Probabilistic Distribution Models for EDA-based GP

Kohsuke Yanai
Dept. of Frontier Informatics,
Graduate School of Frontier Science,
The University of Tokyo.
5-1-5 Kashiwanoha, Kashiwa-shi,
Chiba 277-8561, Japan
yanai@iba.k.u-tokyo.ac.jp

Hitoshi Iba
Dept. of Frontier Informatics,
Graduate School of Frontier Science,
The University of Tokyo.
5-1-5 Kashiwanoha, Kashiwa-shi,
Chiba 277-8561, Japan
iba@iba.k.u-tokyo.ac.jp

ABSTRACT

This paper proposes a novel technique for a program evolution based on probabilistic models. In the proposed method, two probabilistic distribution models with probabilistic dependencies between variables are used together. We empirically comfirm that our proposed method has higher search performance. Thereafter, we discuss the effectiveness of its distribution models.

Categories and Subject Descriptors: G.1.6 [Numerical Analysis]: Optimization; G.3 [Probability and Statistics]: Probabilistic Algorithms; I.2.6 [Artificial Intelligence]: Learning

General Terms: Algorithms

Keywords: Genetic Programming, Program Evolution, Evolutionary Computing, Estimation of Distribution Algorithm, Probabilistic Model-Building Genetic Algorithm

1. INTRODUCTION

In this paper, we propose **Extended Estimation of Distribution Programming** (XEDP), which can be viewed as an extension of EDP [4]. A program population is evolved by the repetition of the estimation of distribution and the program generation without any crossover and mutation, similar to other EDA-based GPs. This paper applies XEDP to standard problems of GP and conducts comparative experiments. According to experimental results, we discuss the effectiveness of XEDP's distribution models.

2. METHOD

The steps of the XEDP algorithm are as follows:

Step 1: Generate an initial population.
Step 2: Estimate individuals and assign fitness value to each.
Step 3: Terminate if termination criterion is satisfied.
Step 4: Learn the probabilistic distribution.
Step 5: Copy elite individuals to a new population.
Step 6: Generate individuals using the acquired distribution.
Step 7: Replace the old population with a new one.

The proposed technique uses the following two probabilistic distribution models together: **Conditional Probability Tree** and **Recursive Distribution**. The conditional probability tree estimates global structure of a program, while the recursive distribution estimates useful substructures.

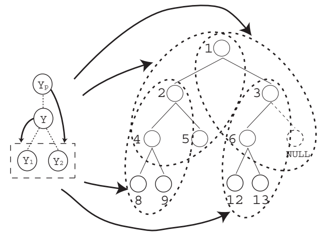

Figure 1: Estimation of the recursive distribution.

As the conditional probability tree, Bayesian Network of a tree structure is used. Each probabilistic variable on the Bayesian Network corresponds with a node in the same position in a program tree. The probability of a symbol depends only on the symbol of its parent node. Accordingly, each probabilistic variable of the conditional probability tree has only one dependent variable, which represents the symbol of the parent node. This is based on the assumption that a probabilistic dependency is strong between the parent node and its child nodes.

For a recursive distribution, the following conditional probability is used as a distribution model.

$$\mathrm{P}(Y_1, Y_2, \cdots, Y_{a_{max}} | Y, Y_p) \tag{1}$$

$Y_1, Y_2, \cdots, Y_{a_{max}}$ are probabilistic variables of child nodes of Y, and Y_p is probabilistic variables of the parent node of Y. Thus, the recursive distribution represents the conditional joint probability of child nodes given the node symbols of its immediate parent and grand parent. For instance, there are four substructures that match the shape of the recursive distribution in the program tree, as shown in Figure 1.

Individuals are generated as follows:

Step 1: Generate a program T using the conditional probability tree.
Step 2: Generate subtree S recursively using the recursive distribution.
Step 3: Replace the arbitrary subtree of T with S.

3. EXPERIMENTAL STUDY

As for the well-known benchmark problems such as the Max problem, the Multiplexer problem, and the Wall-following

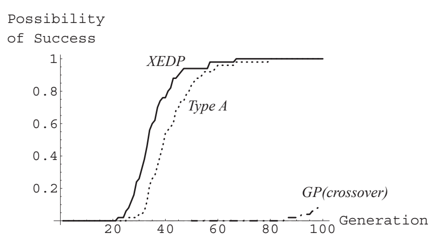

Figure 2: Cumulative probability of success for the Max problem.

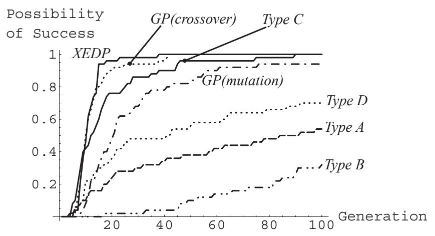

Figure 3: Cumulative probability of success for the Boolean 6-bit multiplexer problem.

