Parallel Genetics Algorithm Application

Mohamed Faidz Mohamed Said [#1]

[#] *Universiti Teknologi MARA*
70300 Seremban Negeri Sembilan, MALAYSIA
[1] faidzms@ieee.org

***Abstract*—Parallel Genetic Algorithms (PGA) are powerful search techniques that already solve many problems in many different disciplines. They are being applied successfully to find acceptable solutions to problems in business, engineering, and science. In this paper, we focus on solving optimizing ion channel model using parallel genetic algorithm on graphical processors. The result shows that parallel genetic algorithm is a simplest technique.**

***Keywords*: parallel, genetic, algorithm, ion channel, application**

I. INTRODUCTION

Genetic algorithm (GA) is a search heuristic that mimics the process of natural selection. This heuristic was usually used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. There are many problems that been solved using parallel genetic algorithm, but this paper will only focus on solving optimizing ion channel model using parallel genetic algorithm on graphical processors [1].

Ion channels are trans-membrane proteins that open and close, leading to a change in ion flow across the membrane. They may be opened or closed by a number of factors, one of them being changes in membrane potential [2].

It was proposed a method for analyzing whole-cell recordings of voltage gated channels and demonstrated that the viability of models of voltage-dependent ion channels can be verified using a genetic optimization algorithm (GA) concurrently with a global fit of experimental data to the model. The method can be applied to all types of voltage-clamp recordings from different neuron classes [3].

* This research was supported by the Ministry of Education under the Fundamental Research Grant Scheme 600-RMI/FRGS 5/3 (158/2013) and Universiti Teknologi MARA.

II. LITERATURE REVIEW

The paper [5] talks about a genetic algorithm application using fuzzy processing times in non-identical parallel machine scheduling problem. The result shows the advantage of Genetics Algorithm (GA). It shows the possibility of using several initial populations while most of the other heuristics give only one solution.

A genetic algorithm for static load balancing in parallel heterogeneous systems by [6]. In this paper, they introduced a method based on genetic algorithms for scheduling and load balancing in parallel heterogeneous multi-processor systems. The simulations results indicate that Genetic Algorithm reduces total response time and it also increases utilization.

The project [7] researches about the automated fault-recovery for four-legged robots using parallel Genetic Algorithm. They propose to use a bio inspired learning algorithm to generate new behaviours on a four-legged robot against unexpected body damages. Experimental result shows that the robot could adapt to damages on different part of robot's body successfully since the learning algorithm can be accelerated using parallelism on multiple machines.

A scalable parallel genetic algorithm for the Generalized Assignment Problem described a scalable parallel genetic algorithm for solving large GAP instances by leveraging massively parallel computing. The result shows asynchronous migration strategy that is developed to improve the numerical performance of PGAP [5].

Parallel genetic algorithm based automatic path planning for crane lifting in complex environments was considered for lifting the path planning problem that takes inputs such as the plant environment, crane mechanical data, crane position, start and end lifting configurations to generate the optimal lifting path by evaluating costs and safety risks. The results show that the method can efficiently generate high quality lifting paths in complex environments [6].

The paper [7] shows research about an efficient cross over architecture for hardware parallel implementation of genetic algorithm. The main idea was based on the efficient use of a genetic algorithm's cross over operator to enhance the speed of algorithm to reach an optimal solution. The properties of FPGAs such as flexibility and parallelism help this purpose.

Selecting informative rules with parallel genetic algorithm in classification problem by [8] show the result that the experiments on data sets validate the effectiveness of the new model and indicate that the model is powerful for volumetric data set.

The researchers [9] do the work about flood simulation using parallel genetic algorithm with integrated wavelet neural networks and the result indicated that the parallel GAWANN has strong capability of rain-run off mapping as well as computational efficiency and is suitable for applications of flood simulation in arid areas.

III. METHODOLOGY

A genetic algorithm (GA) is a search algorithm based on the mechanisms of Darwinian evolution. It was used in random mutation, crossover and selection operators to breed better models or solutions from an originally random starting population [10]. Each individual in the population was described by a parameter set and the model is evaluated for each set. A search space was defined for each parameter to avoid parameter combinations causing instability to the set of differential equations while covering most of the physiological range expected for the parameters. For rate constants (k) the range was set from 0 to 2000 1/s, for voltage-dependence parameters (z) from zero to 2000 1/V, and zero to 100 PS for the conductance.

The surrogate data was subtracted from data generated by the same voltage-clamp protocols according to a parameter vector describing each individual in the population and the result was squared and summed.

$$\chi^2 = \frac{1}{MN}\sum_{i-1}^{M}\sum_{j-1}^{N}(I_{ij} - i_{ij}) \quad (1)$$

where N is the number of voltage traces in the protocol, M is the number of time points every trace, *Iij* is the experimental or surrogate voltage-clamp current and *iij* is the simulated voltage-clamp current [11-13].

The population was sorted according to the value of the cost function of each individual (Eq. (1)) and a new generation was created using selection, crossover, and mutation as operators. Selection was achieved by a tournament in which two pairs of individuals were randomly selected; the individual with the better score from each pair was transferred to the next generation.

