**Journal of Computer Science and Electrical Engineering**
**Print ISSN: 2663-1938**
**Online ISSN: 2663-1946**
**DOI: https://doi.org/10.61784/jcsee3079**

# PARALLEL GENETIC ALGORITHM FOR MAGNETOTELLURIC INVERSION WITH GPU

You Miao, Ge Cheng*
*Geological Exploration Technology Institute of Anhui Province, Hefei 230031, Anhui, China.*
*Corresponding Author: Ge Cheng, Email: 632202633@qq.com*

**Abstract:** A parallel genetic algorithm (GA) for magnetotelluric inversion with CUDA architecture is implemented for improving the accuracy and speed of traditional genetic algorithm. The algorithm is modified to adapt to the CUDA architecture for a more efficient computation. Model verification shows that the inversion computational speed has been dramatically increased with high computational accuracy under the parallel computing architecture. The CUDA architecture is proved to be a powerful tool for parallelizable problems in computational geophysics.
**Keywords:** Magnetotelluric inversion; Parallel genetic algorithm; CUDA architecture; Island based GA

## 1 INTRODUCTION

Magnetotellurics (MT) is a geophysical method that infers the subsurface conductivity distribution by measuring variations in the Earth's surface electromagnetic field. By collecting data on the natural electromagnetic field components at the surface, information about underground structures can be obtained. With a detection depth ranging from several hundred meters to several hundred kilometers, this method is widely used in structural studies, oil and gas exploration, and geothermal investigations.
The 1D MT forward modeling is assumed that the structure of the earth is consist with many horizontal layers. The electrical properties of each layer, which includes the resistivity and depth, are fixed values. The natural electromagnetic field is used as the source field. When this field is entering earth's surface, it will transform to uniform plane wave. The reflection waves of different frequency will be observed by special instruments. We can get detailed information of underground structure with these observation data.
Genetic algorithm (GA) is one of the early algorithms for simulating the genetic system. It was firstly proposed by Fraser in 1950s[1]. This method became popular through work of Holland in the early 1970s[2]. Genetic algorithm simulated genetic evolution of a special population in which individual traits (features) are expressed by genes. The main manipulation of genetic algorithm is selection and restructuring operations. Under a specific condition, the best individuals are selected and their genes are regrouped to evolve the population.
After being introduced to the field of geophysics for decades, genetic algorithm is used widely for solving geophysical inversion problems. Many geophysical optimization problems are nonlinear or based on nonlinear problems. Based on direct space sampling, the genetic algorithm can be used to solve the nonlinear problems without linearization processing. This algorithm is also a global search method, which is able to avoid the local minimum in model space searching. The above factors enable genetic algorithms to be used in MT inversion[3-5]. However, there are several aspects that effect the adoption of this algorithm. Firstly, the whole calculation includes a large amount of forward modeling operations, which results in a time-consuming progress. Secondly, the large number of parameters in most geophysical optimization problems can significantly reduce the efficiency of model search, which also effects the quality of the result and causes extra computational cost.
In this work we describe an implementation of genetic algorithm for solving the magnetotelluric inversion problem for layered earth model. Furthermore, we design a parallel genetic algorithm and make a computing program with CUDA toolkit, which is a powerful tool for parallel computing with graphic card. The program uses GPU (graphic processing unit) to do parallel computing instead of the serial computing with CPU (the central processing unit). With the accompaniment of parallel computing with GPU, the calculation speed has significantly increased and the inversion results are as good as using the serial programs with CPU.

## 2 MT FORWARD MODELING AND INVERSION

The MT forward modeling is a nonlinear problem. In this work we try to use nonlinear inversion to find the best model of underground electrical structure to fit the observing data on the ground.
For each frequency, the impedance Z will be calculated by Equation 1 from the information of resistivity and depth value of layered model[6-7]. Then the impedance Z is used to get apparent resistivity and impedance phase.

$$Z=F(\rho_1,\rho_2,...,\rho_n,h_1,h_2,...,h_{n-1},f)$$
$$\rho_a=\frac{|Z|^2}{2\pi f\mu_0},\Phi=\arctan(\frac{Re(Z)}{Im(Z)}) \quad (1)$$

where f denotes the frequency of plane wave. Z is impedance for frequency f. F is the forward modeling operator. n is the number of layers. $\rho_1,\rho_2,...,\rho_n$are resistivity values of layers,$h_1,h_2,...,h_{n-1}$ are depth values of layers. Here the bottom

layer of model is consider to be infinity. $\mu_0$ is the magnetic permeability of vacuum. $\rho_a$ Is apparent resistivity and $\Phi$ is impedance phase.

