A Bayesian Framework for Evolutionary Computation

Byoung-Tak Zhang
Artificial Intelligence Lab (SCAI)
Dept. of Computer Engineering
Seoul National University
Seoul 151-742, Korea
http://scai.snu.ac.kr/~ btzhang

Abstract- A Bayesian framework for evolutionary computation is presented. Given a data set for fitness evaluation, the best (fittest) individual is defined as the most probable model of the data with respect to the prior knowledge on the problem domain. In each generation, Bayes theorem is used to estimate the posterior fitness of individuals from their prior fitness values. Offspring individuals are then generated by sampling from the posterior distribution combined with the transition probabilities formed by variation operators. The evolutionary inference steps from the prior via posterior distribution of parent fitness to the expected fitness distribution of offspring are essential elements in Bayesian evolutionary computation. One of the most interesting aspects of Bayesian evolution is that it provides principled techniques for controlling evolutionary dynamics. Specifically, we describe two examples of the application of the Bayesian framework. One is a Bayesian evolutionary algorithm (BEA) designed to evolve parsimonious individuals in evolutionary computation with variable-size representation. We show that the adaptive Occam method for program growth control is a special form of Bayesian evolution. The other example is an evolutionary algorithm with incremental data inheritance (IDI). In this BEA, the fitness of individuals is estimated on incrementally chosen data subsets, rather than on the whole data set, and thus the convergence is accelerated by reducing the effective number of fitness evaluations. Experimental results are provided to show the effectiveness of the BEAs.

Keywords: Fitness distribution, evolutionary dynamics, Bayesian evolutionary algorithms, adaptive Occam method, incremental data inheritance.

1 Introduction

Bayesian probabilities are often used in statistical inference for specifying a priori knowledge and combining this knowledge with available data via Bayes theorem [8]. The knowledge about the system before the data are known is encoded in the form of a prior probability distribution. Bayes formula then provides a rule for updating prior probabilities after the data are known.

We propose to use the Bayesian inductive principle as a theoretical framework for evolutionary computation. Starting from an initial population of individuals, evolutionary algorithms iteratively produce the next generation of fitter individuals [1, 3]. By defining the fittest model as the most probable model with respect to the data and the prior knowledge, the Bayes theorem can be used to estimate the posterior fitness of individuals from the prior fitness. The posterior probabilites are then used to generate plausible offspring individuals by sampling from the transition distribution formed by variation operators such as mutation and recombination.

Based on this theoretical foundation we present two Bayesian evolutionary algorithms (BEAs) that employ techniques for effective control of evolutionary dynamics of existing evolutionary algorithms.

The first BEA focuses on the fact that the posterior probability of a model is decomposed into two terms: the probability of the data given the model and the prior probability of models. By appropriately choosing the priors on the models and by making use of the relationship between the probability and optimal code length in coding theory, we derive a Bayesian technique for controlling the model complexity during evolution of variable-size models. In particular, we show that the adaptive Occam method described in [14] can be derived from the Bayesian evolutionary framework.

In the second BEA, we make use of the fact that the Bayes formula suggests an incremental, evolutionary learning rule: infer models of higher posterior probability from the existing models (generated according to the prior probability distribution) by observing new data (and training on them to increase the likelihood of the model). The Bayesian evolutionary algorithm with the incremental data inheritance (IDI) method [12] is an example of this approach. In this BEA, the fitness of individuals is estimated on incrementally chosen data subset, rather than on the whole data set, and thus the convergence is accelerated by reducing the effective number of fitness evaluations.

The paper is organized as follows. In Section 2 we present the Bayesian theory of evolutionary computation and formally describe a canonical Bayesian evolutionary algorithm. Section 3 describes the Bayesian method for program growth control. In Section 4, the method of incremental data inheritance is derived from the Bayesian framework and its effect on convergence rate is experimentally demonstrated. Section

0-7803-5536-9/99/$10.00 ©1999 IEEE

5 discusses further implications of the Bayesian approach for the design and analysis of evolutionary algorithms.

