Program Evolution by Integrating EDP and GP

Kohsuke Yanai and Hitoshi Iba

Dept. of Frontier Informatics, Graduate School of Frontier Science,
The University of Tokyo.
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8654, Japan
{yanai,iba}@iba.k.u-tokyo.ac.jp

Abstract. This paper discusses the performance of a hybrid system which consists of EDP and GP. EDP, Estimation of Distribution Programming, is the program evolution method based on the probabilistic model, where the probability distribution of a program is estimated by using a Bayesian network, and a population evolves repeating estimation of distribution and program generation without crossover and mutation. Applying the hybrid system of EDP and GP to various problems, we discovered some important tendencies in the behavior of this hybrid system. The hybrid system was not only superior to pure GP in a search performance but also had interesting features in program evolution. More tests revealed how and when EDP and GP compensate for each other. We show some experimental results of program evolution by the hybrid system and discuss the characteristics of both EDP and GP.

1 Introduction

1.1 Program Evolution Using Probability Distribution

Recently, attention has been focused on evolutionary algorithms based on a probabilistic model. These are called Estimation of Distribution Algorithms (EDA) [Larranage and Lozano02] or Probabilistic Model Building Genetic Algorithms. EDA is a search method that eliminates crossover and mutation from the Genetic Algorithm (GA) and places more emphasis on the relationship between gene loci. Much research has been performed on this. However, there have been almost no researches on its application to program evolution problems (see Section 4.3).

We have proposed EDP, Estimation of Distribution Programming, based on a probability distribution expression using a Bayesian network [Yanai03a] . EDP is a population based search method and evolves a population by repeating estimation of distribution and program generation. In program evolution experiments, EDP showed different characteristics from GP and could solve GP's weak problems. On the other hand, in GP standard problems, for example, a function regression problem or a boolean problem, GP was far superior to EDP. Therefore, we built the hybrid system of GP and EDP and tried to test it in a function regression problem. If the performance of this hybrid system is worse than pure GP, we can conclude that EDP is useless in this GP standard problem. However, contrary to our expectation, experimental results indicated interesting

K. Deb et al. (Eds.): GECCO 2004, LNCS 3102, pp. 774–785, 2004.

tendencies. Although pure GP was superior in younger generations, the performance of the hybrid system overtook GP on the evolution and was better in later generations.

We were interested in the robustness of this hybrid system's make-up and what causes the "overtaking." This paper discusses the performance and the characteristics of the hybrid system according to various experiments and considers GP's and EDP's defects and how GP and EDP compensate for each other.

This paper is organized as follows: Section 2 describes the algorithm of the hybrid system and the details of estimation of distribution and program generation. Section 3 indicates the performance difference due to the hybrid ratio of GP to EDP and discusses whether the "overtaking" is significant. In Section 4, we show experiments of 2 systems: a system which changes the hybrid ratio for each generation and a system which estimates distribution independent of a past state, and thoroughly analyze the systems. On the basis of these three experiments, an important conclusion about EDP's function is reached. Section 5 summarizes this paper and considers future work.

1.2 Features of EDP

From comparative experiments with GP in a max problem [Langdon02] and a boolean 6-multiplexer problem, the following characteristics of EDP are obtained [Yanai03b].

1. In a max problem, EDP is superior to GP.
2. When adding a harmful node, which is the source of introns in a max problem, EDP is far superior to GP.
3. In a boolean 6-multiplexer problem, EDP cannot search as well as GP.
4. In both, a max problem and a boolean 6-multiplexer problem, EDP can find a better solution than a random search.
5. It is expected that EDP can control introns effectively because it keeps the occurrence probability of harmful nodes low.
6. EDP has positional restriction and useful part trees cannot shift their position, while GP's crossover can move part trees to another position in the tree.

The 6^{th} point is EDP's critical defect and makes its performance low in a boolean 6-multiplexer problem. A radical improvement is under consideration in order to eliminate this defect. The hybrid system introduced in the next Section, which is an easy extension, can overcome this difficulty. In brief, it leaves the shifting of part trees to GP's crossover.

