GAP/D: VLSI Hardware for Parallel and Adaptive Distributed Genetic Algorithms

Kazutaka Kobayashi
InterDesign Technologies Inc., Japan
kobayashi.kazutaka@interdesigntech.co.jp

Norihiko Yoshida
Department of Information and Computer Sciences
Saitama University, Japan
yoshida@mail.saitama-u.ac.jp

Shuji Narazaki
Department of Computer and Information Sciences
Nagasaki University, Japan
narazaki@cs.cis.nagasaki-u.ac.jp

Abstract

This paper presents GAP/D, a VLSI implementation of a dynamic adaptation scheme for the frequency of inter-deme migration in distributed genetic algorithms (GA). Distributed GA, or multi-deme-based GA, uses multiple populations which evolve concurrently. The purpose of dynamic adaptation is to improve convergence performance so as to obtain better solutions. Through simulation experiments, we proved that our scheme achieves better performance than fixed frequency migration schemes.

1. Introduction

VLSI hardware implementations of genetic algorithms (GAs) have already been widely studied. Their extensions towards parallel GA and distributed GA are also being studied to obtain much better performance improvement.

Parallel GA evaluates fitness values of multiple genotypes simultaneously, and thus realizes fine-grained parallel processing. Distributed GA, or multi-deme-based GA, uses multiple populations which are evolving concurrently, and thus realizes coarse-grained parallel processing.

We designed and developed GAP, a general-purpose GA-VLSI with parallel and distributed GA implementations [1].

The distributed GA configuration of GAP is composed of multiple GAPs working concurrently. It is important in distributed GA that some of the genotypes should be "migrated" between demes occasionally in order to prevent isolated evolution and premature convergence. We used the simplest scheme for migration: in every evolution cycle, newly created genotypes are migrated to the next deme.

In this paper, we present a dynamic adaptation scheme for the frequency of inter-deme migration. In short, each processor observes the convergence status of its deme, i. e. observes the gradient of the convergence curve, and determines how often to communicate for migration. The purpose is to reflect convergence properties and adapt dynamically so as to obtain better solutions.

We present GAP/D, a multi-deme-based distributed-GA VLSI design, and through simulation experiments, we prove that our scheme achieves better performance than fixed frequency communication schemes without affecting the overall convergence properties.

Section 2 summarizes the design of the original GAP. Section 3 describes its extensions to parallel GA and distributed GA. Section 4 presents dynamic adaptation scheme for inter-meme migration, and Section 5 shows an experiment results. Section 6 contains some concluding remarks.

2. Design of GAP

Our GA-VLSI system is composed of two modules: a general-purpose problem-independent part for selection, crossover and mutation operations, and a problem-dependent part for fitness evaluation. GAP is the problem-independent module, working together with FEP (Fitness Evaluation Processor) implementing problem-dependent fitness evaluation module (Figure 1).

GAP contains some modules for random number generation and population control along with modules for selection, crossover and mutation. FEP must be designed for a given specific problem. GAP and FEP are connected via a population memory. In this system, the below modules are the most innovative. Their details are found elsewhere [1].

978-0-7695-3605-7/09 $25.00 © 2009 IEEE
DOI 10.1109/CSO.2009.454

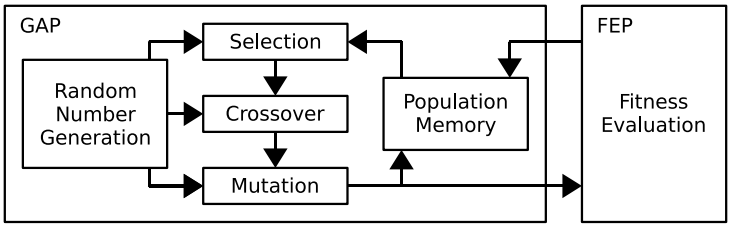

Figure 1. Basic GAP architecture.