2 Bayesian Formulation of Evolutionary Computation

Evolutionary algorithms work by initializing a population of individuals and iteratively producing the next generation of fitter individuals. Let $\mathcal{A}(g)$ denote the population of size M at gth generation

$$\mathcal{A}(g) = \{A_i^g\}_{i=1}^{\mu}, \tag{1}$$

where A_i^g are the individuals. Variation operators such as mutation and recombination are used to produce offspring individuals from the parent individuals. The performance of individuals is measured by a fitness function. Usually, the fitness is measured indirectly in terms of a set of fitness cases or the training data

$$D = \{(\mathbf{x}_c, y_c)\}_{c=1}^{N}, \tag{2}$$

and the individual can be considered as a model of the unknown process f generating the data:

$$y_c = f(\mathbf{x}_c) + \epsilon(\mathbf{x}_c), \tag{3}$$

where the noise $\epsilon(\mathbf{x}_c)$ is assumed to be zero-mean Gaussian. The next generation of individuals is then selected according to their fitness values on the data D. New generations are produced repeatedly until the termination condition is satisfied.

We propose to guide the evolutionary process by the Bayesian formula. The *best* model can be defined as the *most probable* model, given the data D plus the prior knowledge on the problem domain. Bayes theorem provides a direct method for calculating such probabilities. It states that the posterior probability of a model A is

$$P(A|D) = \frac{P(D|A)P(A)}{P(D)}. \tag{4}$$

$P(A)$ is the prior probability distribution for the models, $P(D|A)$ is the likelihood of the model for the data, and $P(D)$ is a normalizing constant and computed as:

$$P(D) = \sum_{A \in \mathcal{A}} P(D|A)P(A), \tag{5}$$

where $\mathcal{A}$ is the space of all possible models.

Let us define the fitness value of model A_i as its posterior probability $P_g(A_i|D)$ computed with respect to the gth population:

$$P_g(A_i|D) = \frac{P(D|A_i)P_{g-1}(A_i)}{\sum_{A_j \in \mathcal{A}(g)} P(D|A_j)P_{g-1}(A_j)}, \tag{6}$$

1. **(Prior distribution)** Generate the initial population $\mathcal{A}(0)$ of μ individuals from the prior distribution $P_0(A)$. Set $g \leftarrow 1$.
2. **(Likelihood)** Estimate the likelihoods $P(D|A)$ of individuals A in $\mathcal{A}(g)$ by evaluating their raw fitness on data D.
3. **(Posterior distribution)** Use Bayes rule to compute the posterior distribution of A_i in $\mathcal{A}(g)$:

   $$P_g(A_i|D) = \frac{P(D|A_i)P_{g-1}(A_i)}{\sum_{A_j \in \mathcal{A}(g)} P(D|A_j)P_{g-1}(A_j)}.$$

   Update the best model as

   $$A_{MAP}^g = \max\{A_{MAP}^{g-1}, \arg\max_{A_i \in \mathcal{A}(g)} \{P_g(A_i|D)\}\}.$$

4. **(Variation)** Generate λ offspring into $\mathcal{A}'(g)$ by sampling from the expected offspring distribution

   $$P'_{g+1}(A'|D) = \sum_{A \in \mathcal{A}(g)} P_g(A|D)P(A'|A),$$

   where the transition probability $P(A'|A)$ is determined by P_S, P_R, and P_M of variation operators.
5. **(Selection)** Select μ individuals from $\mathcal{A}'(g)$ into the next generation $\mathcal{A}(g+1)$ according to the acceptance probability $P_A(A')$.
6. **(Revision of priors)** If the termination condition is met, return $P_g(A|D)$ and A_{MAP}. Otherwise, update the priors as

   $$P_g(A) = h(P_{g-1}(A), P_g(A|D)).$$

   set $g \leftarrow g+1$, and continue with step 2.

Figure 1: Outline of the canonical Bayesian evolutionary algorithm (BEA). For more details and variations, see text.

where $P(D|A_i)$ is the likelihood and $P_{g-1}(A_i)$ is the prior degree of belief in A_i as of generation $g-1$.

