

A Parallel Version of Tree-Seed Algorithm (TSA) within CUDA Platform*

Cinar A.C.^a, Kiran M.S.^{b1}

^aTurkish State Meteorological Service, Konya Division, 42090 Konya – Turkey (cevahircinar@gmail.com)

^bSelçuk University, Faculty of Engineering, Department of Computer Engineering, 42075 Konya- Turkey (mskiran@selcuk.edu.tr)

ABSTRACT:

Recent years, the general purpose computing on graphical processing unit (GPGPU) has gained a huge popularity. The usage of GPGPU becomes widespread due to the fact that the production technology of central processing unit (CPU) reaches the physical limit and big data. In processing of big data, the parallel programming approach is an alternative way to overcome time complexity. In this study, tree-seed algorithm (TSA), one of the population-based algorithms and inspired by the relations between trees and seeds, is implemented within CUDA by using GPU. To measure the processing time of TSA, serial and parallel version of TSA tested on well-known numerical benchmark functions. In the experiments, different sizes of population are considered, the dimensionality of the problems is fixed as 10 and the methods are run 30 times under these conditions. The experimental results show that parallel TSA is accelerated up to about 150 times with respect to serial version of TSA.

KEY WORDS: Parallel Programming, Tree-Seed Algorithm, Cuda, Gpu

* This paper has been produced from the MSc Thesis of Ahmet Cevahir Çınar.

¹ Corresponding Author

E-mail: mskiran@selcuk.edu.tr

Tel: +90 332 223 19 92

Fax: +90 332 241 06 35

1. INTRODUCTION

To overcome requirements of the computation or calculation operations, studies and efforts in the fields of hardware and software are continued. In recent years, processing of exponentially increasing the data (big data) requires new computation technologies. One of the new technologies to process the data is CUDA² (computer unified device architecture) provided by NVIDIA³. The general purpose computing on graphical processing unit (GPGPU) within CUDA is an alternative approach to process big data or to solve time consuming problems. In this paper, tree-seed algorithm to solve continuous optimization problems by inspiring relations between trees and seeds is implemented within CUDA in parallel. Despite the fact that there are much more parallel application development platforms, CUDA is a preferred due to functional library supports, easy writing of codes, an ever-growing platform provided by NVIDIA, redundancy of accessible resources for CUDA. TSA is implemented in parallel, because it presents an appropriate algorithmic framework for parallelization. TSA is firstly proposed in (Kiran, 2015) to solve unconstrained continuous benchmark problems and multi-level thresholding problem. The performance of TSA has been compared with swarm intelligence methods in the first study of TSA. TSA is also used to solve a constrained optimization problem (pressure vessel design-PVD). To overcome constraints of the problem, a penalty function is proposed for TSA. The performance of TSA has been compared with particle swarm optimization (PSO) and artificial bee colony (ABC) algorithm (Kiran, 2016). A parallel version of TSA algorithm for continuous optimization is proposed in (Çınar, 2016). Within CUDA, the other nature-inspired algorithms are implemented: parallel version of genetic algorithm and neural network (Bukharov & Bogolyubov, 2015), PSO for task matching (Solomon, Thulasiraman, & Thulasiram, 2011), image processing (Kneusel, 2014), motion estimation and motion capture (Zhang & Seah, 2012), document classification (Platos, Snašel, Jezowicz, Kromer, & Abraham, 2012), ABC algorithm for classification (Janousešek, Gajdoš, Radecký, & Snašel, 2014). Ant colony algorithm for scientific work-flow problem (Wang, Li, & Zhang, 2015), genetic algorithm for graph coloring (Kai, Ming, Lin, & Xiaoming, 2015).

The rest of paper is organized as follows: Section 2 describes TSA and the parallelization process of TSA is given Section 3. In section 4, the test problems, experimental conditions and results are presented. Results are discussed in Section 5 and the study is finally concluded and a future direction is drawn in Section 6.

2. TREE-SEED ALGORITHM - TSA

TSA is proposed by inspiring the relations between trees and their seeds and it is a population-based optimization algorithm. Trees and seeds in the TSA represent the possible solution for the optimization problem and the tree population is called as stand. In the initialization of TSA, a certain number of trees is generated. These solutions are evaluated to obtain the fitness of the solutions by using the objective function specific for the optimization problems. For each tree, a number of seeds are created by using interactions of the trees in the stand. After generation of seeds, the fitness of the seeds is calculated by using the objective function. If the solution with the best fitness of the seeds of the tree is better than the fitness of the current

tree, this tree is removed from the stand and the new seed (solution) is placed to the stand. The TSA works until a pre-determined termination condition is met. In the basic study of TSA, the maximum number of function evaluations is used as termination condition for the algorithm.