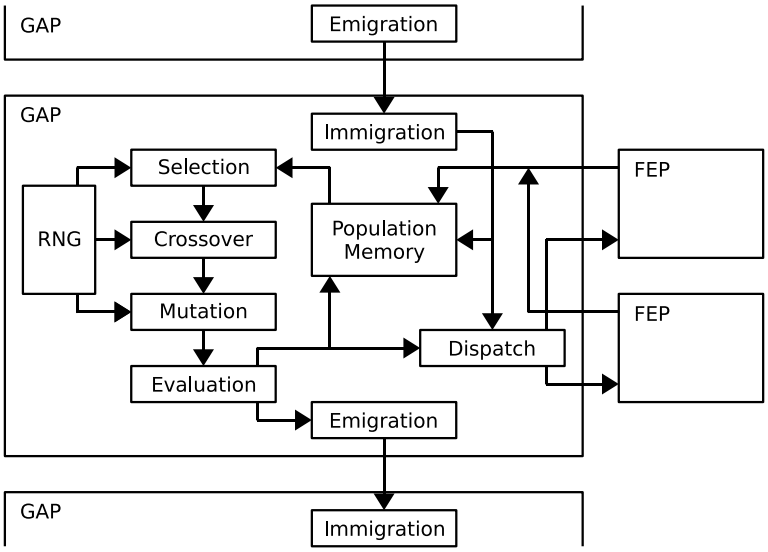

Figure 2. GAP/D architecture.

Population Memory GAP employs the steady-state GA. Genotypes are passed from GAP to FEP as soon as they are created, and then passed back from FEP to GAP as soon as they are evaluated. Genetic operations and fitness evaluations are overlapped in this way, and the whole system works in a pipeline fashion. Only a single set of population memories is required.

Selection Module We have introduced a selection scheme which is suitable for VLSI hardware. It is named the "simplified tournament selection", since it is a simplified version of the tournament selection scheme.

Random Number Generation Module This module generates a sequence of pseudo-random bit strings using the theory of linear cellular automata (CA). The CA scheme was proved theoretically to generate better random sequences, in the sense that the sequences had a longer cycle length, than the scheme of linear feedback shift registers (LFSR) which has been widely used.

3. Extensions to Parallel and Distributed GA

As problems to be solved become more complicated, FEPs turn into bottlenecks for the overall GA system performance. Therefore, multiplication of FEPs in a system can be effective for improving performance.

In the research area of theories and software for GA, there have already been many studies on parallel and distributed processing for GA. Parallel GA evaluates fitness values of multiple genotypes simultaneously, and thus realizes fine-grained parallel processing. Distributed GA, or multi-deme-based GA, uses multiple populations which are evolving concurrently, and thus realizes coarse-grained parallel processing.

The basic architecture of GAP hardware design facilitates extensions for parallel GA and distributed GA.

Parallel GA for GAP The simplified tournament selection scheme creates two new genotypes and evaluates their fitness values in every evolution cycle. Therefore, two FEP chips can be connected to a GAP, as shown in Fig-

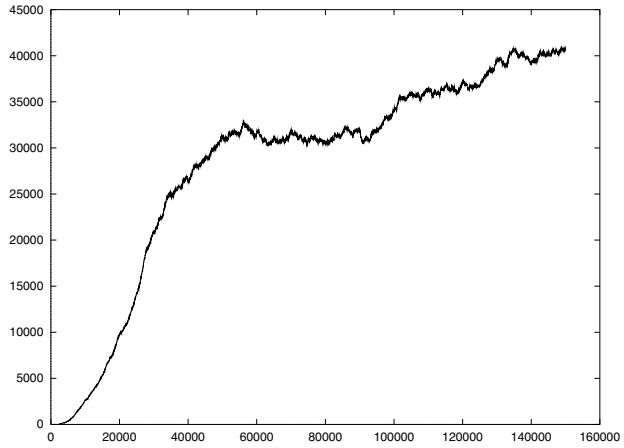

Figure 3. Convergence on GAP.

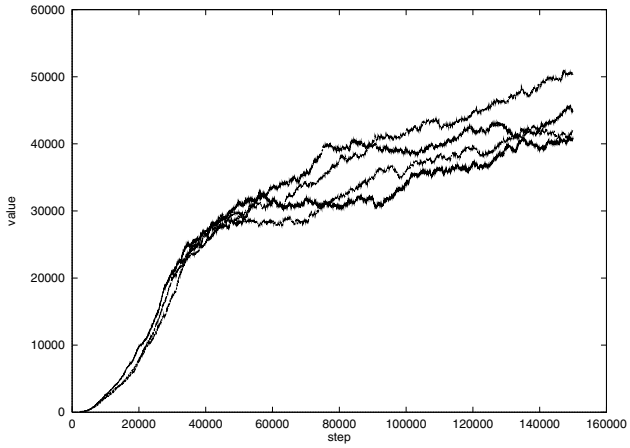

Figure 4. Convergence on GAP/D.

ure 4, and they evaluate the fitness values of the two new genotypes simultaneously. The dispatch module controls these two FEPs. This parallel GA configuration is expected to double the performance of fitness evaluation.