Note in equation (6) that the posterior probability is approximated by a fixed-size population $\mathcal{A}(g)$ which is typically a small subset of the entire model space $\mathcal{A}$. The evolutionary inference step from generation $g-1$ to g is then considered to induce a new fitness distribution $P_g(A|D)$ from $P_{g-1}(A)$ following Bayes rule.

The posterior distribution $P_g(A|D)$ is then used to generate its offspring. If we denote $P(A'|A)$ the probability of offspring A' being generated from parent A, the expected probability distribution of the next generation can be estimated as

$$P'_{g+1}(A'|D) = \sum_{A \in \mathcal{A}(g)} P_g(A|D)P(A'|A), \tag{7}$$

where the transition probability $P(A'|A)$ is further decomposed into the probabilities of variation operators being applied to:

$$P(A'|A) = \sum_{B \in \mathcal{A}(g)} P_S(B|A)$$

$$\left(\sum_{A^+ \in \mathcal{A}} P_R(A^+|A,B) P_M(A'|A^+) \right). \quad (8)$$

Here $P_S(B|A)$ denotes the probability of individual B in $\mathcal{A}(g)$ being selected as a mate for A. $P_R(A^+|A,B)$ is the probability of mating A and B to produce A^+. $P_M(A'|A^+)$ represents the probability of $A+$ being mutated into A'.

By sampling from the expected distribution $P'_{g+1}(A'|D)$ we generate the offspring population $\mathcal{A}'(g)$ of size λ. By selecting from $\mathcal{A}'(g)$ the μ individuals ($\mu \leq \lambda$) with acceptance probability of $P_A(A')$, we finally generate the parent population $\mathcal{A}(g+1)$ of the next generation. In order for the information gained at generation g to get collected, we revise the priors as a function of the current posterior probabilities:

$$P_g(A) = h\left(P_{g-1}(A), P_g(A|D)\right). \quad (9)$$

At each generation g, the Bayesian evolutionary algorithm (BEA) keeps the best individual which is the maximum a posteriori (MAP) model in the population:

$$A^g_{MAP} = \max\left\{ A^{g-1}_{MAP}, \arg\max_{A_i \in \mathcal{A}(g)} \{P_g(A_i|D)\} \right\}. \quad (10)$$

As a result, the Bayesian evolutionary algorithm produces three different outputs: the posterior distribution $P_g(A|D)$ and the best model A^g_{MAP} as well as the population $\mathcal{A}(g)$. The algorithm is summarized in Figure 1.

The behavior of the Bayesian evolutionary algorithm is demonstrated experimentally on a simple function approximation problem. The task is to build a model of 10 data points which were generated from the target function $f(x) = \frac{1}{2}(x^3 - x^2 + 1)$. We use neural trees [15] as model structures whose architectures and connection weights are to be evolved. The evolutionary process is shown in Figures 2-4 where the 10 best models for each generation g ($g = 1, 10, 20$) are shown. Population size was 100. It can be observed that, as generation goes on, the models tend to converge to more accurate ones. The very first models at generation 1 were generated at random using a uniform prior distribution over tree size and weight values.

In the following two sections, we describe two variants of the canonical Bayesian evolutionary algorithm to demonstrate the applicability of the Bayesian framework for evolutionary computation.

3 Bayesian Evolution for Program Growth Control

Statistical theories suggest that models which are too simple lack sufficient learning capability while models which are too complex may generalize poorly on unseen data. In evolving models, what we need in practice is a general mechanism that can flexibly control the model complexity to find

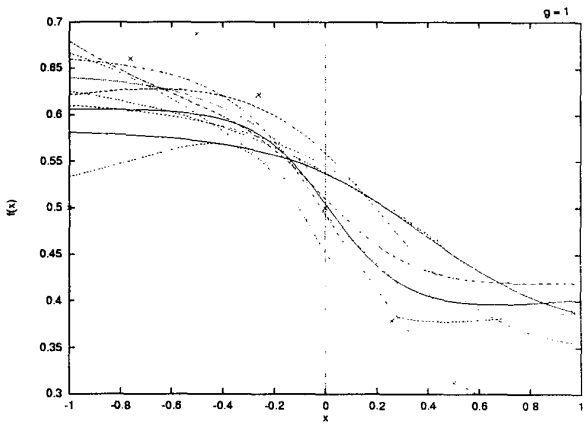

Figure 2: *Ten best models at generation* $g = 0$.

