2015 International Symposium on Nonlinear Theory and its Applications
NOLTA2015, Kowloon, Hong Kong, China, December 1-4, 2015

GPU Acceleration of Method for Solving Optimization Problems Using Chaotic Neural Networks

Toshihiro Tachibana† and Masaharu Adachi‡

†Department of Applied Computer Sciences, Shonan Institute of Technology
1–1–25 Tsujido-Nishi-Kaigan, Fujisawa, Kanagawa 251–8511, Japan
‡Department of Electrical and Electronic Engineering, Tokyo Denki University
5 Senju-Asahi-Cho, Adachi-ku, Tokyo 120–8551, Japan
Email: tachibana@sc.shonan-it.ac.jp, adachi@eee.dendai.ac.jp

Abstract—Recently, graphics processing units (GPUs) are used for general purpose computing which is called GPGPU. A feature of GPGPU is executing parallel computation by many GPU cores. In this paper, GPGPU calculation is used for several methods of optimization problems. We show a reduction of computation time by applying GPGPU in the execution of a method for quadratic assignment problems using chaotic neurodynamics and of a method for multi-objective problems using particle swarm optimization and chaotic neurodynamics.

1. Introduction

The authors have proposed a method for solving multi-objective optimization problem[1]. The proposed method switches more than two particle swarm optimization (PSO) methods[2] with switching by chaotic neurodynamics[3].

Generally, a method for combinatorial optimization problems using chaotic neurodynamics needs to set optimum parameters. However, searching the optimal parameters is difficult. Therefore, we consider the parameter search problems as multi-objective optimization problems.

In this paper, the proposed method is applied to parameter search problems for solving quadratic assignment problems (synchronous exponential chaotic tabu search, SECTS)[4]. The proposed method finds equal or better parameter values for the search performance than that by known values. However, this parameter search problem is very large computational cost problem. We use general-purpose parallel computation using graphics processing units (GPU) called GPGPU. Therefore, we try to speed up the computation by using GPGPU. Hence, the calculation by GPGPU succeeded in finding a solution with high speed as compared with calculation by the CPU.

2. Proposed method

The multi-objective optimization problems (MOP)[5] is an optimization problem when there are more than two objective functions.

We propose an algorithm for switching method that uses a chaotic neural network. A feature of this method is that the number of neurons is equal to the number of algorithms. The updating of the internal state of each neuron is represented by the following set of equations.

$$\xi_i(t+1) = -\beta_{sw}\Delta_{sw}(t), \tag{1}$$

$$\eta_i(t+1) = -W\sum_{k=1,k\neq i}^{N} x_k(t), \tag{2}$$

$$\zeta_i(t+1) = -\alpha_{sw}\sum_{d=0}^{t} k_r^d x_i(t-d) + \theta, \tag{3}$$

$$x_i(t+1) = f(\xi(t+1)+\eta_i(t+1)+\zeta_i(t+1)) \tag{4}$$

where, $x_i(t)$ is the output of the i-th exchanging neuron at time t. $\xi_i(t)$ is the internal state for the gain effect of the exchanging neuron. $\eta_i(t)$ is the internal state for the feedback effect of the exchanging neuron. $\zeta_i(t)$ is the internal state for the tabu effect of the exchanging neuron. k_r is a decay parameter of the gain effect. $\Delta_{sw}(t)$ is a gain of the objective function value when the candidate exchange is executed. θ is a positive bias. α_{sw} is a scaling parameter of the internal effect. β_{sw} is a scaling parameter of the gain effect.

The proposed method[6] uses switching three particle swarm optimization (PSO) methods[2]. These PSO methods are shown in below.

The first method is multi-objective optimization PSO (MOPSO) [7]. The second method is optimized multi-objective PSO (OMOPSO) [8]. This method is added mutations such as in the genetic algorithm to MOPSO. The third method is speed-constrained multi-objective PSO (SMPSO) [9]. This method is added a limit to the speed of particle motions to OMOPSO.

3. Formulation of parameter search for QAP

Reference [10] shows the following. When we minimize the spatial mutual information, a result of the QAP is improved. One of objective functions is minimizing spatial mutual information.

We were focusing on autocorrelation of internal state for

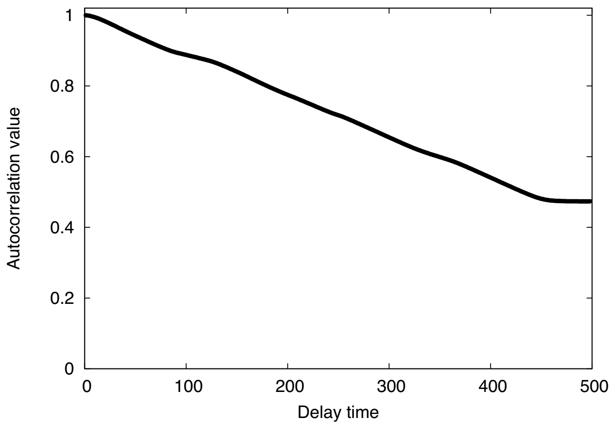

Figure 1: Example of the autocorrelation when SECTS was achieving to the global solution.