Distributed GA for GAP The distributed GA configuration of GAP is composed of multiple GAPs working concurrently. It is important in distributed GA that some of the genotypes should be "migrated" between demes occasionally in order to prevent isolated evolution and premature convergence. At first, we used the simplest scheme for migration: in every evolution cycle, newly created genotypes are migrated to the next deme. GAP chips are connected to each other in a ring form. Newly created genotypes are transferred to the population memory in the next GAP chip via the emigration and immigration modules. This configuration contributes to convergence.

4. Dynamic Adaptation for Migration Frequency

Regarding migration frequencies, most distributed GA systems migrate genes in every generation or at randomly chosen intervals. These migration schemes are so simple that they cannot reflect convergence properties, nor adapt dynamically so as to obtain good solutions. A too low frequency leads to premature convergence, whereas a too high frequency spoils the effect of parallel evolution, and system performance due to communication overhead.

The essence of our idea is very simple in its principle. A processor observes the gradient of the convergence curve of its deme, and performs migration when the gradient gets flat, i. e. its deme is about to converge. On each generation t, the processor computes an average fitness value $f(t)$ and its gradient

$$g(t) = \frac{f(t) - f(t - \Delta t)}{\Delta t}$$

It then checks whether $g(t) \leq g_{th}$ or not against a certain pre-defined threshold g_{th}, and performs migration when this condition stands (Figure 2).

There is a related study for dynamic adaptation based on population distribution [2]. A processor observes standard deviation σ of gene fitness in its demes, and performs migration when σ gets low. However, this scheme disregards absolute values of gene fitness, thus sometimes causes premature convergence towards the low level of fitness. Also, this scheme involves intensive computation, and difficult to fit for hardware implementation.

5. Simulation Platform and Experiments

We implemented the VLSI hardware design of GAP using a hardware description language "SFL" [3], assuming the CMOS 0.8m process technology. We carried out logic simulation and logic synthesis for preliminary evaluation of the design prior to actual VLSI fabrication. The specification of the prototype implementations is:

Population size:	256
Genotype bit length:	64
Fitness bit length:	24
Crossover probability:	1
Mutation probability:	1/32

The comparisons regarding the number of gates (circuit size) and the number of steps per cycle (speed) between the original GAP and the new GAP/D are as below:

	#gates	#clocks/cycle
GAP	53890	17
GAP/D	60477	21

We did some experiments using a well-known standard example for GA experiments called "Royal-road function" [4], which is simple but slow to converge. Figure 3 shows the convergence of Royal-road function on the original GAP, and Figure 4 shows the convergence on the new GAP/D, where the threshold is 32, and four curves correspond to four demes respectively. We observed that the former converges to the value of approximately 41,000, while the latter converges to 50,000 at highest. This proves that the adaptive migration scheme improves the quality of the solution result.

6. Concluding Remarks

We presented a dynamic adaptation scheme for the frequency of inter-deme migration in multi-deme-based distributed GA on GA VLSI. Each processor observes the gradient of the convergence curve, and determines how often to migrate.

Using an example of function minimization problems, we showed that our scheme achieves better performance than fixed frequency migration schemes without affecting the overall convergence properties.

This improvement causes some increases in the circuit size and the execution speed. The increase in the circuit size is not a serious issue. The increase in the execution speed from 17 clocks to 21 clocks per cycle (approximately 124%) can be acceptable, however, we had better improve the technique so as to reduce the increase. It is left for further study.

References

[1] N. Yoshida, T. Yasuoka, T. Moriki, and T. Shimokawa, "VLSI Hardware Design for Genetic Algorithms and Its Parallel and Distributed Extensions", *Int. J. of Knowledge-Based Intelligent Engineering Systems*, Vol. 5, No. 1, 2001, pp. 14–21.

[2] M. Munetomo, Y. Takai, and Y. Sato, "An Efficient Migration Scheme for Subpopulation-based Asynchronously Parallel Genetic Algorithms", *Proc. 5th Int. Conf. on GA*, 1993, p. 649.

[3] Y. Nakamura, K. Oguri, et al., "High-Level Synthesis Design at NTT Systems Labs", *IEICE Trans. on Inf. & Syst.*, Vol. E76-D, No.9, 1993, pp. 1047–1054.

[4] M. Mitchell, and S. Forrest, "Fitness Landscapes: Royal Road Functions", *Handbook of Evolutionary Computation* (T. Back, D. Fogel, and Z. Michalewicz, eds.), Oxford, 1997.