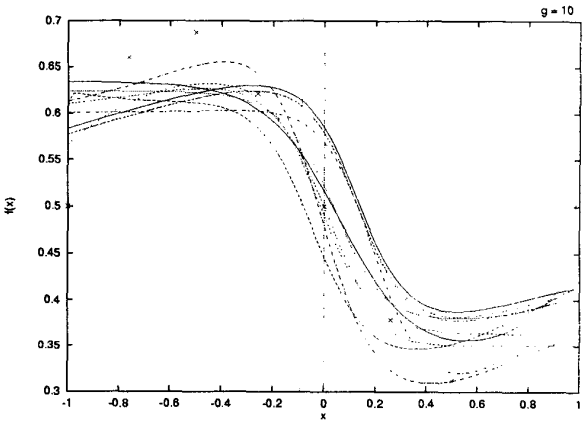

Figure 3: *Ten best models at generation* $g = 10$.

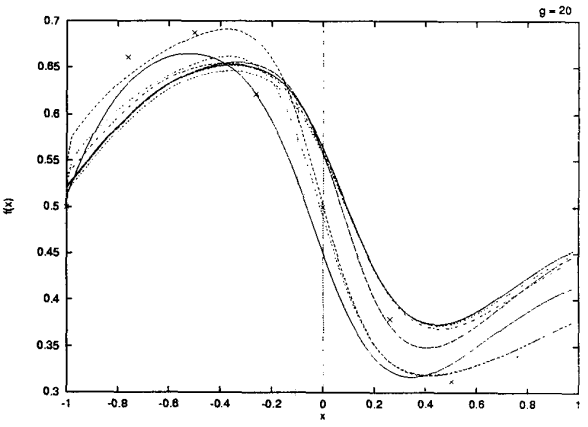

Figure 4: *Ten best models at generation* $g = 20$.

the most parsimonious model while satisfying the desirable training accuracy.

The objective is to find an individual or model A whose evaluation $f_A(\mathbf{x})$ best approximates the desired relation $\tilde{f}(\mathbf{x})$ given an input $\mathbf{x}$. The data misfit of the individual for the data set D is measured by

$$E(D|A) = \frac{1}{N}\sum_{c=1}^{N}(y_c - f_A(\mathbf{x}_c))^2, \tag{11}$$

where N is the number of fitness cases. Considering the individual as a Gaussian model of the data, the likelihood of the model for the data is described by

$$P(D|A) = \frac{1}{Z(\beta)}\exp(-\beta E(D|A)), \tag{12}$$

where $Z(\beta)$ is a normalizing constant, and β is a positive constant determining the sensitivity of the probability to the error value.

The Bayesian evolutionary framework (6) states that the overall fitness of individuals should be measured in terms of both the likelihood of the model for the data, $P(D|A)$, and the prior probability of model, $P(A)$. By defining some prior probability distribution on the models and by making use of the concept of code length in coding theory, we can derive a Bayesian evolutionary algorithm for building both accurate and parsimonious models for the data.

First, we make use of the fact that since $P(D)$ in the Bayes formula $P(A|D) = P(D|A)P(A)/P(D)$ is the same for all models, for the purposes of model comparison, we need only compute

$$P(D|A)P(A) \tag{13}$$

for all A and then choose the model that maximizes this value. Thus it is reasonable to define the evolutionary computation as the maximization of the posterior probability:

$$\begin{aligned} A_{best} &= \arg\max_{A_i \in \mathcal{A}} \{P(A_i|D)\} && (14)\\ &= \arg\max_{A_i \in \mathcal{A}} \{P(D|A_i)P(A_i)\} && (15) \end{aligned}$$

where arg function takes its argument as its result.