Three simulation environments were used here; two applied GPUs and used a high-performance Linux cluster. The high-performance Linux cluster was the one that was used in the previous work [14].

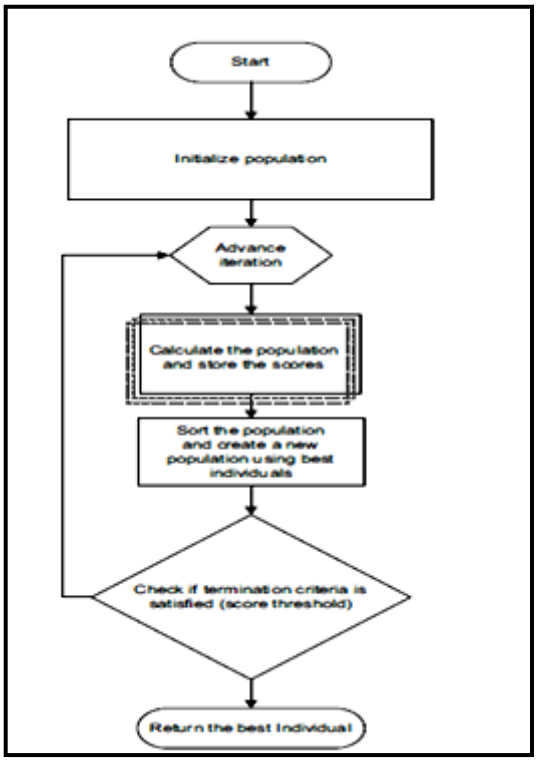

Figure 1. The flow chart of the optimization algorithm model using a genetic algorithm (GA). Flow chart of the optimization algorithm model using a genetic algorithm (GA).

IV. RESULTS

To simulate the kinetics of voltage-gated ion channels it previously used the simulation package NEURON implemented with MPI on a Linux cluster [7] [9]. Programming GPUs using CUDA required writing a different code in C. Thus, a preliminary step in the development of the application was to create a flexible description of voltage-gated channel mechanisms in C. This was achieved by using a simple configuration file. An example of such a file, describing Model B, is given below. The createQmat kernel defining the Q matrix [16] is automatically

generated using a Python script from this configuration file.

```
model:
{
        nStates = 3;    //Total number of states in the model
        nParams = 9;    //Number of parameters describing the model's kinetics
        eRev = -100;    //The reversal potential of the channel
        nOpenStates = 1;        //Number of open (permeable) states in the model

//Name, fit range and the value used in generating surrogate data for each params
params = (
                {name = "a12"; min = 0.0; max = 1.0; val = 0.05;},
                {name = "z12"; min = 0.0; max = 1.0; val = 0.05;},
                {name = "a21"; min = 0.0; max = 1.0; val = 0.05;},
                {name = "z21"; min = 0.0; max = 1.0; val = 0.05;},
                {name = "a23"; min = 0.0; max = 1.0; val = 0.05;},
                {name = "z23"; min = 0.0; max = 1.0; val = 0.05;},
                {name = "a32"; min = 0.0; max = 1.0; val = 0.05;},
                {name = "z32"; min = 0.0; max = 1.0; val = 0.05;},
                {name = "gmax"; min = 0.0; max = 10.0; val = 5.0;}
        );
        //Transition state equations describing the model
        rates = [
                "k12 = a12*exp(z12*v)",
                "k21 = a21*exp(-z21*v)",
                "k23 = a23*exp(z23*v)",
                "k32 = a32*exp(-z32*v)"
        ];
        openStates = [3];       //An array describing the indices of the states.
};
```

Figure 2. Phyton script from configuration file

```
model:

{
        nStates = 5;
        nParams = 6;
        eRev = -100;
        nOpenStates = 1;

params = (
                {name = "a1"; min = 0.0; max = 1.0; val = 0.01;},
                {name = "a2"; min = 0.0; max = 40.0; val = 10;},
                {name = "a3"; min = 0.0; max = 40.0; val = 10;},
                {name = "b1"; min = 0.0; max = 1.0; val = 0.125;},
                {name = "b2"; min = 0.0; max = 200.0; val = 80;},
                {name = "gmax"; min = 0.0; max = 10.0; val = 5.0;}
        );

        rates = [
                "k12 = 4*a1*(a2-v)/(exp((a2-v)/a3))",
                "k21 = b1*exp(-v/b2)",
                "k23 = 3*k21/4",
                "k32 = 2*k21",
                "k34 = 2*k21/4",
                "k43 = 3*k21",
                "k45 = k21/4",
                "k54 = 4*k21",

        ];
        openStates = [5];
};
```

Figure 3. Hodgkin–Huxley voltage-gated potassium code.

