Chinese Journal of Electronics, 2003, 12(2): 254-258

# Simulation Optimization with Multiple-demes Genetic Algorithms in Master-Slave Parallel Mode

Xiaofeng Xie, Wenjun Zhang, Jun Ruan, Zhilian Yang
Institute of Microelectronics, Tsinghua University, Beijing 100084, P. R. China
*Email: xiexiaofeng@ tsinghua.org.cn*

**Abstract** – Simulation optimization for complex space is realized by applying genetic algorithms (GAs) to searching the optimum parameters that satisfying the desired characteristics. It is necessarily to use parallel calculation to obtain satisfying results in a reasonable amount of time. The advantages and drawbacks of existing parallel calculation mode are analyzed. Then the multiple-demes GAs in master-slave mode (MDMS) is proposed to make progress in not only the quality of solutions but also the speedup for multiple-processors, when comparing to conventional master-slave mode, while shows less limitations on the number of processors than that of coarse-grained and neighborhood mode. The test example in semiconductor device synthesis shows the efficiency.

**Key words**: simulation optimization, parallel calculation, master-slave, multiple-demes genetic algorithms

## 1. Introduction

Integrating optimization routines into simulation packages has become almost a necessity, in order to seek improved settings of user-selected system parameters with respect to the performance measure(s) of interest [1].

In simulation optimization, it is often consider that simulation model as a "black box", and the optimization procedure uses the outputs from the simulation model which evaluate the outcomes of the inputs that fed into the model, and decides upon a new set of input values based on the current and the past evaluations.

Recently, many studies using modern heuristic techniques for simulation optimization have been encountered [3-6]. The comparison [7, 8] shows genetic algorithms (GAs) are converged faster than tabu search (TS) and simulated annealing (SA), which makes it more suitable for simulation optimization, because in most cases, the major drawback of simulation for practical applications is time-consuming. However, to solve some complex problems, the computational requirements can be extremely large because GAs may still require hundreds or thousands of evaluations.

Fortunately, GAs is known as inherently parallelism [9]. Therefore, it is reasonable to use parallel calculation to obtain satisfying results in a reasonable amount of time.

The key requirements in realizing parallel calculation for simulation optimization include:1) it has high speedup for different number of processors, while with high quality of solutions; 2) it must be robust when network failures are encountered between some processors.

The paper is organized as follows. Section 2 provides a basic view for multi-demes GAs, and Section 3 analyzes the drawbacks of existing parallel calculation models. Section 4 studies the feasibility and speedup of multiple-demes GAs in master-slave mode. Section 5 gives an example in semiconductor device synthesis to show the efficiency of the methodology.

## 2. Multiple-demes GAs (MDGAs) [10]

### 2.1 Algorithm description

For multiple-deme GAs, the population is partitioned into a number of isolated demes, each one of which evolves independently, trying to optimize the same function. Typically, a rings-type neighborhood structure (Fig. 1) is defined over this set of demes, so that each deme swaps some good individuals with its neighbors in a given rate. This swapping activity is called migration.

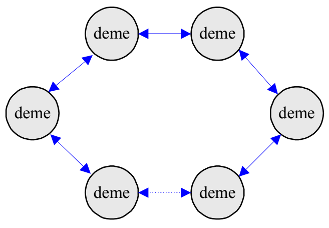

Fig. 1. Rings-type neighborhood structure

Fig 2 shows the principle of standard GAs (sGAs) in each deme, which is considered as an evolved population. For each generation, the population is evolved by genetic operations, such as mutation, crossover, etc., and updated by replacing some individuals.

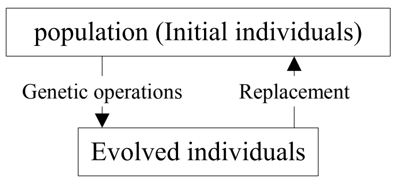

Fig. 2. Principle diagram of sGAs in a deme

The relation of solution quality with the evolution generations $G_n$ is mainly affected by the number of demes $M$, and for each deme, the sub-population size $S_{is}$, and evolved individuals in each generation $S_{es}$, interval generation of migration $k$, migration rate $m_r$. For the convenience of comparison, the initial sub-populations in different demes use same individuals. The total number of individuals needed to be evaluated during the evolution process is equal to $S_{is}+ G_n*S_{es}*M$.