2 Hybrid System of EDP and GP

2.1 Algorithm of Hybrid System

In this Section, we explain our hybrid system which consists of EDP and GP. This hybrid system carries out a search using the following procedure:

Step 1 Initialize a population.
Step 2 Evaluate individuals and assign fitness values.
Step 3 If a termination criterion is satisfied, then go to Step 9.
Step 4 Estimate the probability distribution.
Step 5 Use the elitist strategy.
Step 6 Generate new $rM - E_S$ individuals with GP operator.
Step 7 Generate new $(1-r)M$ individuals with EDP operator.
Step 8 Replace the population and go to Step 2.
Step 9 Report the best individual.

In Step 1, according to function node generation probability P_F and terminal node generation probability P_T $(= 1 - P_F)$, initial M individuals are generated randomly, where M is the population size. However, if tree size limitation is reached, terminal nodes are generated. For example, the probabilities of function node "+" and terminal node "x" are given:

$$\text{if tree size limitation is not reached,} \tag{1}$$

$$\begin{cases} P(X = \text{"}+\text{"}) & = P_F \times \frac{1}{N_F} \\ P(X = \text{"}x\text{"}) & = P_T \times \frac{1}{N_T} \end{cases} \tag{2}$$

$$\text{if tree size limitation is reached,} \tag{3}$$

$$\begin{cases} P(X = \text{"}+\text{"}) & = 0 \\ P(X = \text{"}x\text{"}) & = \frac{1}{N_T} \end{cases} \tag{4}$$

where N_F is the number of function nodes and N_T is the number of terminal nodes.

Next, each individual in the current population is evaluated by a fitness function and assigned its fitness value (Step 2). If a termination criterion is met, then go to Step 9. Usually a termination criterion is a previously specified maximum number of generations (Step 3).

In Step 4, superior individuals with high fitness values are selected within sampling size S_S, and a new distribution is estimated based on those selected individuals (see Section 2.3). We use the elitist strategy in Step 5, i.e., elite E_S individuals are selected from the population in the order of fitness superiority and copied to the new population, where E_S is the elite size.

In Step 6, nearly $100r\%$ $(0 \leq r \leq 1)$ of the population, precisely $rM - E_S$ individuals, is generated by standard GP operators: crossover and mutation. It selects superior individuals of GP operator's target by tournament selection with tournament size T_{gp} and performs mutation with the mutation probability P_M or crossover with the probability $1 - P_M$. Note that mutation and crossover which violate tree size limitation are not performed, and generated individuals are under tree size limitation.

Then in Step 7, the remaining $100(1-r)\%$ of the population, that is $(1-r)M$ individuals, is generated by using a newly acquired distribution (see Section 2.4). This new distribution is considered better than the previous one because it samples superior individuals in the population.

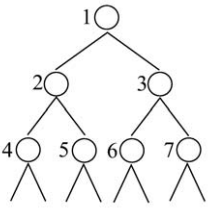

Fig. 1. Program tree.

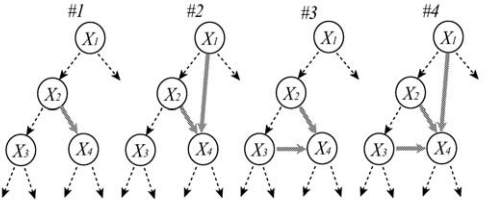

Fig. 2. Efficient network topology.

This process is repeated until a termination criterion is met. Finally in Step 9, the best individual is reported as the solution to the problem.

r is the most important parameter, it decides the system behavior and the ratio of GP to EDP in an individual generation, called the hybrid ratio. Through the combination of EDP and GP, the difficulty indicated in Section 1.2 might be overcome. However, it is not obvious whether GP gains anything from hybridization. In Section 3, we test the system performance in a function regression problem changing the hybrid ratio r from 0 to 1.

2.2 Distribution Model

We use a Bayesian network as the distribution model of programs. Values of probabilistic variables are symbols for each node in the program tree. Assign the index numbers to each node of evolving programs as in Fig. 1, the range of probabilistic variable X_i is the symbols of node i, that is, $X_i \in T \cup F$, where F is the function node set, T is the terminal node set.