There are two important points in TSA: first is seed production and second one is the control parameter of TSA named as search tendency-ST. For seed production, Eq. 1 or Eq. 2 is used.

$$S_{k,j} = T_{i,j} + \alpha_{i,j} \times (B_j - T_{r,j}) \quad (1)$$

$$S_{k,j} = T_{i,j} + \alpha_{i,j} \times (T_{i,j} - T_{r,j}) \quad (2)$$

where, $T_{i,j}$ is the j th dimension of i th tree, $S_{k,j}$ is the j th dimension of k th seed for $T_{i,j}$, B_j is the j th dimension of the best solution obtained so far, $T_{r,j}$ is the j th dimension of randomly selected tree from the stand, $\alpha_{i,j}$ is the scaling factor in range of $[-1,1]$. Which equation is used for production the seed is determined by ST parameter. ST is a pre-determined value in range of $[0,1]$ and a random number is generated in range of $[0,1]$. If this random number is less than ST parameter, Eq.1 is used, otherwise Eq.2 is used for seed production for the tree. Briefly, the search capabilities of TSA are controlled by seed production and ST parameter. After these explanations, the algorithmic framework and working diagram of TSA is given in Figure 1 and Figure 2, respectively.

Initialization

1. Determine the number of trees in the stand.
2. Determine the ST parameter.
3. Generate the stand in solution space.
4. Calculate fitness of the trees in the stand by using objective function.

Seed Production

5. For each tree
 - Generate the seeds by using ST parameter and update equations.
 - Determine the fitness of seeds by using objective function.
 - If the seed with best fitness is better than current tree, remove the tree and place the best seed to the stand.

Termination

6. Select the best tree from the stand.
6. If the termination condition is met, report the best solution. Otherwise, go to Seed Production.

Figure 1. The algorithmic framework of TSA

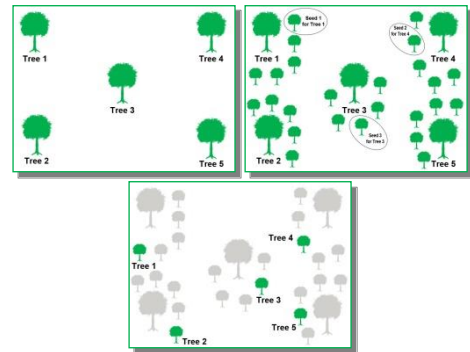


Figure 2. The working diagram of TSA

As seen from the algorithmic framework and working diagram of TSA, a simple and usable method is proposed in (Kiran, 2015). In the seed production phase, how many seeds are created for a tree is important point. The inventor of TSA (Kiran, 2015) proposed a range (between %10 and %25 of the

² CUDA is a trademark of NVIDIA Corp.

³ NVIDIA is a trademark of NVIDIA Corp.

stand) for production number of seeds. The parallel version of the algorithm, we used the number of seeds for each tree as 20% of the stand in the experiments.

3. PARALLELIZATION OF TSA

For parallelization of TSA, two different phases those are data parallelism and task parallelism are considered. In the parallel version of TSA, for data parallelism random numbers which required by the TSA sent to GPU memory every iteration at the same time. It showed at Fig 3. CURAND function library is not used for every thread so we gain extra time because calling library more expensive in terms of time.

```
r_r = rand(N, NS*D, 'double', 'gpuArray');
r_alfa = -1 + (2) * rand(N, NS*D, 'double', 'gpuArray');
r_k = randperm(N);
r_komsu = gpuArray(r_k);
```

Figure 3. Random numbers which required by the TSA sent to GPU memory every iteration at the same time

In serial version of TSA, Trees and Seeds storage row-major order in memory as shown at Figure 4. In parallel version of TSA, 1 Tree and 4 Seeds (for dimension=5) of this tree storage one after another row-major order in memory as shown at Figure 5. Trees global best positions and same fitness storage for serial and parallel version of TSA are shown in Figure 6.

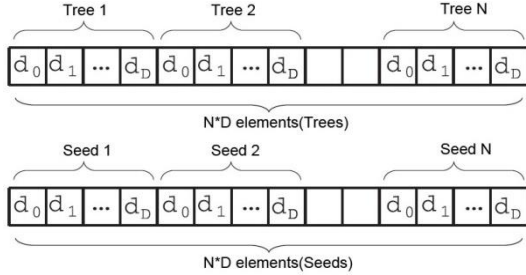


Figure 4. Trees and Seeds storage as row-major order for serial version of TSA.