The MT forward modeling is a nonlinear system with the input parameters of model and frequency sequence. The output are apparent resistivity and impedance phase.

The objective of inversion is finding a model that corresponds to the minimum objective function. As defined in Equation 2, the objective function consists of the L2-norm of the difference between the observed (obs) and calculated (cal) apparent resistivity and phase (Perez-Flores and Schultz 2002).

$$OF=\|\rho_a^{obs}-\rho_a^{cal}\|^2+\|\Phi^{obs}-\Phi^{cal}\|^2 \quad (2)$$

## 3 ISLAND BASED PARALLEL GA INVERSION WITH CUDA

Genetic Algorithm (GA) is one of the global optimization methods. As the basic feature of these methods, global searching in whole model space is used to find the best solution. In the first step of GA, a population contains individuals is initialized randomly. Each individual in this population has its own chromosome, which includes the input data called genes. Then some steps of operations such as selection, crossover and mutation are used repeatedly to evolve the population to a new generation. An individual with the best chromosome will be found after some generations. The genes of the best chromosome also mean the best solution for the inversion problem.

In the progress of genetic algorithm, the evolution of each individual is independent with each other. It means that the selection, crossover and mutation operation is able to be paralleled. So we can use the powerful parallel computer architecture called CUDA (Compute Unified Device Architecture) to solve our problem effectively. We apply the genetic algorithm on GPU architecture. Different GPU memories and structures are used and organized to accelerate the intensity calculation. According the GPU hardware, there are global memory, local memory, shared memory, register, constant memory and texture memory. The global and local memory has a huge storage and very slow data I\O speed. Other memory has limited storage but a fast I\O speed as register. The GPU issue massive threads, which are organized in grids and blocks, to hide the memory latency. Each block in grid fix on a SM (stream multi-processor), and each thread in block fix on a SP (stream processor). The mapping for island based GA to GPU architecture will be introduced in the following sections[8].

In GA inversion progress, there is a total population consist with individuals. In this work, whole population is split to groups on different islands in same size. The gene of individual's chromosome are depth and resistivity value of underground layers. The real number code, which is convenient and intuitive, is used for model parameters. As shown in Figure 1, each individual is mapped to one thread and each population is mapped to one block (considered as one island).

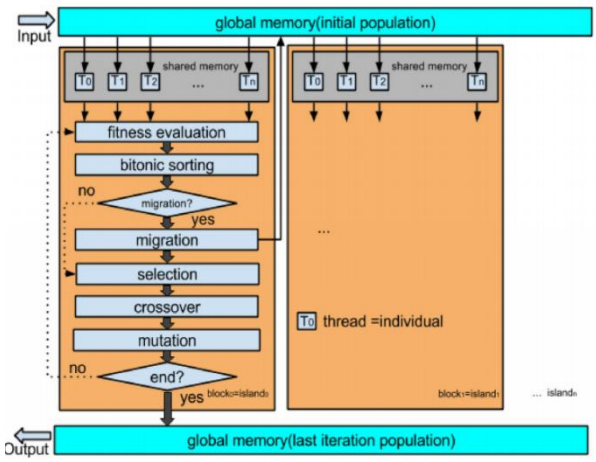


**Figure 1** Island Based Genetic Algorithm on CUDA Architecture

As shown in Figure 1, the GA steps such as selection, crossover and mutation are just in a population for each individual. That means each block or thread is independent in computation with a concurrency trait. The global memory has a low I\O speed, so we storage the individual information in each block's shared memory which has a very fast I\O speed. The global memory is used only in information exchanging between islands and the whole population such as migration and finishing calculation. In order to accelerate convergence, some individuals of each island will migrate to its neighboring island. Shared memory and global memory will exchange data in this step. The random numbers are used in most steps of GA progress. The cuRAND library from CUDA toolkit is used to generate random numbers in parallel.

## 4 IMPLEMENTATION

### 4.1 Population

In island based GA, each island has its population with individuals. The population evolves separately on its island. Each individual has its own chromosome. The genes of chromosome are input parameters of a forward modeling. In MT inversion, model parameters, which contain resistivity and depth value of each layer, are used as genes. In this work, an individual has one chromosome which includes parameter of one model.

### 4.2 Fitness

Here the fitness is equal to objective function. Individuals with the smaller fitness are better ones.