According to coding theory [2], the probability of models can be expressed as model complexity or code length: if $P(\mathbf{x})$ is given, then its code length is given as $L(P(\mathbf{x})) = -\log(P(\mathbf{x}))$. Maximizing $P(D|A)P(A)$ is thus equivalent to minimizing code length:

$$\begin{aligned} L(A|D) &= L(P(D|A)P(A)) \\ &= -\log(P(D|A)P(A)) \\ &= L(D|A) + L(A), \end{aligned} \tag{16}$$

where $L(D|A) = -\log P(D|A)$ and $L(A) = -\log P(A)$. Here $L(D|A)$ is the code length of the data when encoded using the model A as a predictor for the data D, and $L(A)$ is the length of the model itself. This leads to the minimum description length (MDL) principle [9]. The idea is to estimate the simplest density that has high likelihood by minimizing the total length of the description of the data:

$$\begin{aligned} A_{best} &= \arg\min_{A_i \in \mathcal{A}} \{L(A_i|D)\} && (17)\\ &= \arg\min_{A_i \in \mathcal{A}} \{L(D|A_i) + L(A_i)\}. && (18) \end{aligned}$$

The adaptive Occam method proposed in [14] can be regarded as a method for effective balancing of accuracy and parsimony of evolved structures. It measures the fitness of a program A given a training set D in its most general form as

$$\begin{aligned} F(A|D) &= F_D + F_A \\ &= \beta E(D|A) + \alpha C(A), \end{aligned} \tag{19}$$

where the parameters α and β control the trade-off between complexity $C(A)$ and fitting error $E(D|A)$ of the program.

To balance the parsimony with accuracy dynamically, the adaptive Occam method fixes the error factor β at each generation and change the complexity factor α adaptively. Let $E_i(g)$ and $C_i(g)$ denote the error and complexity of ith individual at generation g. In case of neural trees [15], it is defined as the sum of the numbers of layers, units, and weights in the tree:

$$C_i(g) = L(A_i^g) + U(A_i^g) + W(A_i^g). \tag{20}$$

In essence, these are the parameters determining the structure and size of the models.

The fitness of an individual i at generation g is defined as follows:

$$F_i(g) = E_i(g) + \alpha(g)C_i(g), \tag{21}$$

where $0 \le E_i(g) \le 1$ and $C_i(g) > 0$ are assumed. Here $\alpha(g)$ is called the adaptive Occam factor and expressed as

$$\alpha(g) = \begin{cases} \frac{1}{N^2}\frac{E_{best}(g-1)}{\hat{C}_{best}(g)} & \text{if } E_{best}(g-1) > \epsilon \\ \frac{1}{N^2}\frac{1}{E_{best}(g-1)\cdot\hat{C}_{best}(g)} & \text{otherwise,} \end{cases} \tag{22}$$

where N is the size of training set. User-defined constant ϵ specifies the maximum training error allowed for the final solution. $E_{best}(g-1)$ is the error of the best model of generation $g-1$. $\hat{C}_{best}(g)$ is the complexity of the best model at generation g estimated at generation $g-1$. These are used to balance the error and complexity terms to obtain models as parsimonious as possible while not sacrificing their accuracy.

The performance of the adaptive Occam method to complexity control has been studied in [14] and the results are

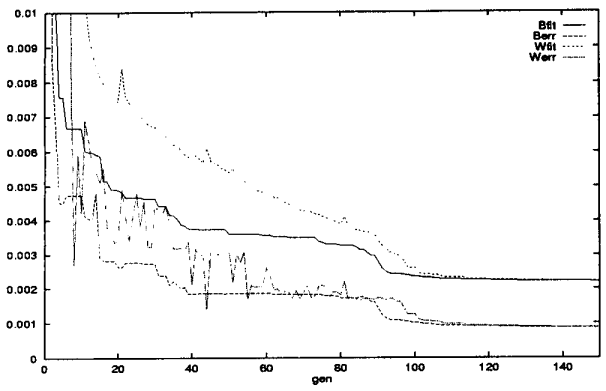

Figure 5: Evolution of error and complexity of the models for the laser data. The top two curves plot the overall fitness (error + complexity penalty) of the worst and best individuals, and the bottom two curves show the error factor of the worst and best individuals, respectively.