Tree 1					Seed 1 for Tree 1					Seed 2 for Tree 1					Seed 3 for Tree 1					Seed 4 for Tree 1				
d0	d1	d2	d3	d4	d0	d1	d2	d3	d4	d0	d1	d2	d3	d4	d0	d1	d2	d3	d4	d0	d1	d2	d3	d4

Figure 5. Tree and Seeds location as row-major order in memory for parallel version of TSA.

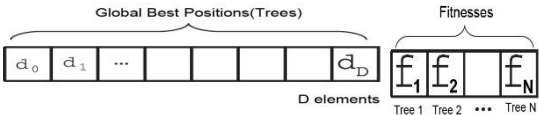


Fig 6. Global best position of stand and same fitness storage for serial and parallel versions of TSA

In the parallel version of TSA, every benchmark function recoded separately for trees and seeds for task parallelism. For example, there is one part in serial version of Sphere function coded at the programming platform and there are two parts in parallel version of Sphere function coded at the programming platform. First part is for trees, second part is for seeds. During the iterations, computations are performed simultaneously.

4. EXPERIMENTAL MATERIALS, CONDITIONS AND RESULTS

To analyze performances of the serial and parallel versions of TSA, the numerical benchmark functions are used. Well-known

5 functions widely used in literature are considered and these are given in Table 1 which is in appendix.

The dimensionality of all functions is fixed as 10. Therefore, the solution space for using initialization of the stand is 10-dimensional space and for the initialization of the trees, Eq.3. is used.

$$T_{i,j} = L_{j,\min} + r_{i,j}(H_{j,\max} - L_{j,\min}) \quad (3)$$

where, $T_{i,j}$ is the j th dimension of i th tree, $L_{j,\min}$ is the lower bound of the j th dimension, $H_{j,\max}$ is the upper bound of j th dimension and $r_{i,j}$ is a random number produced in range of [0,1] for each tree and dimension.

The control parameter of the serial and parallel version of the TSA (ST parameter) is selected as 0.1 according to the first study of TSA (Kiran, 2015). The population size is taken as 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100. While population size is changed, the termination condition is not changed. The termination condition in the experiments is maximum number of function evaluations (Max_FEs). Max_FEs is taken as $1.5E+5$. The number of the seeds which will be produced for each tree is another important point. The number of seeds is selected as 20% of the stand size and it is properly adjusted according to the number of trees in the stand. For instance, if the size of stand is 20, two seeds are produced for each tree at the seed production phase.

Based on the aforementioned conditions, the serial and parallel versions of TSA are run 30 times on a machine that the configuration of this machine is given in Table 2. In terms of working time, the methods are compared and the comparative results are given in Table 3. The values in the Table 3 shows how many times the parallel version of TSA is faster than the serial version of TSA. Due to space limitations, all the results tables are given in Appendix.

In the comparison table, when the stand size is taken as 10, parallel version of TSA is two times faster than the serial version of TSA on Sphere function.

5. RESULTS AND DISCUSSION

When the results are analyzed, it is seen that different results are obtained. Basically, there is no complex calculations in the function such as Sphere and less number of trees in the stand, the acceleration of TSA implemented within CUDA look worse because the number of calculations which will be performed in parallel is limited. If there are a huge number of complex calculations and the number of trees in the stand is high, the performance of parallel TSA is clearly indicated. If the implementations of serial and parallel versions of TSA are performed on the different machines and GPUs, different results can be obtained but it is expected that the results are the proportionally similar.

6. CONCLUSION AND FUTURE WORKS

In this study, a parallel version of TSA is presented and serial and parallel versions of TSA are applied for solving the numerical benchmark functions. Obtained results are compared with each other in terms of working time. The worst acceleration of working time of TSA in parallel is about two times and the best acceleration is about 150 times. Therefore, the parallel TSA is faster than the serial version of TSA about 150 times in best case. In the near future, we plan to apply the parallel version of TSA to solve different optimization problem such as constraint optimization problems, binary optimization problems.

ACKNOWLEDGEMENTS

The authors wish to thank the Coordinatorship of Scientific Research Projects (CSRP) at Selcuk University for institutional supports. In addition, the parallel TSA is run on the hardware that purchased with a research project by support of CSRP. The research project number is BAP-15401121.