For instance, assume $F = \{+, -, *, /\}$ and $T = \{x_1, x_2\}$,

$$P(X_5 = \text{"}+\text{"} | X_2 = \text{"}/\text{"}) = \frac{2}{7} \tag{5}$$

means that the conditional probability that node 5 becomes " + " is $\frac{2}{7}$ if node 2 is "/". C_i is the set of probabilistic variables which X_i is dependent on. In the former example, $C_5 = \{X_2\}$.

Although there are several efficient topologies of a Bayesian network as indicated in Fig. 2, the simplest one, that is, #1 in Fig. 2, is used for our experiments. The topology of a Bayesian network is tree-like and it is the same as the program's topology.

2.3 Estimation of Distribution

The probability distribution is updated incrementally [Baluja94] as follows:

$$P_{t+1}(X_i = x | C_i = c) = (1 - \eta)\hat{P}(X_i = x | C_i = c) + \eta P_t(X_i = x | C_i = c) \tag{6}$$

where $P_t(X_i = x|C_i = c)$ is the distribution of the t^{th} generation and $\hat{P}(X_i = x|C_i = c)$ is the distribution estimated based on superior individuals in the $(t+1)^{th}$ population, η is the learning rate which means dependence degree on the previous generation.

$\hat{P}(X_i = x|C_i = c)$ is estimated as follows. At first, S_S individuals are sampled by tournament selection with tournament size T_{edp}, and maximum likelihood estimation is performed based on these selected individuals. Therefore,

$$\hat{P}(X_i = x|C_i = c) = \frac{\sum_{j=1}^{S_S} \delta(j, X_i = x, C_i = c)}{\sum_{j=1}^{S_S} \sum_{x \in F \cup T} \delta(j, X_i = x, C_i = c)} , \tag{7}$$

where

$$\delta(j, X_i = x, C_i = c) = \begin{cases} 1 & \text{if } X_i = x \text{ and } C_i = c \\ & \quad \text{at the individual } j . \\ 0 & \text{else} \end{cases} \tag{8}$$

2.4 Program Generation

At first, the acquired distribution $P_t(X_i = x|C_i = c)$ is modified like Laplace correction [Cestnik90] by

$$P'_t(X_i = x|C_i = c) = (1 - \alpha)P_t(X_i = x|C_i = c) + \alpha P_{bias}(X_i = x|C_i = c), \tag{9}$$

where α is a constant that expresses the Laplace correction rate, $P_{bias}(X_i = x|C_i = c)$ is the probability to bias distribution. This modification makes all occurrence probabilities of node symbols positive. Next, according to $P'_t(X_i = x|C_i = c)$, node symbols are decided in sequence from root to terminals.

2.5 Parameter Control

Table 1 indicates the parameters used for experiments.

3 Performance Difference Due to the Hybrid Ratio

3.1 Function Regression Problem

Consider a function regression problem. $prog_i$ is a function expressed by a program tree and f_{obj} is the function to be approximated. The fitness value is given with the following formula:

$$fitness = 1000 - 50 \sum_{j=1}^{30} |prog(X_j) - f_{obj}(X_j)|, \tag{10}$$

where

$$X_j = 0.2(j - 1). \tag{11}$$

Table 1. Parameters for a function regression problem.

Common parameters for EDP and GP	
M: population size	1000
E_S: elite size	5
F: function node set	$\{+, -, *, /, \cos, \sin\}$
T: terminal node set	$\{x, 0.05, 0.10, 0.15, \cdots, 1.00\}$
N_F: the number of function nodes	6
N_T: the number of terminal nodes	21
P_F: generation probability of function node	0.8
P_T: generation probability of terminal node	0.2
Tree size limitation in initializing population	max depth = 6
EDP parameters	
α: Laplace correction rate	0.3
$P_{bias}(X_i = x \mid C_i = c)$: the probability to bias distribution	$\frac{1}{N_F+N_T}$
η: learning rate	0.2
S_S: sampling size	200
T_{edp}: tournament size for sampling	20
Tree size limitation	max depth = 6
GP parameters	
P_M: mutation probability	0.1
T_{gp}: tournament size for GP operator	5
Tree size limitation	max depth = 6