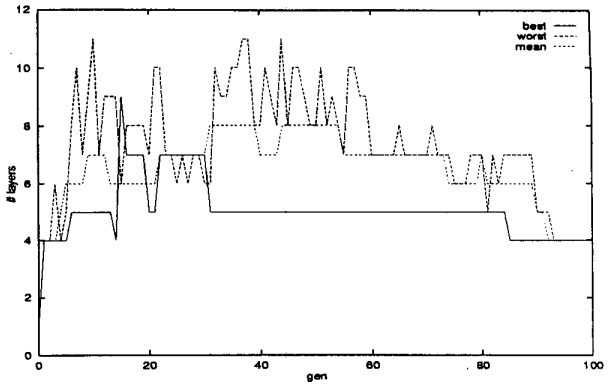

Figure 6: Evolution of complexity in terms of the number of layers for the laser data. The top and bottom curves show the complexity of the worst and best individuals, whereas the middle curve is the mean complexity of the population. The curves demonstrate the desired effect of the adaptive Bayesian approach to complexity control; it promotes the trees to grow when significant error reduction is required, while it prefers smaller trees to larger ones when their errors are comparable.

reproduced here in Figures 5 and 6. The test domain was the laser intensity time series [5] and the model class evolved was the neural trees [15].

Figure 5 plots the evolution of the overall fitness (error + complexity penalty) and its error factor of the best and worst individuals in each generation. The corresponding evolution of tree complexity in terms of the number of layers (depth) was shown in Figure 6. Comparing both figures, the tendency is observed that the program size first grows and then shrinks during which the error of the best individual steadily decreases. This demonstrates the desired effect of the Occam's Razor implemented in the adaptive Bayesian approach for complexity control; it promotes the trees to grow when significant error reduction is required, while it prefers smaller trees to larger ones when their errors are comparable.

4 Bayesian Evolution with Incremental Data Inheritance

Another implication of the Bayesian formulation of evolutionary computation (6) is that it provides a sequential way to calculate the posterior probability $P(A|D^g)$ based on its prior probability $P(A)$ as new data D^g are observed. This fact can be used to derive an incremental evolutionary algorithm based on Bayesian inference.

Let us consider a Bayesian evolution with the following properties:

1. *Separate data sets.* Each model A_i^g in the population has its own data D_i^g, where D_i^g is a subset of the original data set $D^{(N)}$ of size N.

2. *Same data size.* The sizes of data sets are equal for all models at the same generation, but their elements may be different:

$$|D_i^g| = |D_j^g|,\ D_i^g \neq D_j^g,\ \ i,j = 1,...,M. \qquad (23)$$

3. *Data inheritance.* Just as models inherit their structural features from their parents, data sets D_i^g of models inherit partial data from their parents' D_p^{g-1} and D_q^{g-1}, where p and q are parents of i.

4. *Monotonic growth.* Data sets grow monotonically as generation goes on, i.e. $|D_i^g| > |D_i^{g-1}|$.

All these features together constitute the incremental data inheritance (IDI) method [12]. The monotonicity is important to ensure the method ultimately uses all the given data. This guarantees the method not to lose known information about the problem domain. This is contrasted with the dynamic subset selection method [4] where only one common data set for each generation exists, data are chosen randomly, and the data size remains fixed through the generations.

The Bayesian evolutionary algorithm with incremental data inheritance is then formulated as:

$$A_{best}^{(g)} = \max_{g \leq g_{max}} \text{argmax}_{A_i^g, D_i^g} P_i(g) \qquad (24)$$

where $A_i^g \in \mathcal{A}$, $D_i^g \subset D^{(N)}$, and the fitness function is defined as:

$$P_i(g) \equiv \frac{P(D_i^g|A_i^g)P(A_i^g)}{\sum_{A_j^g, D_j^g} P(D_j^g|A_j^g)P(A_j^g)}. \qquad (25)$$

Note that each model A_i^g is now associated with its own data set D_i^g with its size depending on the generation number.

