order $O(mns)$. The step size influences the quality of the solution, as well as the number of

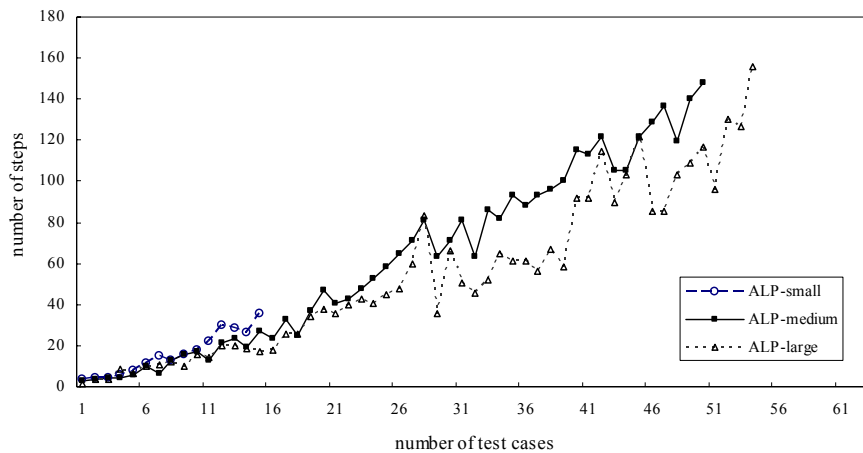


Fig. 7. The number of steps executed by the local search algorithm. For simplicity, only parts of data are shown for the medium and large cases.

Table 1. The relationship between the number of steps executed by the local optimization algorithm and the prediction correctness.

maximum number of steps		1	2	3	4	5	6	7	8	9	10
Large	longest response time	0.453	0.882	1.135	1.453	1.875	2.325	3.412	4.003	4.679	5.120
	prediction correctness	53%	56%	63%	66%	70%	73%	78%	82%	82%	83%
Medium	longest response time	0.237	0.432	0.683	0.797	1.012	1.198	1.375	1.728	2.045	2.502
	prediction correctness	60%	63%	66%	71%	79%	83%	86%	88%	90%	91%

steps executed by the algorithm. The termination condition, however, is the main influence on the number of steps. Fig. 7 shows the number of steps executed by the local optimization algorithm for all three cases. Since the algorithm stops if a solution with a fitness less than a predefined threshold value is found, the number of steps increases linearly as the training data increase. This heavily degrades the response time of the system. Therefore, another termination condition is chosen that stops the algorithm if a predefined maximum number of steps are reached. As a result, if the predefined maximum number of steps is small enough, the time complexity of the algorithm would be reduced to $O(mn)$ because prediction correctness is sacrificed. Table 1 lists the relationship between prediction correctness and the longest response time with different numbers of steps. Only the medium and large cases are listed.

The number of steps only ranges from 1 to 10 because the response time becomes unacceptable beyond 10. In Table 1, 6 steps are acceptable for large cases and 9 or 10 steps are acceptable for medium cases.

Finally, in the context of product design, estimating the utility value of each level across attributes is very important. These utilities can be used to calculate attribute importance and the marketing share of different possible profiles, which is valuable marketing information. Therefore, compared to NN, both ALP and BLA have an inherent advantage in the product design problem: they can predict every possible solution. In other words, ALP and BLA can estimate utilities of any arbitrary profiles, along with the

utility of every single level, even when using all complete profile instances. In contrast, NN can only predict utilities of profiles if the training data set consists of profile instances.

4. CONCLUSION

This study compared three different fitness prediction schemes and found that ALP outperformed NN and BLA in small or medium problem sizes. Fewer training data, in other words, are required for learning a user's preferences through ALP. In such a case, ALP is more suitable than NN or BLA to reduce human fatigue in IGA. The current implementation of ALP once suffered from the efficiency problem. After modifying the termination conditions of local optimization, however, the efficiency and solution quality are now both acceptable for small and medium cases. The efficiency for large cases was also improved. A new method for addressing the human fatigue problem in IGA has been implemented and tested in this paper. Future research can implement ALP using multi-objective genetic algorithms to further enhance the performance of the ALP-based system.

REFERENCES

1. N. Tokui and H. Iba, "Music composition with interactive evolutionary computation," in *Proceedings of the 3rd International Conference on Generative Art*, 2000, pp. 215-226.
2. H. S. Kim and S. B. Cho, "Application of interactive genetic algorithm to fashion design," *Engineering Application of Artificial Intelligence*, Vol. 13, 2000, pp. 635-644.
3. F. C. Hsu and P. Huang, "Providing an appropriate search space to solve the fatigue problem in interactive evolutionary computation," *New Generation Computing*, Vol. 23, 2005, pp. 115-127.
4. H. Takagi, "Interactive evolutionary computation: fusion of the capacities of EC optimization and human evaluation," in *Proceedings of the IEEE*, Vol. 89, 2001, pp. 1275-1296.
5. R. J. Solomonoff, "Two kinds of probabilistic induction," *The Computer Journal*, Vol. 42, 1999, pp. 256-259.
6. B. Johanson and R. Poli, "GP-music: an interactive genetic programming system for music generation with automated fitness raters," in *Proceedings of the 3rd Annual Conference on Genetic Programming*, 1998, pp. 181-186.
7. J. A. Biles, P. G. Anderson, and L. W. Loggi, "Neural network fitness functions for a musical IGA," in *Proceedings of the International Symposium on Intelligent Industrial Automation and Soft Computing*, 1996, pp. 339-344.
8. R. J. Solomonoff, "The application of algorithmic probability to problems in artificial intelligence," in L. N. Kanal and J. F. Lemmer, eds., *Uncertainty in Artificial Intelligence*, Elsevier Science Publishers B.V., North Holland, 1986, pp. 473-491.
9. R. J. Solomonoff, "Progress in incremental machine learning," Technical Report No. IDSIA-16-03, revision 2.0, 2003.

10. J. Gan and K. Warwick, "A genetic algorithm with dynamic niche clustering for multimodal function optimization," in *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms*, 1999, pp. 248-255.
11. D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal-function optimization," in *Proceedings of the 2nd International Conference on Genetic Algorithms and their Applications*, 1987, pp. 41-49.
12. C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: overview and conceptual comparison," *ACM Computing Surveys*, Vol. 35, 2003, pp. 268-308.
13. R. Kohli and R. Krishnamurti, "Optimal product design using conjoint analysis: computational complexity and algorithms," *European Journal of Operational Research*, Vol. 40, 1989, pp. 186-195.
14. R. Kohli and R. Krishnamurti, "A heuristic approach to product design," *Management Science*, Vol. 33, 1987, pp. 1523-1533.



Leuo-Hong Wang (王柳鋐) is a faculty member at Department of Information Management in Aletheia University. He received his B.S., M.S. and Ph.D. degree in Computer Science and Information Engineering from National Taiwan University in 1990, 1992 and 1997, respectively. His research interests include evolutionary computation, web mining, text mining and chance discovery.