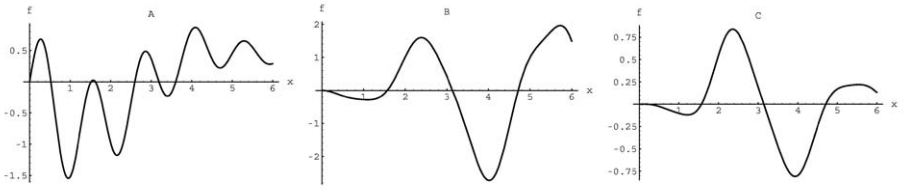

Fig. 3. Objective functions.

Objective functions are

$$\mathrm{A} : f_{obj}(x) = (2 - 0.3x)\sin(2x)\cos(3x) + 0.01x^2 \tag{12}$$

$$\mathrm{B} : f_{obj}(x) = x\cos(x)\sin(x)(\sin^2(x)\cos(x) - 1) \tag{13}$$

$$\mathrm{C} : f_{obj}(x) = x^3\cos(x)\sin(x)e^{-x}(\sin^2(x)\cos(x) - 1) \tag{14}$$

which are plotted in Fig. 3. Objective function C is cited from [Salustowicz97]. Although B is obtained from simplification of C, B is more difficult to search (see fitness values in Fig. 5 and 6). A is our original function and the most difficult of the three objective functions.

Fig. 4, 5, and 6 show the mean of max fitness values for 100 runs, that is,

$$\bar{f}_{max}m = \frac{1}{100}\sum_{k=1}^{100} f_{max}k,m \tag{15}$$

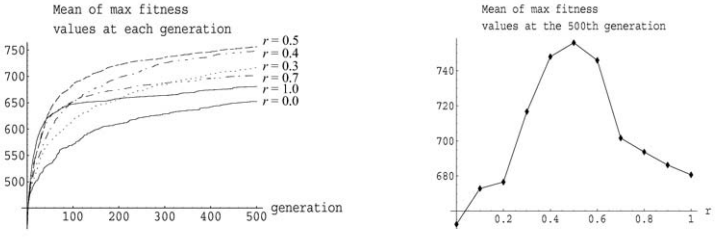

Fig. 4. Results for objective function A.

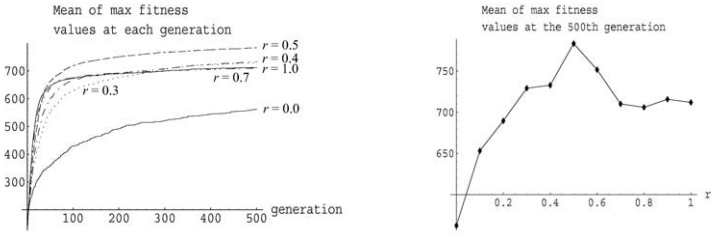

Fig. 5. Results for objective function B.

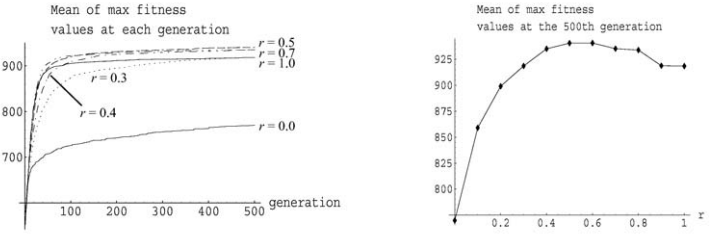

Fig. 6. Results for objective function C.

