A Bayesian Framework for Evolutionary Computation

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

Presented at World Congress on Evolutionary Computation (CEC-99), Washington D.C.

Outline

- Bayesian formulation of evolutionary computation
- Two Bayesian evolutionary algorithms (BEAs):
  - BEA for model complexity control
  - BEA for accelerating convergence
- Performance of the Bayesian evolutionary algorithms
- Concluding remarks

Introduction

- Bayesian probabilities are often used in statistical inference for specifying a priori knowledge and combining it with available data via Bayes theorem.
- We propose to use the Bayesian inductive principle as a theoretical framework for evolutionary computation:
  - The *best* or *fittest* individual can be defined as the *most probable* model, given data *D* (fitness cases) plus the priori knowledge on the problem domain.
  - Evolutionary computation (EC) is then viewed as an iterative process of *generating the individuals of ever higher posterior probabilities* from the priors and the observed data.

Bayesian Formulation of EC

- Bayes' rule for combining priors and likelihoods:

$$P(A \mid D) = \frac{P(D \mid A)P(A)}{P(D)} = \frac{P(D \mid A)P(A)}{\int_A P(D \mid A)P(A)}$$

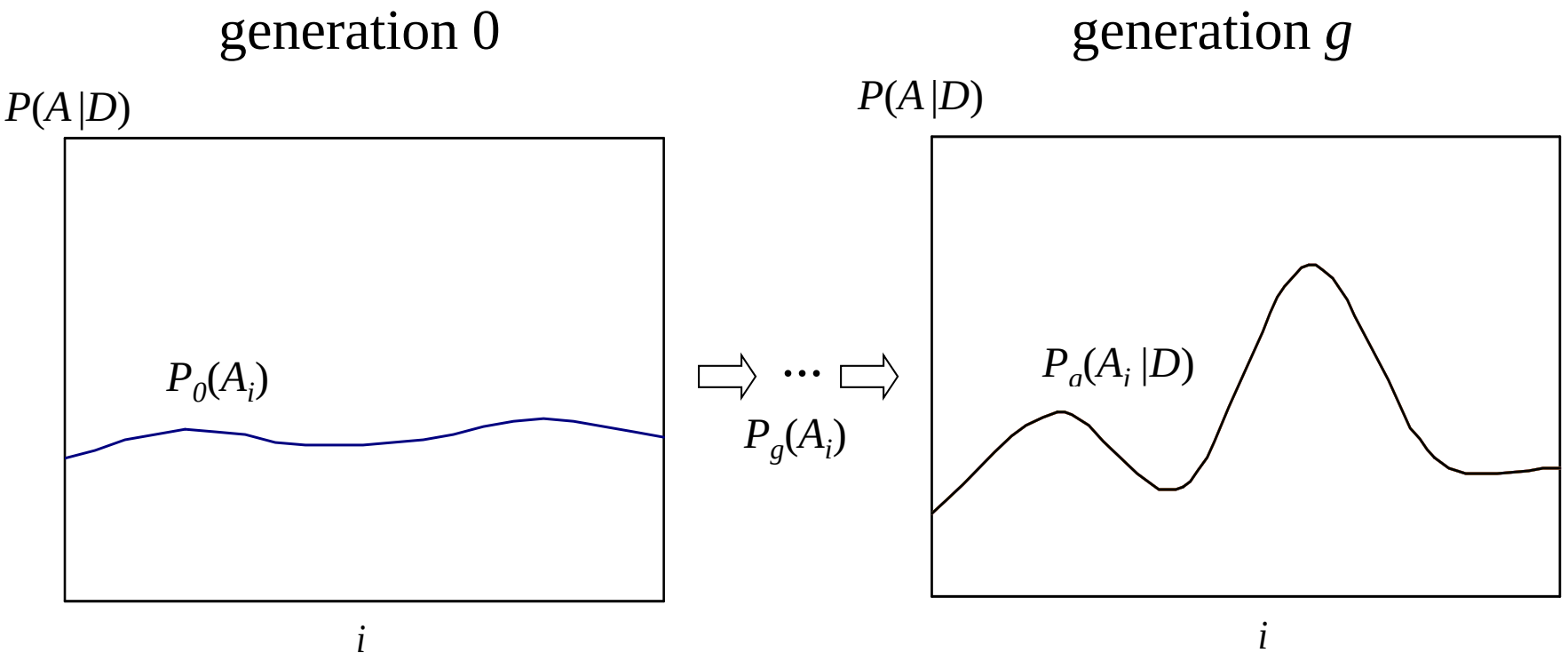
- Evolutionary computation (EC) can estimate the posterior probability of model A_i using the population $\boldsymbol{A}(g)$:

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

- The fittest model for the Bayesian EC to find is:

$$A_{MAP}^{g} = \min_{g \le g_{\max}} \arg\max_{A_i \in A(g)} \{P_g(A_i \mid D)\}$$

Bayesian Evolutionary Computation



Bayesian Evolutionary Algorithm (BEA)

1. Sample M individuals A_i ($i=1,\ldots,M$) from $P_0(A)$. Set $g=1$.

2. Compute the posterior fitness $P_i(g) = P_g(A_i|D)$ for $i=1,\ldots,M$:

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

3. Generate offspring A_i' by sampling from the posterior distribution using variation operators, such as mutation and recombination:

$$P'_{g+1}(A_i^{'} \mid D) = \sum_{A_i \in A(g)} P_g(A_i \mid D) P(A_i^{'} \mid A_i)$$

4. Select the individuals into the next generation with acceptance probability

$$a_g(A_i^{'} \mid A_i) = \min\left\{1, \frac{P_g(A_i^{'} \mid D)}{P_g(A_i \mid D)}\right\}$$

5. Revise the priors $P_g(A) = h(P_{g-1}(A), P_g(A \mid D))$.

Set $g=g+1$ and go to step 2.

Implementation of the BEA as an Adaptive Fitness Function

$$F_i(g) = -\log P_i(g) = -\log P_g(A_i \mid D)$$

$$A_{best}^{g} = \min_{g \le g_{\max}} \arg\min_{A_i \in A(g)} F_i(g)$$

$$F_i(g) = -\log P(D \mid A_i) - \log P(A_i)$$

$$P(D \mid A_i) = \frac{1}{Z_D(\beta)} \exp(-\beta E(D \mid A_i))$$

$$P(A_i) = \frac{1}{Z_A(\alpha)} \exp(-\alpha C(A_i))$$

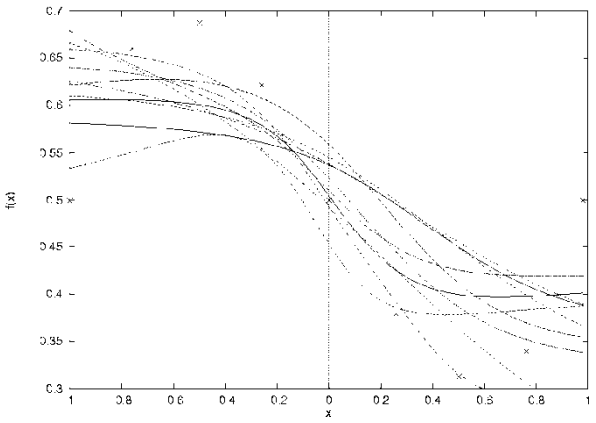
$$\begin{aligned} F_i(g) &= F_D + F_A \\ &= \beta E(D \mid A_i) + \alpha C(A_i) \\ &= E_i(g) + \alpha(g) C_i(g) \end{aligned}$$