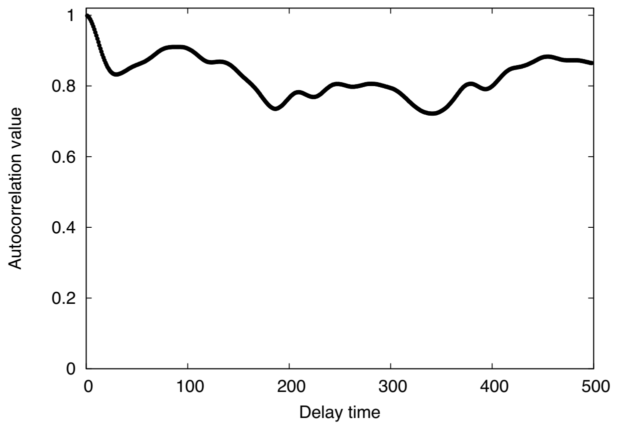

Figure 2: Example of the autocorrelation when SECTS was NOT achieving to the global solution.

the tabu effect of the exchanging neuron.

$$\phi(\tau) = \sum_{m=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N}\zeta_{ij}(m)\cdot\zeta_{ij}(m+\tau) \quad (5)$$

$$(\tau = 0, 1, 2, \cdots, N),$$

$$A(\tau) = \phi(\tau)/\phi(0) \quad (6)$$

where, $A(\tau)$ is the autocorrelation of constituent neurons in the neural network. $\zeta_{ij}(\cdot)$ is the internal state for the tabu effect of SECTS.

As a preliminary experiment, we used Lipa20b problem [11]. It is reported that SECTS achieves to the global solution in every trial [10]. We set SECTS's parameters values as $\alpha_\zeta = 1.0$, $\alpha_\eta = 0.1$, $\beta = 0.4$, $k_r = k_f = 0.9$, $R = 0.1$. Fig. 1 shows an example of the autocorrelation when SECTS was achieving to the global solution. Fig. 2 shows an example of the autocorrelation when SECTS failed to achieve the global solution.

From these results, we confirmed that as the delay time τ increases, $A(T)$ is reduced. We used this property as an evaluation function of the parameter search. In concrete, we evaluated the summation of $A(\tau)$ delay time from $\tau = 0$ to a certain time. In the case shown in Fig. 1, the summation of $A(\tau)$ became a small value. However, in the case shown in Fig. 2, the summation of $A(\tau)$ became large value. The summation of $A(\tau)$ can be also useful for evaluating periodicity at a certain range of summation.

Therefore, we formulate two objective function are shown in the following.

$$\text{minimize } f_1(\mathbf{r}) = \sum\nolimits_{\tau=0}^{T} A(\tau), \quad (7)$$

$$\mathbf{r} = \{\alpha_\eta, k, R\},$$

$$\text{subject to } 0 < \alpha_\eta < 1,\ 0 < k < 1,\ 0 < R < 1,$$

$$\text{minimize } f_2(t) = \frac{1}{n^2}\sum\nolimits_{i=1}^{n}\sum\nolimits_{j=1}^{n} MI_{ij} \quad (8)$$

$$MI_{ij} = H(x_i(t)) - H(x_i(t)|x_j(t-1)),$$

where, α_η is the scaling parameter of SECTS, k is the time decay constant of SECTS. We set $k = k_r = k_f$. R is a positive bias of SECTS, MI_{ij} denotes the spatial mutual information of the i th neuron and the j th neuron of SECTS, $H(x_i(t))$ is the entropy of the output of the i th neuron at time t, $H(x_i(t)|x_j(t-1))$ is the conditional entropy of $x_i(t)$ and $x_j(t)$.

4. Application of GPGPU to Proposed Method

A parameter search of SECTS is a very expensive problem in computation cost. The first one is the procedure to update the position and the speed of particles of PSO. The second one is the procedure that updates CNN of SECTS. These procedures can be realized in parallel computing. We can accelerate PSO and SECTS by applying GPGPU. In our previous work[12], SECTS was implemented in GPGPU.

4.1. Parallel computation of PSO

PSO requires the calculation of the position and the velocity of many particles. The calculation of each particle the same; therefore, calculation of one particle is can be executed as one thread.

4.2. Parallel computation of SECTS

We have implemented SECTS [12]. We use the same method as in [12] in this paper.