## 2.2 Performances of MDGAs

From the previous studies [10-12], the MDGAs have an appealing trait in that they often provide high quality solutions than sGAs, with several striking characteristics: 1) its decentralized search, which allows speciation (different demes evolve towards different solutions); 2) the larger diversity levels (many search regions are sought at the same time); 3) an intensive exploration performed by all the demes; and 4) exploitation inside the demes, i.e., refining the better partial solutions found at any moment.

Migration policy is an important design factor, due to two reasons: 1) enhances the number of samples with high fitness in each deme to speed up evolution, so the frequent migrations induce fast convergence; 2) introduces new genes into each deme, so reasonable migrations enhance the diversity of demes to fighting premature convergence, while in the same time, it blends the genes with the cycle $M*k$, i.e. frequent migrations will decrease the diversity of demes. "Considerable isolation" [11] is very important.

To make an intuitionistic view, a test based on a nonlinear function $G_9$ [13] is used to show the typical properties of MDGAs. The parameter $F=Log(F_{calc}-F_{opt})$ is used to describe the convergence quality. Where $F_{calc}$ represents the mean value of the best evolution results (here we perform calculation 100 times), and $F_{opt}$ is the optimum value. For MDGAs, $S_{is}$=100, $S_{es}$=20, $m_r$=2%, $G_n$ =300. Fig. 3 shows the relations between $F$ and $G_n/k$ (i.e. migration times), when $M$=3, 10.

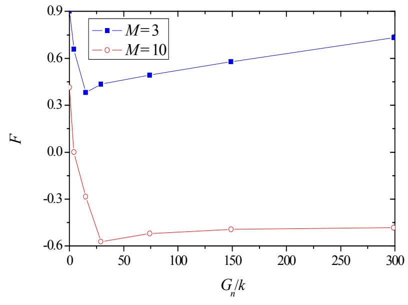

Fig. 3. $F$ versus $G_n/k$, when $G_n$=300

The bound case that $G_n/k$=300 is equivalent to a single large population approximately, and the bound case that $G_n/k$=1 means no migration occurs among demes.

It can be found: 1) with appropriate migration times, the multiple-demes algorithm can get better convergence quality than the above bounding cases; 2) the quality of solution is more stable for large $M$ with high frequent migrations, which is due to the slowly blending of genes.

Fig. 4 shows the relations between $F$ and $M$, where $M*G_n$=3000, $k$ =10. For $M$=1, it represents a sGAs. It can be found that the best $F$ occurs with appropriate $M$ and $G_n$, since $F$ is decreased when $M$ or $G_n$ is increased.

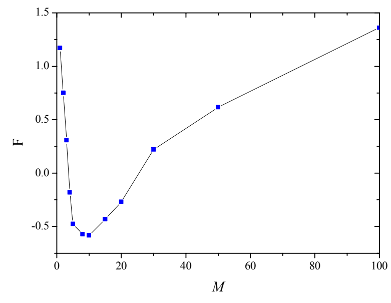

Fig. 4. $F$ versus $M$, when $M*G_n$=3000

The superior of MDGAs can be explained by shifting-balance theory [14]. This began with the idea that interactions among loci could result in populations achieving a genotypic state that was locally relatively fit (a 'peak'), but was not as fit as possible because of intervening unfit genotypes ('valleys'). It suggested several mechanisms by which evolution might allow a population to reach higher adaptive peaks, including novel favorable mutations, relaxation of selection, qualitative changes in the environment.

Overall, one may conclude that the multiple-deme model is well suited to observe the well-balanced role of diversification and intensification during the search process. Each deme might be seen as intensification in a particular region and a great number of demes provide some diversification in the global space.

## 3. Parallel calculation mode[10-12]

The parallel GAs (PGA) has been implemented by several ways, that is, coarse-grained, neighborhood and master-slave mode, which are the classification according to the connection topology of multi-processors.

To ensure the reduction of calculation time, the total number of individuals had better not be increased. As shown in Fig. 4, in order to provide high quality results, the parameters of PGAs, such as number of demes, population size, etc., should be well adjusted.