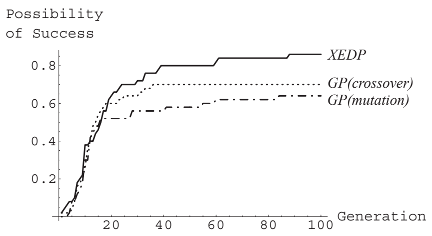

Figure 4: Cumulative probability of success for the Wall-following problem.

problem, comparative experiments of XEDP and GP were carried out.

In order to investigate the features of XEDP, a comparative experiment was also conducted for methods that partly use XEDP operators: **Type A** generates a program using only the conditional probability tree; **Type B** generates a program using only the recursive distribution; **Type C** generates a program T using the conditional probability tree and replaces the arbitrary subtree of T with a randomly generated tree; **Type D** uses a tree-like model in which probabilistic variables are independent ,instead of the conditional probability tree.

Figure 2, 3, and 4 show the cumulative probability of success for each generation. Thease graphs indicate that XEDP is better than GP and other methods in any problem. As seen from Figure 2, XEDP obtained the optimum solution until the 66^{th} generation in every run, while GP with crossover shows a low success probability of 8% even at the 100th generation. GP with mutation has never produced the optimum solution till the 100th generation. Search with Type A ($P_{rep} = 0$) was also successful. This means that program generation using the conditional probability tree is effective regarding the Max problem. [3] reports that GMPE required 13590 evaluations to achieve the success rate 60%. On the other hand, XEDP needed only 13200 evaluations ($= 200 \times 66$) for 100% success rate, which means XEDP has higher search performance than GMPE as for the Max problem.

In the wall-following problem, the cumulative probabilities of success with XEDP got better during the latter half of evolution. Then probability of success of XEDP at the 50th generation is higher than that of GP. On the other hand, GP with crossover shows no fluctuation after the 40th generation.

4. DISCUSSION

As for the Max problem, the Multiplexer problem, and the Wall-following problem, XEDP marked as high search performance as or higher than GP. PIPE [1] is available only for the search problem of real value function. Besides, eCGP [2] and GMPE do not show significant search performance regarding the problems for which GP search is effective. On the other hand, XEDP marked higher performance than GMPE as with the Max problem. XEDP is considered to be a higher general-purpose technique than PIPE, eCGP, and GMPE. XEDP turned out to be effective for evolution of robot program too. In general, XEDP seems highly effective as a technique for program search.

Search by Type B was the worst on the Boolean 6-bit multiplexer problem. Therefore, estimating global structure of a program by the conditional probability tree turns out to be essential. Besides, the search performance of Type C is worse than that of XEDP on the Boolean 6-bit multiplexer problem. This means that generation by the recursive distribution is not random generation. Thus, it is considered that the recursive distribution could extract useful substructures for a good program from a population. From what has been discussed above, we can conclude that combination of the conditional probability tree and the recursive distribution is effective for program evolution.

Search by Type A is successful regarding the Max problem, which implies that generation using the conditional probability tree has a function to widen the redion where search is possible. It can be presumed that this function led to the higher performance of XEDP than that of GP regarding the Boolean 6-bit multiplexer problem and the Wall-following problem.

Type D can estimate both global structure and substructures of a program. However, Type D can not deal with probabilistic dependencies between nodes in a program tree when estimating global structure, while XEDP can involve probabilistic dependencies by the **conditional** probability tree. Search by Type D also failed and therefore the probabilistic dependencies between variables in the conditional probability tree is thought to play an important role in program search.

5. REFERENCES

[1] R. P. Salustowicz and J. Schmidhuber. Probabilistic incremental program evolution: Stochastic search through program space. In M. van Someren and G. Widmer, editors, *Machine Learning: ECML-97*, volume 1224, pages 213–220. Springer-Verlag, 1997.

[2] K. Sastry and D. E. Goldberg. Probabilistic model building and competent genetic programming. In *Genetic Programming Theory and Practise*, pages 205–220. Kluwer, 2003.

[3] Y. Shan, R. McKay, R. Baxer, H. Abbass, D. Essam, and H. Nguyen. Grammar model-based program evolution. In *Proceedings of the Congress on Evolutionary Computation: CEC-2004*, pages 478–485, 2004.

[4] K. Yanai and H. Iba. Estimation of distribution programming based on Bayesian network. In *Proceedings of Congress on Evolutionary Computation: CEC-2003*, pages 1618–1625, 2003.