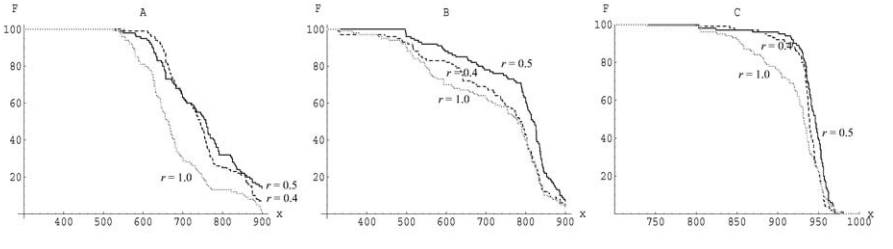

Fig. 7. $F(x)$: frequency of max fitness at the 500 th generation greater than x, with objective functions A and B.

where $f_{max\,k,m}$ is the maximum fitness value in a population of the m^{th} generation at the k^{th} run. Note that $\bar{f}_{max\,m}$ is not a mean fitness value of a population, but a mean value of the maximum fitness value $f_{max\,k,m}$. The solution in an evolutionary computing is given by an individual who has the maximum fitness value in a population. Therefore, system performances should be compared in maximum fitness values.

Fig. 7 shows the frequency of runs in which the maximum fitness value at the 500^{th} generation is over x, that is,

$$F(x) = \sum_{k=1}^{100} \delta(x \leq f_{max\,k,500}) \tag{16}$$

where

$$\delta(x \leq a) = \begin{cases} 1 & : x \leq a \\ 0 & : x > a \end{cases}. \tag{17}$$

Fig. 4, 5, 6, and 7 indicate the similar tendency in each case. Although the $r = 1.0$ system which is pure GP, demonstrated the best performance in younger generations, gradually hybrid systems overtook pure GP one after another. The "overtaking" was conspicuous when $r = 0.3$ or $r = 0.4$. At the 500^{th} generation, the performance of the $r = 0.5$ system was the best in all cases. The system performances at the 500^{th} generation reached a peak at $r = 0.5$, and got worse as the hybrid ratio was biased.

Mean cannot give adequate information for system performances, hence we showed Fig. 7. Fig. 7 demonstrates that the hybrid system is also superior to pure GP in the success rate of a search. For instance, in the case of A, the probabilities that the maximum fitness value at the 500^{th} generation is over 700 are $\frac{63}{100}$ with $r = 0.5$ and $\frac{30}{100}$ with pure GP respectively.

3.2 Analysis of the Results

The system performances are estimated by $\bar{f}_{max\,m}$. However, in order to conclude that the differences of these values are statistically significant and reliable, not only mean but also standard deviation and sample size (= 100) should be taken into consideration. We used Welch's test for the obtained experimental results. By means of Welch's test, it can be judged whether 2 data sets are samples from the same statistical population or not. As a result of Welch's test with 10% significance level, the differences between the $r = 0.5$ system and pure GP at the 500^{th} generation were significant in all cases. Statistically speaking, the null hypothesis that data in the $r = 0.5$ system and in pure GP were sampled from the same statistical population was rejected (the probability that the null hypothesis is correct is less than 10%). In the case of objective function C, although the difference in values was slight, standard deviation was negligible (see Fig. 7); Welch's test concluded that the differences were significant.

In the $r = 0.5$ hybrid system, the updating times of the maximum fitness values at each generation of the EDP operator and the GP operator are counted respectively. Surprisingly, the EDP operator hardly contributes to construction of the best individual directly, and only the GP operator does.

The summary of results is as follows:

1. The success probability of the hybrid system in a search is higher.
2. Statistical testing proved that the $r = 0.5$ system was superior to pure GP ($r = 1.0$) at the 500^{th} generation.
3. In any case, the same tendencies, the "overtaking", pure GP's superiority in younger generations and so on, were found.
4. Pure EDP was worse.
5. The obtained graphs were consistent and well formed.
6. The EDP operator could not produce better individuals, but played some invisible roles.

We consider these features of the hybrid system to be universal. In other words, the parameter r characterizes the system behavior and the performance. Besides, hybridization helps GP and EDP compensate for their defects, and build a better evolutionary system.

Here are some follow-up questions:

1. Why is the hybrid system superior to pure GP? What are EDP's roles?
2. Is the low performance in younger generations important?
3. How should r be controlled? What method is the best?