This format of model description also allows a non-Markovian description of ion channels. In principle, a Markovian chain of transitions can also describe Hodgkin–Huxley models. For example, the Hodgkin–Huxley voltage-gated potassium conductance can be described by the following scheme and script:

$$C \underset{\beta}{\overset{4\alpha}{\rightleftharpoons}} C \underset{2\beta}{\overset{3\alpha}{\rightleftharpoons}} C \underset{3\beta}{\overset{2\alpha}{\rightleftharpoons}} C \underset{4\beta}{\overset{\alpha}{\rightleftharpoons}} O$$

$$\alpha = 0.01(10 - v)/(\exp((10 - v)/10) - 1)$$
$$\beta = 0.125\exp(-v/80) \qquad (2)$$

Since a code different from NEURON is used to solve the equations, it was necessary to validate the implementation. The simulation of the Markov channel models is performed and described in the methods using both MATLAB and NEURON with the same set of parameters and compared the resulting ionic currents. Both the MATLAB and NEURON implementations provided results almost identical to the CUDA implementation. The mean error in all cases tested was smaller than 0.05%.

Parallelization of a genetic algorithm on a multi-core cluster is straightforward. A population of parameter vectors is created on the master node. Each individual is sent to a different slave node which simulates the response of the model to different voltage clamp protocols, generates a set of data vectors, subtracts these vectors from the surrogate data vectors and calculates _2 (Eq. (2)). This is then returned to the master node (Figure 2). The master node applies the operators of selection, mutation and crossover to the population. The modified population is repeatedly sent to the slaves until the termination criteria are met.

V. CONCLUSION

Genetic optimization algorithms have previously been applied to constrain parameters of Markov ion channel models using high performance Linux clusters. Computational exhaustion is encountered due to the complexity of the models and the amount of electrophysiological data. Here it is shown that, using a commercially available GPU, it is possible to

speed up the execution of the algorithm on a standard desktop computer. This substantial increase in speed was achieved by using the shared and constant memories of the GPU for rapidly accessed variables and reducing the memory copied to and from the GPU. This approach can be applied to speed up other data-intensive applications requiring iterative solutions of differential equations.

ACKNOWLEDGEMENTS

The author would like to thank the Ministry of Education for the FRGS grant 600-RMI/FRGS 5/3 (158/2013). The author is with the Department of Computer Science, Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Malaysia (e-mail: mohdfaidz@uitm.edu.my).

REFERENCES

[1] Baslıgil, P. A. H., A genetic algorithm application using fuzzy processing times in non-identical parallel machine scheduling problem. *Advances in Engineering Software*, *45*, 272–280, 2012.

[2] Garshasbib, M. E. M. S., A Genetic Algorithm for Static Load Balancing in Parallel Heterogeneous Systems. *Procedia - Social and Behavioral Sciences*, *129*, 358 – 364, 2014.

[3] Kim, H. P. K. J., The Automated Fault-Recovery for Four-Legged Robots using Parallel Genetic Algorithm. *Procedia Computer Science*, *24*, 158 – 166, 2013.

[4] Wang, Y. Y. L. S., A scalable parallel genetic algorithm for the Generalized Assignment Problem. *Parallel Computing*, *46*, 98–119, 2015.

[5] Zheng, P. C. Y. C. I. C. J., Parallel genetic algorithm based automatic path planning for crane lifting in complex environments. *Automation in Construction*, *62*, 133–147, 2016.

[6] Eichner H, Klug T, Borst A. Neural simulations on multi-core architectures. Front Neuroinform; 3:21, 2009.

[7] Gurkiewicz M, Korngreen A. A numerical approach to ion channel modelling using whole-cell voltage-clamp recordings and a genetic algorithm. PLoS Comput Biol; 3:e169, 2007.

[8] Gurkiewicz M, Korngreen A. Recording analysis, and function of dendritic voltagegated channels. Pflugers Arch; 453:283–92, 2006.

[9] Gurkiewicz M, Korngreen A, Waxman SG, Lampert A. Kinetic modeling of Nav1.7 provides insight into erythromelalgia-associated F1449V mutation. J Neurophysiology; 105:1546–57, 2011.

[10] Hines ML, Carnevale NT. Expanding NEURON's repertoire of mechanisms with NMODL. Neural Computing; 12:995–1007, 2000.

[11] Hines ML, Carnevale NT. NEURON: a tool for neuroscientists. Neuroscientist; 7:123–35, 2001.

[12] John ES, Jan S, David JH, Kirby LV, Wen-mei WH, Klaus S. High performance computation and interactive display of molecular orbitals on GPUs and multi-core CPUs.In : Proceedings of 2nd workshop on general purpose processing on graphics processing units. Washington, D.C.: ACM; 2009.

[13] Keren N, Bar-Yehuda D, Korngreen A. Experimentally guided modelling of dendritic excitability in rat neocortical pyramidal neurones. J Physiol; 587: 1413–37, 2009.

[14] Keren N, Peled N, Korngreen A. Constraining compartmental models using multiple voltage recordings and genetic algorithms. J Neurophysiol; 94:3730–42, 2005.

[15] Nageswaran JM, Dutt N, Krichmar JL, Nicolau A, Veidenbaum AV. A configurable simulation environment for the efficient simulation of large-scale spiking neural networks on graphics processors. Neural Networks; 22:791–800, 2009.

[16] Won-Ki J, Beyer J, Hadwiger M, Blue R, Law C, Vazquez-Reina A, et al. Analysis tools for large-scale neuroscience data sets. IEEE Comput Graph Application ;30:58–70, 2010.