: adaptive fitness function

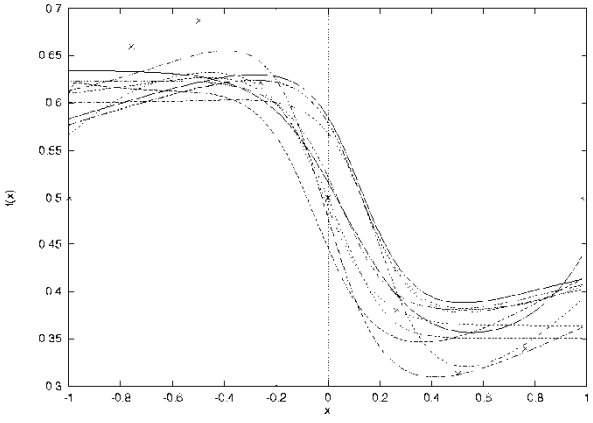
[Zhang and Muehlenbein, ECJ-95]

Simple Function Approximation

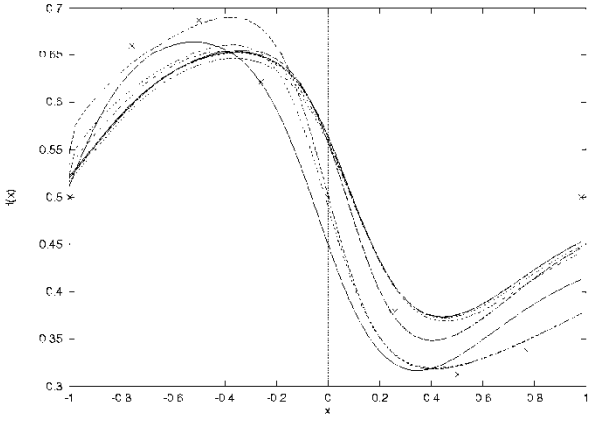
- Target function: $f(x) = \frac{1}{2}(x^3 - x^2 + 1)$
- Model structure: neural tree
- Population size: 100
- Ten best models at generation g = 1, 10, 20



g = 1



g = 10



g = 20

BEA for Program Growth Control

- According to coding theory, maximizing the posterior probability is equivalent to minimizing the code length

$$A_{best} = \arg\max_{A_i \in A}\{P(A_i \mid D)\} = \arg\max_{A_i \in A}\{P(D \mid A_i)P(A_i)\}$$

$$= \arg\min_{A_i \in A}\{L(A_i \mid D)\} = \arg\min_{A_i \in A}\{L(D \mid A_i) + L(A_i)\}$$

where $L(P(x)) = -\log(P(x))$

Adaptive Occam's Razor

- Effective balancing of accuracy and parsimony of evolved structures

$$F_i(g) = F_D + F_A = L(D \mid A_i^g) + L(A_i^g)$$
$$= \beta E(D \mid A_i^g) + \alpha C(A_i^g)$$
$$= E_i(g) + \alpha(g) C_i(g)$$