The next section will answer some of these.

4 Discussion

4.1 Change of Hybrid Ratio at Each Generation

This section investigates the hybrid system's performance, changing the hybrid ratio r at each generation. In Fig. 4, until the 50^{th} generation, the higher the GP ratio of the system is, the better its performance is. Therefore, the system that has a high GP ratio in younger generations and decreases the ratio later is expected to have higher performance.

Comparative experiments were carried out in 8 systems, shown in Fig. 8. Objective function is A given in the formula (12). In system D, the GP ratio is linearly increased from 0, at the 0^{th} generation, to 1.0, at the 500^{th} generation. On the other hand, the system E decreases the ratio linearly. System G switches the ratio from 1.0 to 0.3 at the 205^{th} generation because the $r = 0.3$ system overtook pure GP at the 205^{th} generation, as shown in Fig. 4. System H was prepared in the same manner as G. Therefore, H and G are the top favorites in these systems.

Fig. 9 and 10 show the results of comparative experiments. Surprisingly, system A overtook G (B also overtook H). As a result of Welch's test with

System	r
A: classical hybrid	$r = 0.3$
B: classical hybrid	$r = 0.5$
C: pure GP	$r = 1.0$
D: linear increasing	$r = \frac{i}{500}$
E: linear decreasing	$r = 1 - \frac{i}{500}$
F: random	r is a random value from 0 to 1 and different for each generation.
G: switching	$r = \begin{cases} 1.0 & : i < 205 \\ 0.3 & : i \geq 205 \end{cases}$
H: switching	$r = \begin{cases} 1.0 & : i < 40 \\ 0.5 & : i \geq 40 \end{cases}$

Fig. 8. Systems with changing r, where i is the generation number.

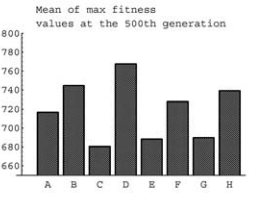

Fig. 9. Mean of max fitness values at the 500^{th} generation.

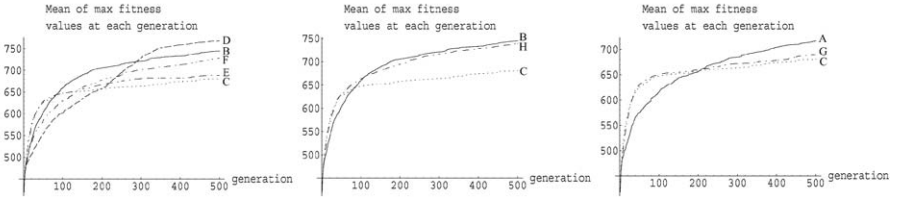

Fig. 10. Mean of max fitness values at each generation.

10% significance level, the differences were significant. This result means that population states of A and G are far different in spite of close performance at the 205^{th} generation. In other words, EDP's behavior before the 205^{th} generation likely has a good influence later.

Another interesting result is that system D was superior to all other systems, especially E. As a result of Welch's test with 10% significance level, the differences were significant. Although it was expected that D would be worse than E, judging from Fig. 4, the result was quite the opposite. This point is evidence that EDP functions well in early generations.

How does the hybrid system transmit EDP's work in an early stage of evolution to posterity?

1. The probability distribution (Bayesian network) learned incrementally memorizes the past population state.
2. With EDP, the diversity of the population is maintained at each generation and useful part structures can survive.

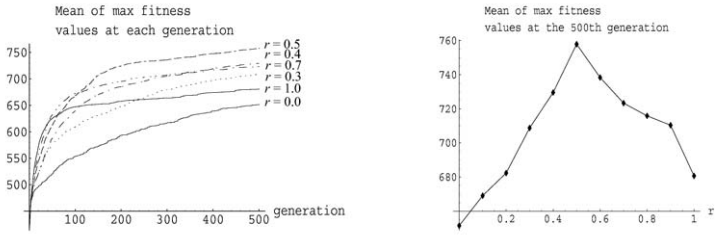

Fig. 11. System of $\eta = 0$

3. There is diversity inside individuals. Individuals constructed by EDP have more multifarious part structures. These various structures are put together in later generations of evolution.