However, for coarse-grained PGAs, the number of demes is equal to the number of processors. For neighborhood PGAs, the population size is multiple of the number of processors. The additional limitations will enhance the difficulty to find high quality solutions.

Moreover, for coarse-grained and neighborhood PGAs, the neighboring processors will swap information, which makes the demes should be evolved synchronously. Then it needs to wait for the slowest processor. For the worst case, the system will be stopped if any network failure is encountered between the processors.

The master-slave mode can avoid the above drawbacks of coarse-grained and neighborhood mode. And they have many advantages: 1) be easily to implement, and significant reduction for calculation time are possible, especially when communication time can be neglected; 2) provide an ability to balance the speed difference between processors by allocating the individuals to idle processors, even network failure occurs between master and slaves.

However, some key drawbacks of the conventional master-slave mode should to be emphasized: 1) it is based on sGAs, which provide lower quality solutions; 2) the communication time and the unbalanced load among processors may reduce the speedup.

## 4. Multiple-demes GAs in master-slave mode (MDMS)

As shown in Fig. 5, a scheduler in master side will be used to collect all active individuals in the demes of MDGAs, and deliver each individual to be calculated by the simulation model in any idle slave processor. An active individual means it is independently to other individuals that will be delivered to the slaves.

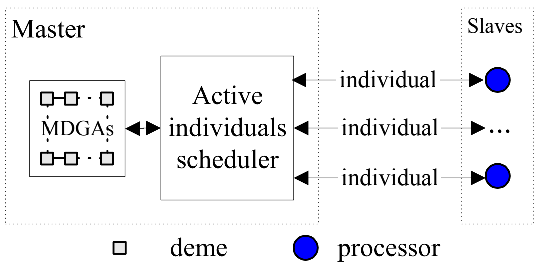

Fig. 5. The concept of MDMS

### 4.1 Feasibility from the effect of communication time

For the parallel calculation of simulation optimization, the evolution of an individual includes genetic operations, network communication, and simulation to evaluate the fitness. The simulation is often time-consumption, so the time for communication and genetic operations can be neglected. For an example, a typical device simulation by costs several minutes, while the communication time in local area network costs only several seconds, and the genetic operations costs only several milliseconds.

### 4.2 Speedup estimation

Suppose there have $N$ processors are used to realize the parallel calculation for simulation optimization, and each individual finishes its simulation in same time $T$.

As shown in Table 1, three type of GAs are compared, where $M_x$ is an integer and is large than 1, $G_x$ can be divided exactly by $M_x$. The total number of evolved individuals $S_{et}$ is same, which is equal to $S_x*G_x$. For case 3), the interval generation of migration is $k$.

Table 1. Different GAs to be compared

| Cases | $M$ | $S_{es}$ | $G_n$ |
|---|---|---|---|
| 1. sGAs(S) | 1 | $S_x$ | $G_x$ |
| 2. sGAs(L) | 1 | $S_x*M_x$ | $G_x/M_x$ |
| 3. MDGAs | $M_x$ | $S_x$ | $G_x/M_x$ |

For the convenience of discussion, we define a function **{**$x$**}** to represent the minimum integer that not less than $x$

$$\{x\} = \begin{cases} [x] & \text{if x is an integer} \\ [x]+1 & \text{if x is not an integer} \end{cases}$$

where $[x]$ means the integer part of $x$.

For master-slave mode, if there have $S_t'$ individuals that are independently to each other to be simulated at different $N$ processors, then the calculation time $T_{It}=T*\{S_t'/N\}$. Because the probability of that $S_t'$ can divide exactly $N$ is only $1/N$, for most case, some processors will be idle in the last stage.

For GAs, the evolution process is a Markov chain, which the current generation is effected by the previous generation. Only the individuals in same generation are independently to each other. The real calculation time is

$$T_{Rt} = T_i + T_{et}$$

Where $T_i = T*\{S_i/N\}$ and $T_{et}$ are the calculation time for initial and evolved individuals, respectively.

The speedup for parallel GAs is defined as

$$R_s = T_s/T_{Rt} \qquad (1)$$

where $T_s$ is the calculation time for the serial version.