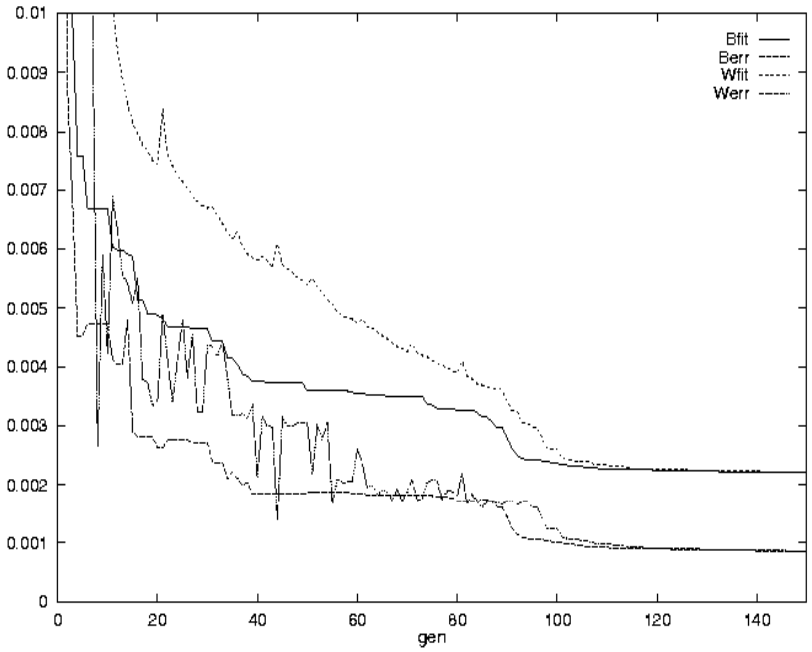
where

$$E_i(g) = \frac{1}{N} \sum_{c=1}^{N} (y_c - f(\mathbf{x}_c; A_i^g))^2$$: fitting error

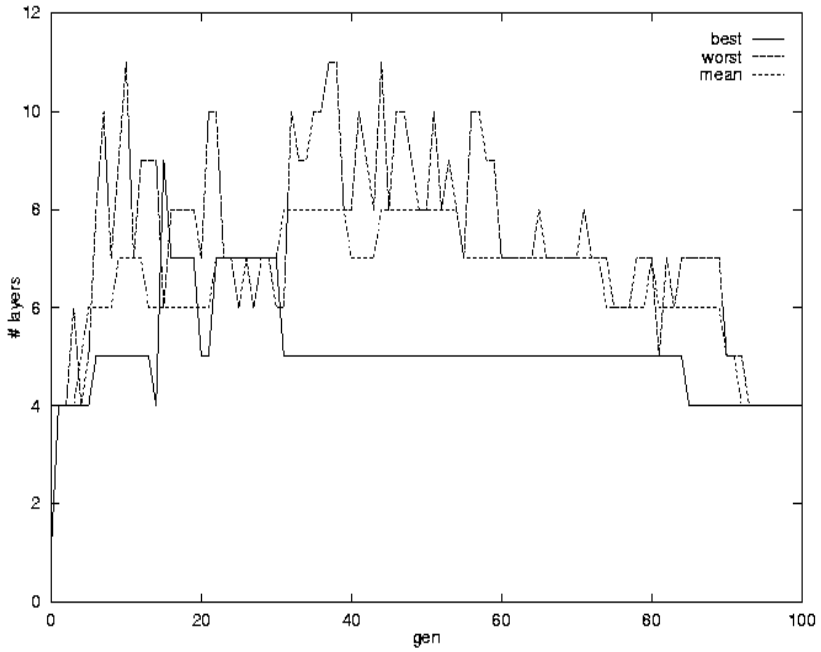
$$C_i(g) = l(A_i^g) + u(A_i^g) + w(A_i^g)$$: model complexity