The next section considers these possibilities.

4.2 System of $\eta = 0$

In order to test the hypothesis that the probability distribution memorizes the past EDP's work, the system of $\eta = 0$ was simulated. This system estimates distribution without referring to the past distribution (see Section 2.3). Objective function A was used.

As indicated in Fig. 11, the characteristic of the hybrid system was kept. The "overtaking" still took place and the $r = 0.5$ system was the best. Therefore, the past information accumulated in the probability distribution does not cause the high performance of the hybrid system.

The result shown in Fig. 9 suggests the third possibility mentioned in Section 4.1. This is because system D, which has the best performance of all, cannot benefit from EDP in later generations. However, in order to obtain more reliable evidence, we are currently working on testing the second possibility.

4.3 Related Work

Probabilistic Incremental Program Evolution (PIPE) [Salustowicz97] was used to perform a program search based on a probabilistic model. However, PIPE assumes the independence of program nodes and differs from our approach using a Bayesian network in this assumption. The merits of having probabilistic dependency relationship are as follows:

1. Because an occurrence probability of a node symbol is dependent on its parent node, estimation and generation are serial from a parent node to a child. Therefore, it can derive and generate building blocks.
2. The past dominant structure can survive after switching the probability distribution based on a parent node symbol.

On the other hand, optimization using a Bayesian network is much researched. [Larranaga et al.00a] [Larranaga et al.00b]. However, their application is limited to fixed length array search problems.

5 Conclusion

In this paper, we proposed a hybrid system of EDP and GP, and demonstrated that the hybrid system was superior to both pure GP and pure EDP. The experimental results indicated that EDP worked effectively in early generations and contributed to later high performance. It turned out that pure GP could not generate enough various part trees in early generations to build excellent individuals. On the other hand, EDP cannot shift useful part trees to another position in the tree. Hybridization helps EDP and GP compensate for each other.

However, it is not clear how EDP works in the hybrid system. In future work, the detail of EDP's function in early generations will be researched. We are also interested in the greatest control of the hybrid ratio r and the robustness of the behavior that the hybrid system exposed in our experiments.

References

[Salustowicz97] Rafal Salustowicz and Jurgen Schmidhuber (1997) "Probabilistic Incremental Program Evolution," Evolutionary Computation 5(2):123-141.

[Baluja94] Baluja S. (1994) "Population Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning," Technical Report No. CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, Pennsylvania.

[Larranage and Lozano02] Pedro Larranage and Jose A. Lozano (2002) "Estimation of Distribution Algorithms," Kluwer Academic Publishers

[Larranaga et al.00a] Larranaga, P., Etxeberria, R. Lozano, J. A. and Pena, J.M. (2000) "Combinatorial Optimization by Learning and Simulation of Bayesian Networks," Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, Stanford, pp343-352.

[Larranaga et al.00b] Larranaga, P., Etxeberria, R. Lozano, J. A. and Pena, J.M. (2000) "Optimization in continuous domains by Learning and simulation of Gaussian networks," Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program, pp201-204.

[Cestnik90] Cestnik, B. (1990) "Estimating probabilities: A Crucial Task in Machine Learning," Proceedings of the European Conference in Artificial Intelligence, pp.147-149.

[Langdon02] William B. Langdon, Riccardo Poli (2002) "Foundations of Genetic Programming," Springer-Verlag Berlin Heidelberg, pp175-176.

[Yanai03a] Kohsuke Yanai and Hitoshi Iba (2003) "Estimation of Distribution Programming based on Bayesian Network," In Proc. of Congress on Evolutionary Computation (CEC) 2003, pp1618-1625.

[Yanai03b] Kohsuke Yanai and Hitoshi Iba (2003) "Program Evolution using Bayesian Network," In Proc. of The First Asian-Pacific Workshop on Genetic Programming (ASPGP03), pp16-23.