For a convenient implementation of the Bayesian method we take the minus logarithm of $P_i(g)$ and use it as the fitness function

$$F_i(g) = -\log(P_i(g)). \tag{26}$$

Then the evolutionary algorithm is reformulated as a minimization process

$$\begin{aligned} A_{best}^{(g)} &= \min_{g \le g_{max}} \operatorname{argmin}_{A_i^g, D_i^g} F_i(g), \\ \min F_i(g) &= \min\{-\log P(D_i^g|A_i^g) - \log P(A_i^g)\} \\ &= \min\{L(D_i^g|A_i^g) + L(A_i^g)\}, \end{aligned} \tag{27}$$

where $L(D_i^g|A_i^g)$ and $L(A_i^g)$ are code lengths for the data and the model [2].

The exact calculation of code lengths requires the true probability distribution of underlying data structure which is in most real situations unknown. Instead, we define an adaptive fitness function in its most general form as

$$\begin{aligned} F_i(g) &= L(D_i^g|A_i^g) + L(A_i^g) \\ &= \beta F_D + \alpha F_A \\ &= E(D_i^g|A_i^g) + \alpha(g) C(A_i^g), \end{aligned} \tag{28}$$

where $E(D_i^g|A_i^g)$ and $C(A_i^g)$ are the measures for error and complexity of the model, and the parameter $\alpha(g)$ balances these two factors as follows

$$\alpha(g) = \begin{cases} \frac{1}{N_g^2} \frac{E_{best}(g-1)}{\hat{C}_{best}(g)} & \text{if } E_{best}(g-1) > \epsilon \\ \frac{1}{N_g^2} \frac{1}{E_{best}(g-1) \cdot \hat{C}_{best}(g)} & \text{otherwise.} \end{cases} \tag{29}$$

This is a generalization of the adaptive Occam method in that N_g is now a variable, rather than fixed, scheduled to increase monotonically as a function of generation g.

In the experiments, the mean squared error is used as the error measure:

$$E(D_i^g|A_i^g) = \frac{1}{N_g} \sum_{c=1}^{N_g} (y_c - f(\mathbf{x}_c; A_i^g))^2, \tag{30}$$

where $(\mathbf{x}_c, y_c) \in D_i^g$ are training cases, N_g is the size of the training set at generation g, and $f(\mathbf{x}_c; A_i^g)$ is the output of the model A_i^g for input $\mathbf{x}_c$. The model complexity can be defined in various ways since the adaptive Occam method uses relative complexity rather than absolute values.

To see the effect of the Bayesian evolutionary algorithm with incremental data inheritance, we measured the computational time as a product of the population size and the data size for each generation. The performance of the methods in this pseudo-cpu time is compared in Figure 7. The

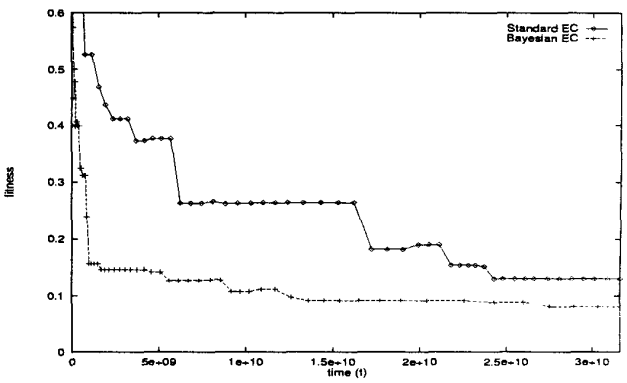

Figure 7: Comparison of the convergence behaviors for the standard and Bayesian evolutionary computations. The data was the laser intensity time series and the model class evolved was neural trees.

performance of the Bayesian evolutionary computation improved very fast, especially in the early generations, and approached a fitness level of approximately 0.1. In contrast, the fitness of the standard evolutionary computation converged to around 0.27 when the Bayesian method with IDI reached 0.1. Though the standard EC could eventually reach the performance level of the Bayesian EC if run further, but it took so long that this cannot be seen in Figure 7. In effect, the Bayesian evolutionary algorithm with incremental data inheritance achieved for this problem a speed-up of up to an order of magnitude compared to the standard evolutionary algorithm.

5 Concluding Remarks

We have proposed a Bayesian framework for evolutionary computation and presented specific examples of the Bayesian evolutionary algorithms that are designed to improve the performance of existing evolutionary algorithms.

