Improving Evolvability of Genetic Parallel Programming Using Dynamic Sample Weighting

Sin Man Cheang, Kin Hong Lee, and Kwong Sak Leung

Department of Computer Science and Engineering
The Chinese University of Hong Kong
{smcheang,khlee,ksleung}@cse.cuhk.edu.hk

Abstract. This paper investigates the sample weighting effect on Genetic Parallel Programming (GPP) that evolves parallel programs to solve the training samples captured directly from a real-world system. The distribution of these samples can be extremely biased. Standard GPP assigns equal weights to all samples. It slows down evolution because crowded regions of samples dominate the fitness evaluation and cause premature convergence. This paper compares the performance of four sample weighting (SW) methods, namely, Equal SW (ESW), Class-equal SW (CSW), Static SW (SSW) and Dynamic SW (DSW) on five training sets. Experimental results show that DSW is superior in performance on tested problems.

1 Introduction

Genetic Programming (GP) has been used to learn computer programs for data classification. Linear GP (LGP) uses a linear representation which is directly corresponding to the program statements based on a register machine. In this paper, experiments are performed on a novel LGP paradigm, Genetic Parallel Programming (GPP) [1], which evolves parallel programs based on a multiple ALU register machine, Multi-ALU Processor (MAP). The MAP employs a tightly coupled, shared-register, MIMD architecture. It can perform multiple operations in parallel in each processor clock cycle. GPP has been used to evolve parallel programs for symbolic regression, recursive function regression and artificial ant problems. Besides of evolving parallel programs to make use of the parallelism of the specific parallel architecture, the GPP accelerating phenomenon [3] reveals the efficiency of evolving algorithms in parallel form.

Sample Weighting (SW) is a technique to change the samples' weights in order to increase the efficiency of fitness-based evolutionary algorithms. The basic principle of SW is to balance the biased distribution of sample points in a solution space. Researchers have proposed different SW methods which determine samples' weights in runtime. For example, we have applied Dynamic SW (DSW) to Genetic Algorithms (GA) to evolve Finite State Machines [2]. In this paper, we show that DSW can also be applied to GP to increase evolutionary efficiency of Boolean function regression and data classification problems. We compare DSW to other three SW methods,

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 1802–1803, 2003.

Equal SW (ESW), Class-equal SW (CSW) and Static SW (SSW). In ESW, all samples' weights are set to equal without considering the distribution of samples. In CSW, the calculation of samples' weights is based on the number of classes. In DSW, samples' weights are recalculated periodically based on the contributions of individual samples to the population. SSW is a special case of DSW that does not modify samples' weights after the initialization.

2 Experiments, Results, and Conclusions

We have applied four SW methods on five training sets. Three training sets are Boolean functions, 3-even-parity (*3ep*), 5-even-parity (*5ep*) and 5-symmetery (*5sm*). The remainings are UCI data classification databases, Wisconsin Breast Cancer (*bcw*) and Bupa Liver Disorder (*bld*). Three Boolean function sets (F_1={and,or,xor,nop}, F_2={and,or,nand,nor,nop} and F_3={and,or,not,nop}) are used for Boolean functions. One arithmetic function set (F_4={+,–,×,%,move,nop}) is used for UCI databases. Table 1 below shows the relative computation effort of ESW versus DSW. Obviously, DSW boosts the evolutionary efficiency of all tested problems (all values are greater than 1.0). In the *5sm*_F_3 experiment, DSW reduces the computational effort by 744.5 times. This shows that DSW is superior in evolutionary speedup on this problem. Even if the improvements are not significant on the *bcw* and *bld* problems, DSW is a helpful SW method to speed up evolution because it is easily to be implemented on the original population-based evolutionary algorithms.

Research in the near future will be focused on extending the usage of DSW to different applications areas, e.g. multi-class classification and numeric function regression problems. The main concern is to apply DSW to non-count based algorithm, e.g. floating-point function regression problems in which the fitness of an individual program is based on the total error of the expected outputs and the program outputs.

Table 1. Relative computation effort (E_{ESW}/E_{DSW})

	*3ep*_F_3	*5ep*_F_1	*5ep*_F_2	*5sm*_F_1	*5sm*_F_2	*5sm*_F_3	*bcw*	*bld*
E_{ESW}/E_{DSW}	11.8	3.7	1.8	51.4	226.7	744.5	1.2	1.1

References

1. Leung, K.S., Lee, K.H., Cheang, S.M.: Evolving Parallel Machine Programs for a Multi-ALU Processor. Proc. of IEEE Congress on Evolutionary Computation (2002) 1703-1708
2. Leung, K.S., Lee, K.H., Cheang, S.M.: Balancing Samples' Contributions on GA Learning. Proc. of the 4th Int. Conf. on Evolvable Systems: From Biology to Hardware (ICES), Lecture Notes in Computer Science, Springs-Verlag (2001) 256–266
3. Leung, K.S., Lee, K.H., Cheang, S.M.: Parallel Programs are More Evolvable than Sequential Programs. Proc. of the 6th Euro. Conf. on Genetic Programming (EuroGP), Lecture Notes in Computer Science, Springs-Verlag (2003)