### 4.3 Selection

Tournament selection is used for this step for crossover operation. Every two neighboring threads (individuals) can be seen a pair. The individual with a less fitness value is chosen as one parent. The other parent is randomly selected from the whole population in each island.

### 4.4 Crossover

Every pair of individuals will be given a random number which will be compared with the crossover probability to decide whether to perform this operation or not. The arithmetic crossover shown in Equation 3 is performed in this step[8]. The pair of parents is replaced by a pair of off-springs after a crossover operation.

$$\begin{aligned} O_1 &= a \cdot P_1 + (1-a) \cdot P_2 \\ O_2 &= (1-a) \cdot P_1 + a \cdot P_2 \end{aligned} \tag{3}$$

where $O_1$ and $O_2$ represent off-springs, $P_1$ and $P_2$ represent parents and a called the aggregation weight in arithmetic crossover.

### 4.5 Mutation

The mutation probability is a constant number. A uniform random number is generated in parallel for each individual. This random number is compared with mutation probability to make decision for mutation. Then a gene of chromosome is selected randomly and changed to a random value within the constraints of model setting.
After all steps above, the population is replaced by the newly generated off-springs. Then the fitness of each individual is calculated in parallel. Afterwards the population goes on to the next iteration.

### 4.6 Migration

The migration operation occurs once for each 10 generations. In this step 10% best individuals are exchanged between two neighboring islands.

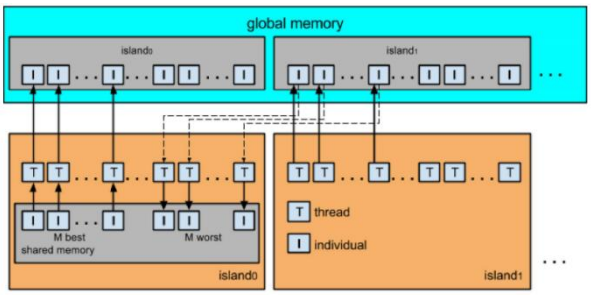


**Figure 2** Migration between Islands

Migration operation is illustrated in Figure 2. The exchange is done asynchronously in GPU main memory (called global memory). After being sorted by fitness, M worst individuals in an island are overwritten by M best individuals from a neighboring island. Both sorting and migrations are done in parallel for all individuals.

## 5 ODEL TEST

We use an artificial model to test our CUDA program. The range of frequency in MT model test is set from $10^{-3}$to $10^{3}$ with totally 31 frequency points. The real model is a five layers model (Table 1) and the populations iterate for 3000 generations. The graphic card is NVIDIA A5000 and CPU is Intel(R) i7-14900KF in this work.
The speedups on GPU against CPU are shown in in Figure 3. According to different scales of population sizes and number of islands, the speedup is from the 7.8 times minimum to 757 times maximum. It can be seen that GPU shows its superiority against CPU under a large population size and high number of islands.

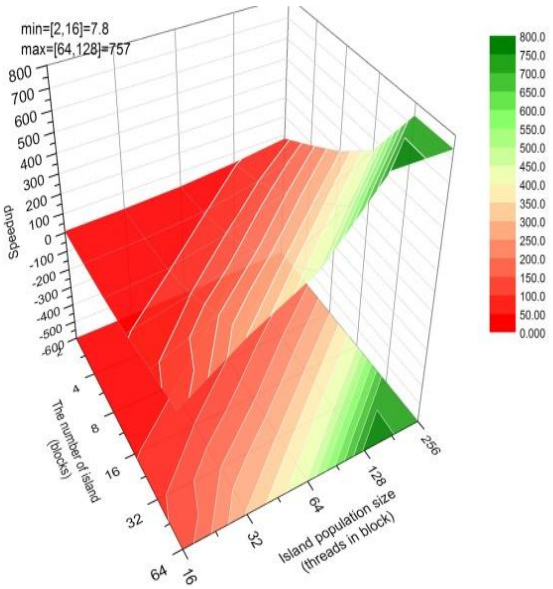


**Figure 3** The GPU Speedup of 5-Layers Mode Inversion

Table 1 gives the five layers model setting and GPU inversion result. The number of islands is 16 and population size on each land is 128. The inversion result in Tab.1 shows a good solution which is close to the true mode. The convergence variety with generations is shown in Figure 4. The fitness curve is oscillating at early generations but the convergence becomes stable after about a few generations.