5. GPGPU Computer

We show the specifications of two computers used in the experiments in Tables 1 and 2.

Table 1: Spec of Computer 1

CPU1	Intel Xeon E5–2620 (octa core, 2.0GHz)
CPU2	Intel Xeon E5–2620 (octa core, 2.0GHz)
RAM	DDR3–1600 16GB (quad channel)
GPU	nVIDIA NVS 300 (VRAM: DDR3 512MB)
GPGPU	nVIDIA Tesla K20c (VRAM: GDDR5 5GB)
OS	Microsoft Windows Server 2012 R2 (64bit)
ADE	Microsoft Visual Studio 2013 Update 4
	CUDA Toolkit 6.5 Release

Table 2: Spec of Computer 2

CPU	Intel Core i7 2820QM (quad core, 2.3GHz)
RAM	DDR3–1333 8GB (dual channel)
GPU	nVIDIA GeForce GT 540M (VRAM: DDR3 1GB)
OS	Microsoft Windows 8.1 Update (64bit)
ADE	Microsoft Visual Studio 2013 Update 4
	CUDA Toolkit 6.5 Release

The computer 1 is a high-performance configuration that has two physical CPU. This CPU has 8 physical cores per one unit that is a maximum of 16 cores per one unit, including Hyper-Threading Technology. Also, the GPGPU board of the computer 1 in Kepler architecture, the number of single precision CUDA core is 2496, the number of double-precision CUDA core is 832.

However, the computer 2 is a general configuration that has one physical CPU. This CPU has 4 physical cores that are a maximum of 8 cores, including Hyper-Threading Technology. Also, the GPU board of the computer 2 in Fermi architecture, the number of CUDA core is 96.

6. Results of numerical experiment

We numerically evaluated the performance of the proposed method for some benchmark problems of QAPLIB[11], by performing 10 trials for each problem. We show results of the average of computation time of the proposed method and that of the conventional method in Fig. 3 and Table 4. In Fig. 3 and Table 4, the calculation using GPGPU is about 4 to 8.5 times faster than calculating using the CPU. From these result, we find that efficiency of using GPGPU in the computer 2 is higher than the computer 1.

From Fig. 3 and Table 4, the computer 1 is seem to be slower than the computer 2. The difference is caused by the following two factor. The computer 1 is 2.5GHz maximum frequency when turbo boost is in use. However, the computer 2 is a 3.4GHz maximum frequency when turbo boost is in use. Thus, the difference can be caused by the effect when it is calculated in a single thread. Also, the difference between Kepler and Fermi architectures was also affected for the GPU.

7. Conclusion

In this paper, we have proposed a method of the parameter search of QAP in high speed by using the GPGPU. As a result, it is possible to realize 8 times faster as compared to the calculation by the CPU as maximum performance.

References

[1] T. Tachibana, M. Adachi, "Method for Solving Optimization Problems Using Algorithm Switching by Chaotic Neural Networks", *Proc. of NOLTA 2014*, pp. 329–332 , 2014.

[2] J. Kennedy, R. Eberhart, "Particle Swarm Optimization". *Proc. of IEEE Neural Networks*, 1942–1948, 1995.

[3] K. Aihara, T. Takabe, M. Toyoda, "Chaotic Neural Networks", Physics Letters A, vol. 144, pp. 333–340, 1990.

[4] N. Yokota, Y. Horio, K. Aihara, M. Hasegawa, "A Modified Synchronous Exponential Chaotic Tabu Search for Quadratic Assignment Problems", Technical Report of IEICE, vol. NLP107, no. 561, pp. 49–54, 2008 (in Japanese).

[5] C. A. Coello Coello, G. B. Lamont, D. A. Van Veldhuizen, "Evolutionary Algorithms for Solving Multi-Objective Problems", Springer, 2007.

[6] T. Tachibana, M. Adachi, "Solving Multi-objective Optimization Problems Using Particle Swarm Optimization Methods with Switching by Chaotic Neurodynamics", *Technical Report of IEICE*, **111**(498), 51–56, 2012 (in Japanese).

[7] C. A. Coello Coello, M. S. Lechuga,"MOPSO: a proposal for multiple objective particle swarm optimization" ,*Proc. of IEEE CEC 2002*, 1051–1056, 2002.

[8] M. R. Sierra, C. A. Coello Coello, "Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ϵ-Dominance", *Proc. of EMO 2005*, 505–519, 2005.