Since $T_i$ is same for the GAs to be compared, in order to estimate the differences of the calculation time for different GAs, only $T_{et}$ is needed to be discussed. The minimum calculation time for all the evolved individuals $T_{et,min}= T*\{S_{et}/N\}$ occurs when they are all independent to each others, such as in random search.

For case 1), the calculation time is

$$T_{et,\,1} = \{S_x/N\}*G_x*T \qquad (2)$$

Suppose $S_x=X_a*N+X_b$, where $X_a=[S_x/N]$, $X_b=S_x\%N$ is the modulus. Then

$$T_{et,\,1}/T = X_a*G_n+\{X_b/N\}*G_n$$

If $X_b \neq 0$, then

$$T_{et,\,1}/T = G_x*(X_a+1)$$

Since $T_{s,et}/T = S_x*G_x$, then from (1), the speedup for evolved individuals $R_{s,\,et}$ is

$$R_{s,\,et} = T_{s,et}/T_{et,\,1}=N-(N-X_b)/(X_a+1) \qquad (3)$$

If $X_a$ is small, i.e. $S_x$ is small, then $R_{s,et}$ may be small and unstable for large $N$.

For case 2), the calculation time is

$$T_{et,\,2} = \{S_x*M_x/N\}*G_x*T/M_x \tag{4}$$

For case 3), Fig. 6 shows the evolved individuals of $k$ generations between the interval of two migrations for multiple-demes GAs, where $S_{x,i,j}$ represents the evolved individuals at $j$th deme in $i$th generation.

For the same generation, the individuals in all the $M_x$ demes are actively, which can be sending to the scheduler for parallel calculation immediately.

Moreover, the demes are isolated to each others, so the individuals in different demes are independently. If the individuals in $S_{x,i,j}$ have been calculated, after few time for genetic operations, the individuals in next generation $S_{x,i+1,j}$ are actively and can be deliver to the scheduler for parallel calculation.

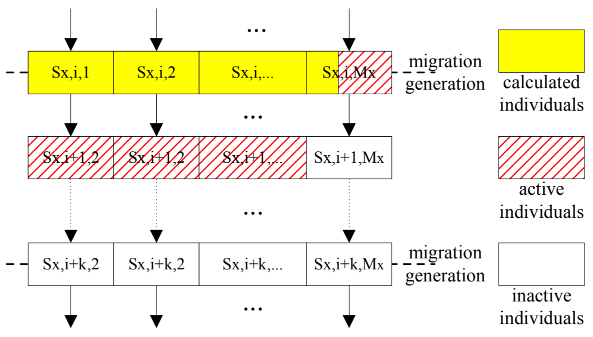

Fig. 6. The evolved individuals between two migrations

For the $i$th generation, the number of individuals $S_{xt,i}=M_x*S_x$, which are all active. Suppose $S_{xt,i}=X_a*N+X_b$, where $X_a=[S_{xt,i}/N]$, $X_b=S_{xt,i}\%N$. Then as shown in Fig. 6, after $X_a$ times parallel calculations, totally $X_a*N$ individuals are calculated, then there have $[X_a*N/S_x]$ demes in next generation are actively. For the $X_a+1$ times, there have $X_b+[X_a*N/S_x]*S_x$ active individuals. In order to avoid idle processors, it only needs to satisfy

$$N \le X_b+[X_a*N/S_x]*S_x \tag{5}$$

i.e. $S_{xt,i}\text{-}N \ge S_x*\{X_b/S_x\}\text{-}X_b$.

Suppose $X_b=X_c*S_x+X_d$, where $X_c=[X_b/S_x]$, $X_b=X_b\%S_x$. Then we should have

$$S_{xt,i}\text{-}N \ge S_x*\{X_d/S_x\}\text{-}X_d \tag{6}$$

If $S_{xt,i}>N$, then $S_{xt,i}\text{-}N \ge X_b$. To make (6) exists, it needs

$$X_b \ge S_x*\{X_d/S_x\}\text{-}X_d$$

i.e. $2X_d \ge S_x*\{X_d/S_x\}\text{-}X_c$ (7)