**Table 1** Model Setting and GPU Inversion Result of a 5-Layers Artificial Model

| | Layer thickness(km) | | | | |
|---|---|---|---|---|---|
| Layer | 1 | 2 | 3 | 4 | 5 |
| True model | 1 | 1 | 2 | 2 | infinity |
| Inversion model | 0.96 | 0.96 | 1.70 | 3.10 | -- |
| Model | Resistivity($\Omega m$) | | | | |
| True model | 100 | 1000 | 100 | 10 | 1000 |
| Inversion model | 100 | 512 | 140 | 14 | 1126 |

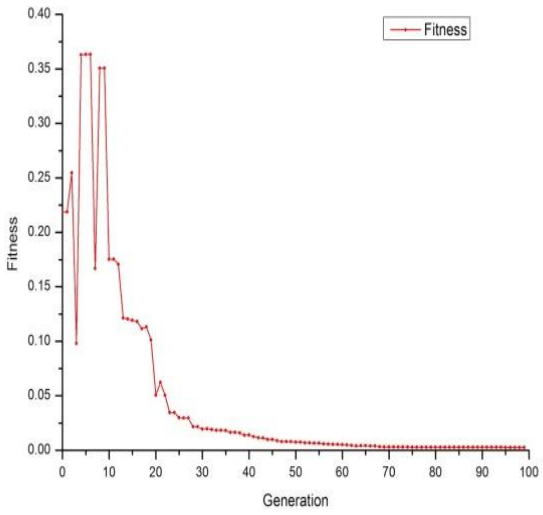


**Figure 4** The Convergence of Fitness with Generation

Table 2 presents the final optimal fitness results under different settings of island numbers and population sizes. The best fitness value, 0.00004, is achieved when the number of islands is 64 and the population size per island is 256. However, when the population size is small (e.g., 16), increasing the number of islands yields limited improvement in accuracy. Therefore, in practical applications, an appropriate combination of island number and population size can be selected based on available computational resources to balance efficiency and accuracy.

**Table 2** Best Fitness under Different Island Number and Size

| island size \ island number | 4 | 16 | 64 |
|---|---|---|---|
| 16 | 0.01225 | 0.0152 | 0.00068 |
| 64 | 0.0137 | 0.00063 | 0.00014 |
| 256 | 0.00077 | 0.00013 | 0.00004 |

## 6 CONCLUSION

The high speedup clearly proves that GPU has ability of accelerating the genetic algorithm for solving optimization problems. The experimental result also shows that the migration can significantly improve the efficiency to find the best solutions. The size and interval time of migration significantly affect the results of inversion. A less interval time means a more time consuming but will accelerate the diffusion of the optimal solution. In that condition, a more precise solution will be obtained. How to choose migration scheme need more model tests in the future.
In this work, we present an island based genetic algorithm inversion for MT on CUDA architecture. The model testing gives a good inversion result for the algorithm. The future work will focus to introducing parallel GA to other optimization problems in geophysics.

## COMPETING INTERESTS

The authors have no relevant financial or non-financial interests to disclose.

## FUNDING

This research is supported by The Science and Technology Project of Anhui Provincial Department of Natural Resources “Research on Deep Exploration Technology of Hidden Rock Masses-taking the Luzong Basin in Anhui as an Example” (2024-k-1).

## REFERENCES

[1] Eraser A S. Simulation of genetic systems by automatic digital computers. I. Introduction. Australian Journal of Biological Sciences, 1957, 10: 484-491.
[2] Holland J H. Genetic algorithms and the optimal allocation of trials. SIAM Journal on Computing, 1973, 2(2): 88-105.
[3] Everett M E, Schultz A. Two-dimensional nonlinear magnetotelluric inversion using a genetic algorithm. Journal of Geomagnetism and Geoelectricity, 1993, 45(9): 1013-1026.
[4] Perez-Flores M A, Schultz A. Application of 2-D inversion with genetic algorithms to magnetotelluric data from geothermal areas. Earth, Planets and Space, 2002, 54(5): 607-616.
[5] Roux E, Moorkamp M, Jones A G, et al. Joint inversion of long-period magnetotelluric data and surface-wave dispersion curves for anisotropic structure: Application to data from Central Germany. Geophysical Research Letters, 2011, 38(5).
[6] Wait J R. On the relation between telluric currents and the earth’s magnetic field. Geophysics, 1954, 19(2): 281-289.
[7] Wait J R. Theory of magnetotelluric fields. J. Res. NBS D, 1962, 66(5): 509-541.
[8] Pospichal P, Jaros J, Schwarz J. Parallel genetic algorithm on the cuda architecture. In Applications of Evolutionary Computation. Springer, 2010: 442-451.