Performance of AOR

- For the laser data

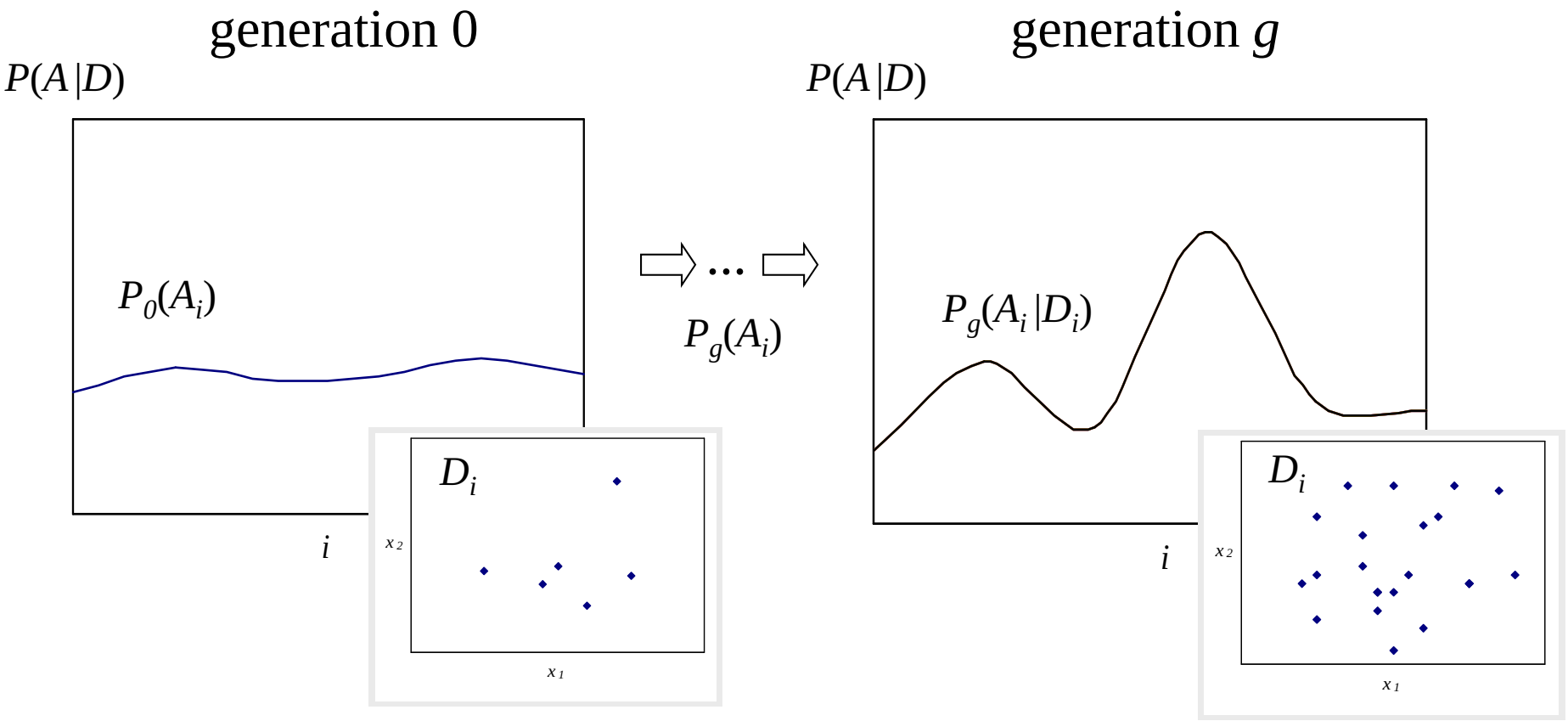


Error and complexity



The number of layers

Bayesian Evolution with IDI



$$\left|D_i^1\right| < \left|D_i^2\right| < \cdots < \left|D_i^{g_{\max}}\right| = \left|D\right|$$

Incremental Data Inheritance (IDI)

- *Separate data sets.* Each model in the population has its own data.
- *Same data size.* The sizes of data sets are equal for all models at the same generation, but their elements may be different:

$$\left|D_i^g\right| = \left|D_j^g\right|, \quad D_i^g \neq D_j^g, \quad i, j = 1,\ldots,M$$

- *Data inheritance.* Just as models inherit their structural features from their parents, data sets of models inherit partial data from their parents.
- *Monotonic growth.* $\left|D_i^g\right| > \left|D_i^{g-1}\right|$

Bayesian Evolution with IDI

- Bayesian formulation

$$A_{best}^{g} = \min_{g \le g_{\max}} \arg\max_{A_i^g, D_i^g} P_i(g)$$

$$P_i(g) \equiv \frac{P(D_i^g \mid A_i^g) P(A_i^g)}{\sum_{A_j^g} P(D_j^g \mid A_j^g) P(A_j^g)}$$