Since $S_x \ge S_x*\{X_d/S_x\}\text{-}X_c$. To make (7) exists, it needs

$$2X_d \ge S_x \tag{8}$$

Since $S_x*\{X_d/S_x\} \le S_x$, with (8), it has

$$S_x/2 \ge S_x*\{X_d/S_x\}\text{-}X_d \tag{9}$$

With (6) and (9), if $S_{xt,i}\text{-}N \ge S_x/2$, then (5) come into existence. It means enough active individuals in $i+1$ generation can be delivered to the idle processors. Analogically, it means that all the individuals in the interval generations between two adjacent migration generations can be seemed as whole independently to each others. The number of independent individuals is equal to $k*M_x*S_x$, then calculation time is

$$T_{et,3} = [G_x/(k*M_x)]*\{\,k*M_x*S_x/N\}*T$$
$$+\{((G_x/\,M_x)\%k)*\,M_x*S_x/N\}*T \tag{10}$$

Since for any integer $a\ge 0$ and any float $x\ge 0$, it has

$$\{ax\} \le a*\{x\}$$

Then from (2), (4), (10), we have $T_{et,3} \le T_{et,2} \le T_{et,1}$.

When $k$=1, then $T_{et,3}=T_{et,2}$; and when no migration occurred during the evolution, then $T_{et,3}=T_{et,min}$.

If $0 \le S_{xt,i}\text{-}N \le S_x/2$, the estimation for calculation time are complicated. However, the idle processors in a generation is $N\text{-}X_b\text{-}[X_a*N/S_x]*S_x$, which is less than $N\text{-}X_b$ in case 2). Then it still has $T_{et,3} \le T_{et,2}$.

If $S_{xt,i} \le N$, it means all the individuals in same generation are calculation by processors, then $T_{et,3}=T_{et,2}$.

## 5. A semiconductor device synthesis example

We have introduced the MDMS mode for device synthesis [15] to find parameters with desired electrical performances, which the simulator is PICSCES-2b.

The test example is a 0.35$\mu m$ FIBMOS device [16], which is composed of two parameterized structures: a fundamental MOSFET device structure (characterized by $L_{eff}$, $T_{ox}$, $X_j$, $N_{sd}$, $N_{sub}$, etc.) and a FIB implantation structure (characterized by $X$, *Dose*, and *Energy*).

The device performance of FIBMOS includes on current ($I_{on}$), off current ($I_{off}$), and dynamic output resistance ($R_{out}$). The device designables include lateral implantation position ($X$), dose, and energy. The device performance and designables are shown in Table 2, 3.

Table 2 Device performance of FIBMOS

| Name | Objective | Unit |
|---|---|---|
| $I_{on}$ | Maximum | A |
| $I_{off}$ | $I_{off}\le$1e-12 | A |
| $R_{out}$ | $R_{out}\ge$1e5 | Ω |

Table 3 Device designables of FIBMOS

| Name | Min | Max | Unit |
|---|---|---|---|
| *X* | 0.05 | 0.30 | μm |
| *Dose* | 2e13 | 2e18 | $cm^{-2}$ |
| *Energy* | 10 | 200 | Kev |

Table 4 Test results of device synthesis

| No. | *M* | *k* | $R_{s,\,N=9}$ | $R_{s,\,N=18}$ | $I_{on,mean}$ (*A*) |
|---|---|---|---|---|---|
| 1 | 1 | No | 6.943 | 10.650 | 0.0036791 |
| 2 | 3 | 1 | 8.408 | 15.783 | 0.0038174 |
| 3 | | 4 | 8.823 | 17.344 | 0.0038389 |
| 4 | | 10 | 8.913 | 17.698 | 0.0038439 |
| 5 | | No | 8.945 | 17.872 | 0.0038053 |
| 6 | 20 | 1 | 8.910 | 17.576 | 0.0038091 |
| 7 | | No | 8.948 | 17.879 | 0.0037944 |

Table 4 show the test results for different $M$ and $k$($k$=*No* means no migration), when $M*G_n$=120. For a deme, $S_{is}$=100, $S_{es}$=20, and $m_r$=2%. Where $R_{s,\ N=9}$ and $R_{s,\ N=18}$ are speedups when $N$=9, 18 respectively, and $I_{on,mean}$ is mean value for $I_{on}$. The tests are performed 5 times, respectively.