Specifically, we have shown that the adaptive Occam method [14] conforms to the Bayesian evolutionary framework. We also have shown that the incremental data inheritance method [12] is a natural, Bayesian extension to the conventional evolutionary algorithms that evaluates fitness based on training samples.

The Bayesian approach to evolutionary computation provides a number of important features. One is that by viewing the evolutionary computation process as a Bayesian inference, principled techniques for driving evolutionary dynamics of conventional evolutionary algorithms can be developed. The BEA with the adaptive Occam method and the BEA with incremental data inheritance are just two examples.

Another interesting feature of Bayesian inference is that it allows background knowledge in the problem domain to be incorporated in a formal way. This property is important especially for solving real-life problems of practical interest and needs further work.

Decision-making can be made more robust by combining multiple models instead of a single model [10, 11]. Evolutionary algorithms usually generate a number of models. The Bayesian evolutionary framework provides a principled way to combine multiple models to build a committee machine [13].

Finally, the Bayesian evolutionary framework appears to be general enough to incorporate existing evolutionary algorithms as special approximations to the full Bayesian approach. The generality of the framework is crucial for exploration and comparative analysis of various evolutionary algorithms.

Acknowledgments

This research was supported in part by the Korea Science and Engineering Foundation (KOSEF) under Grant 981-0920-350-2 and by the Korea Ministry of Science and Technology through KISTEP under Grant BR-2-1-G-06. Thanks to David Fogel, Heinz Mühlenbein, and Gerhard Paaß for stimulating discussions and comments on the draft version of this paper.

Bibliography

[1] T. Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, 1996.

[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley, 1991.

[3] D. B. Fogel (ed.) *Evolutionary Computation: The Fossil Record*, IEEE Press, 1998.

[4] C. Gathercole and P. Ross, "Small populations over many generations can beat large populations over few generations in genetic programming," *Genetic Programming 1997*, pp. 111-118, 1997.

[5] H. Hübner, C. O. Weiss, N. B. Abraham, and D. Tang, "Lorenz-like chaos in the NH3-FIR laser," In A. Weigend and N. Gershenfeld (Eds.), *Time Series Prediction: Forecasting the Future and Understanding the Past*, Addison-Wesley, pp. 73-104, 1993.

[6] H. Mühlenbein and T. Mahnig, "FDA – An evolutionary algorithm for additively decomposed functions," *The 1999 Congress on Evolutionary Computation* (CEC99), IEEE Press, 1999 (to appear).

[7] H. Mühlenbein and Schlierkamp-Voosen, "The science of breeding and its application to the breeder genetic algorithm," *Evolutionary Computation*, vol. 1, no. 4, pp 335-360, 1994.

[8] S. J. Press, *Bayesian Statistics: Principles, Models, and Applications*, Wiley, 1989.

[9] J. Rissanen, "Stochastic complexity and modeling," *The Annals of Statistics*, vol. 14, pp. 1080-1100, 1986.

[10] X. Yao and Y. Liu, "Making use of population information in evolutionary artificial neural networks," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 28B, No. 2, 1998.

[11] B.-T. Zhang and J.-G. Joung, "Enhancing robustness of genetic programming at the species level," *Genetic Programming 1997*, Morgan Kaufmann, pp. 336-342, 1997.

[12] B.-T. Zhang and J.-G. Joung, "Genetic programming with incremental data inheritance," *The 1999 Genetic and Evolutionary Computation Conference 1999* (GECCO-99), Orlando, Florida, Morgan Kaufmann, 1999 (to appear).

[13] B.-T. Zhang and J.-G. Joung, "Time series prediction using committee machines of evolutionary neural trees," *The 1999 Congress on Evolutionary Computation* (CEC99), Washington, D.C., IEEE Press, 1999 (to appear).

[14] B.-T. Zhang and H. Mühlenbein, "Balancing accuracy and parsimony in genetic programming," *Evolutionary Computation*, vol. 3, pp. 17-38, 1995.

[15] B.-T. Zhang, P. Ohm, and H. Mühlenbein, "Evolutionary induction of sparse neural trees," *Evolutionary Computation*, vol. 5, no. 1, pp. 213-236, 1997.