- Reformulation as a minimization process

$$A_{best}^{g} = \min_{g \le g_{\max}} \arg\min_{A_i^g, D_i^g} F_i(g)$$

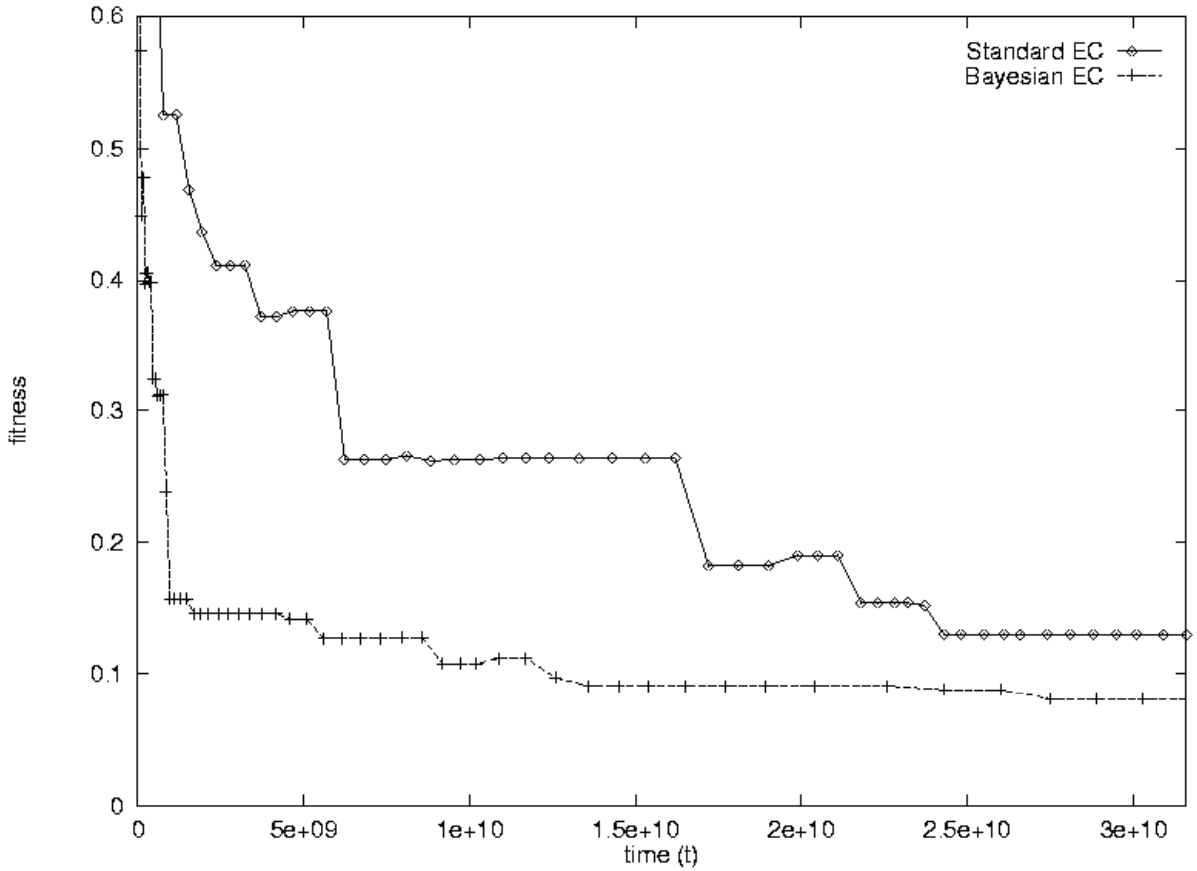
- Adaptive fitness function

$$F_i(g) = F_D + F_A = \beta E(D_i^g \mid A_i^g) + \alpha C(A_i^g)$$
$$= E_i(g) + \alpha(g) C_i(g)$$

$$E_i(g) = \frac{1}{N_g} \sum_{c=1}^{N_g} (y_c - f(\mathbf{x}_c; A_i^g))^2$$

Performance of the BEA with IDI:

Results on the Laser Time Series Data



Concluding Remarks

- A Bayesian framework for developing principled techniques for driving the dynamics of evolutionary computation.
- Two applications of Bayesian evolutionary computation: (1) BEA with AOR, (2) BEA with IDI
- Background knowledge in the problem domain can be incorporated in a formal way.
- Generality of the framework for the design and analysis of evolutionary algorithms.