In all the cases, the speedup for different processors and the quality of solution of MDMS are better than sGAs (case No.1). Large $k$ produces large speedup, and the best speedup occurs when no migrations. The best result occurs when $M$=3, $k$=10, with acceptable speedup for different $N$. Users need to select appropriate parameters of MDMS, which provide large $k*M$, while with high quality solutions. Sometimes it needs a tradeoff according to their requirements. When $M$=20, the solution quality is not very well. For coarse-grained mode, If $N$=20 or more, $G_n$ should be enlarged to ensure the acceptable solution quality, which will increase the calculation time and decrease the speedup simultaneously.

## 6. Conclusion

In this paper, the MDMS, i.e. MDGAs in master-slave mode is applied to fulfill simulation optimization.

Since the demes are located on master processor, the parameters such as number of demes, population size, etc., can be well adjusted to provide higher quality results than in coarse-grained mode.

Furthermore, for master-salve mode, since the demes are isolated between two migrations, which provide more active individuals, is proved that the speedup of the MDMS is superior to the sGAs and original MDGAs.

It should be noticed that $M*k$ has important effects, not only for the quality of solutions, but also for the speedup. One can adjust other parameters of the MDMS to provide a larger $M*k$ in order to have larger speedup, while with acceptable quality for solutions.

The device synthesis example shows that it is an efficient way to utilizing the superior performance on improving the utilization efficiency of processors and maintaining the quality of solutions, while eliminate the limitations of the coarse-grained parallel mode.

The MDMS is very useful for the optimization for time-consuming system, such as TCAD, to reduce the total calculation time.

## REFERENCES

[1] Fu M C, Andradottir S, Carson J S, et al. Integrating optimization and simulation: research and practice. Proceedings of the Winter Simulation Conference. 2000: 610-616

[2] Glover F, Kelly J P, Laguna M. New advances and applications of combining simulation and optimization. Proceedings of the Winter Simulation Conference. 1996: 144-152

[3] Paul R J, Chanev T S. Simulation optimization using a genetic algorithm. Simulation Practice and Theory. 1998, 6(6): 601-611

[4] Azadivar F, Tompkins G. Simulation optimization with qualitative variables and structural model changes: a genetic algorithm approach. European J. Operational Research. 1999,113(1): 169-182

[5] Dengiz B, Alabas C. Simulation optimization using TABU search. Proceedings of the Winter Simulation Conference. 2000: 805-810

[6] Haddock J, Mittenthal J. Simulation optimization using simulated annealing. Computers and Industrial Engineering. 1992, 22(4): 387-395

[7] Youssef H, Sait S M, Adiche H. Evolutionary algorithms simulated annealing and tabu search: a comparative study. Engineering Applications of Artificial Intelligence. 2001, 14(2): 167–181

[8] Hasan M, AlKhamis T, Ali J. A comparison between simulated annealing, genetic algorithm and tabu search methods for the unconstrained quadratic Pseudo-Boolean function. Computers and Industrial Engineering. 2000, 38(3): 323-340

[9] Holland J H, Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, 1975

[10] Cantú-Paz E, Goldberg D E. Efficient parallel genetic algorithms: theory and practice. Computer Methods in Applied Mechanics and Engineering. 2000, 186(2): 221-238

[11] Alba E, Troya J M. Influence of the migration policy in parallel distributed GAs with structured and panmictic populations. Applied Intelligence. 2000, 12(3): 163–181

[12] Kohlmorgen U, Schmeck H, Haase K. Experiences with fine-grained parallel genetic algorithms. Annals of Operations Research. 1999, 90: 203–219

[13] Michalewicz Z, Schoenauer M. Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Computation. 1996, 4(1): 1-32

[14] Wright S. Evolution in Mendelian populations. Genetics. 1931, 16: 97–159

[15] Li Z, Xie X F, Zhang W J, Yang Z L. Realization of Semiconductor Device Synthesis with the Parallel Genetic Algorithm. Asia and South Pacific Design Automation Conference. 2001: 45 – 49

[16] Shen C C, Murguia J, Goldsman N, et al. Use of focused-ion-beam and modeling to optimize submicron MOSFET characteristics. IEEE Trans. Electron Devices, 1998, 45(2): 453-459