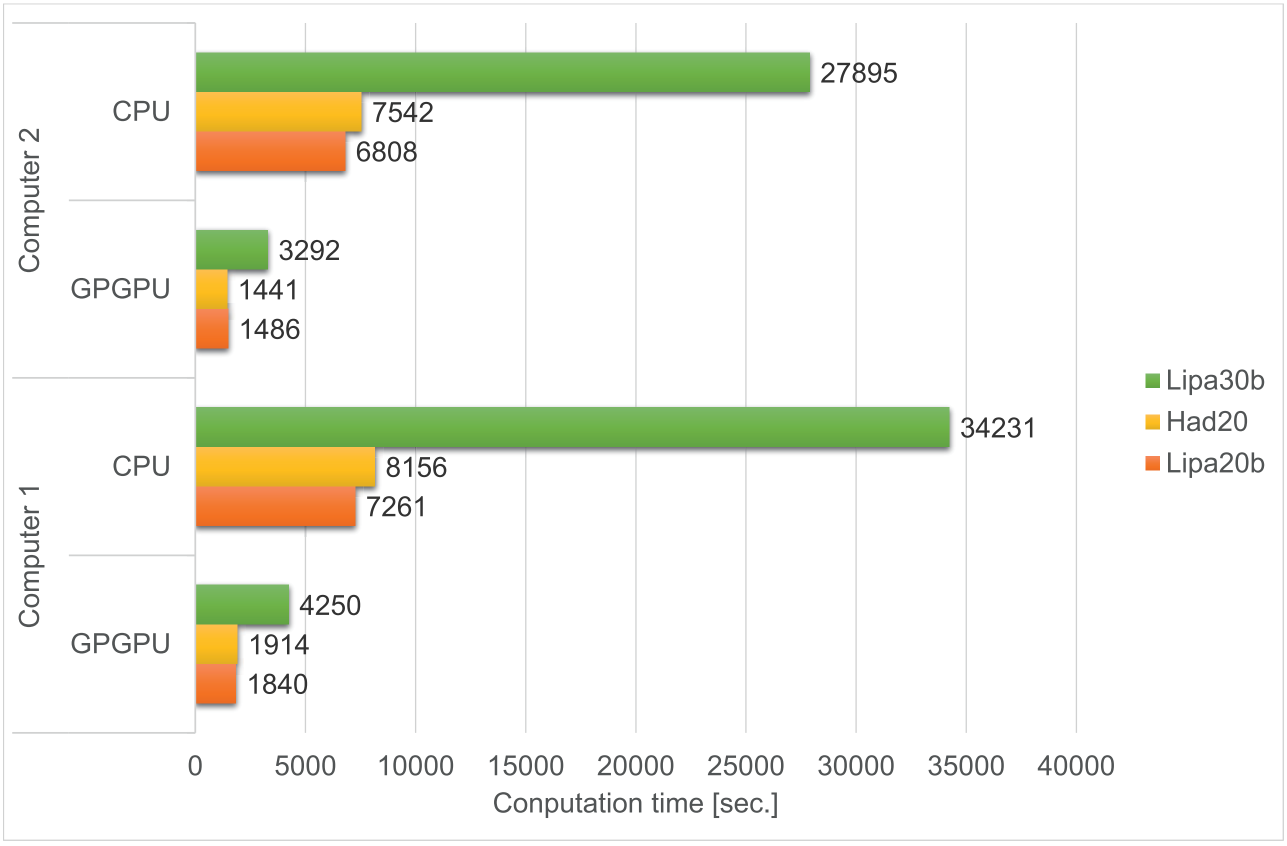

Figure 3: Result of computation time.

Figure 4: Result of computation time.

Computer		Lipa20b	Had20	Lipa30b
Computer 1	GPGPU	1840 [sec.]	1914 [sec.]	4250 [sec.]
	CPU	7261 [sec.]	8156 [sec.]	34231 [sec.]
	Speedup	3.95	4.26	8.05
Computer 2	GPGPU	1486 [sec.]	1441 [sec.]	3292 [sec.]
	CPU	6808 [sec.]	7542 [sec.]	27895 [sec.]
	Speedup	4.58	5.23	8.47

[9] A. J. Nebro, J. J. Durillo, J. García-Nieto, “SMPSO: A new PSO-based metaheuristic for multi-objective optimization”, *Proc. of MCDM 2009*, 66–73, 2009.

[10] T. Kawamura, Y. Horio, M. Hasegawa, “Mutual Information Analyses of Neuron Selection Techniques in Synchronous Exponential Chaotic Tabu Search for Quadratic Assignment Problems”, IEEJ Trans. on Electronics, Information and Systems, vol. 131, no. 3, pp. 592–599, 2011 (in Japanese).

[11] R. E. Burkard, S. E. Karisch, F. Rendl,“QAPLIB – A quadric assignment problem library”, http://www.opt.math.tu-graz.ac.at/qaplib/

[12] T. Tachibana, M. Adachi, “Accelerating Computation of Combinatorial Optimization Problems Using GPGPU”, vol. NLP111, no. 106, pp. 53-58, 2011.