REFERENCES

- Bukharov, O. E., & Bogolyubov, D. P. (2015). Development of a decision support system based on neural networks and a genetic algorithm. *Expert Systems with Applications*, 42(15), 6177-6183.
- Çınar, A. C. (2016). *A Cuda-based Parallel Programming Approach to Tree-Seed Algorithm*. (MSc Thesis), MSc Thesis, Graduate School of Natural Sciences, Selcuk University.
- Janousešek, J., Gajdoš, P., Radecký, M., & Snašel, V. (2014). *Classification via Nearest Prototype Classifier Utilizing Artificial Bee Colony on CUDA*. Paper presented at the International Joint Conference SOCO'14-CISIS'14-ICEUTE'14.
- Kai, Z., Ming, Q., Lin, L., & Xiaoming, L. (2015). Solving Graph Coloring Problem by Parallel Genetic Algorithm Using Compute Unified Device Architecture. *Journal of Computational and Theoretical Nanoscience*, 12(7), 1201-1205.
- Kiran, M. S. (2015). TSA: Tree-seed algorithm for continuous optimization. *Expert Systems with Applications*, 42(19), 6686-6698.
- Kiran, M. S. (2016). An Implementation of Tree-Seed Algorithm (TSA) for Constrained Optimization *Intelligent and Evolutionary Systems* (pp. 189-197): Springer.
- Kneusel, R. (2014). Curve-Fitting on Graphics Processors Using Particle Swarm Optimization. *International Journal of Computational Intelligence Systems*, 7(2), 213-224.
- Platos, J., Snašel, V., Jezowicz, T., Kromer, P., & Abraham, A. (2012). *A PSO-based document classification algorithm accelerated by the CUDA platform*. Paper presented at the Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on.
- Solomon, S., Thulasiraman, P., & Thulasiram, R. (2011). *Collaborative multi-swarm PSO for task matching using graphics processing units*. Paper presented at the Proceedings of the 13th annual conference on Genetic and evolutionary computation.
- Wang, P., Li, H., & Zhang, B. (2015). A GPU-based Parallel Ant Colony Algorithm for Scientific Workflow Scheduling. *International Journal of Grid and Distributed Computing*, 8(4), 37-46.
- Zhang, Z., & Seah, H. S. (2012). *CUDA acceleration of 3D dynamic scene reconstruction and 3D motion estimation for motion capture*. Paper presented at the Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference on.

APPENDIX

Table 1. The numerical benchmark functions used in the experiments

Name	Range	Formulae
Sphere	$-100 < x_i < 100$	$f(x) = \sum_{i=1}^d x_i^2$
Rosenbrock	$-2.048 \leq x_i \leq 2.048$	$f(x) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 - (x_i - 1)^2]$
Rastrigin	$-5.12 \leq x_i \leq 5.12$	$f(x) = 10d + \sum_{i=1}^d x_i^2 - 10\cos(2\pi x_i)$
Griewank	$-600 \leq x_i \leq 600$	$f(x) = \sum_{i=1}^d \frac{x_i^2}{4000} - \prod_{i=1}^d \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
Ackley	$-32.768 \leq x_i \leq 32.768$	$f(x) = -a \exp\left(-b \sqrt{\frac{1}{d} \sum_{i=1}^d x_i^2}\right) - \exp\left(\frac{1}{d} \sum_{i=1}^d \cos(cx_i)\right) + a + \exp(1)$

Table 2. The machine configuration

Device	CPU	GPU
Processor	Intel® Xeon® Processor E5-2670	GeForce GTX TITAN X
Number of cores	8 physical 16 thread	3072 CUDA Cores
Clocks	2.6 GHz (max 3.3 GHz)	1000 MHz(max 1075 MHz)
Memory	DDR3 800/1066/1333/1600	GDDR5
Memory size	32 GB	12 GB
Operating System	Windows 10	-
Compute Capability	-	5.2
CUDA version	-	7.5

Table 3. The comparison of serial and parallel versions of TSA in terms of working time

Function / Stand Size	10	20	30	40	50	60	70	80	90	100
Sphere	2.38	6.85	10.33	13.17	15.92	19.19	23.78	27.21	30.86	34.39
Ackley	3.14	11.01	22.66	35.36	50.09	62.84	80.35	94.94	109.25	123.16
Rastrigin	3.14	11.42	23.84	38.96	56.87	74.07	89.14	110.23	126.48	148.02
Griewank	2.81	11.45	24.22	39.21	58.36	77.58	93.48	112.70	138.62	149.47
Rosenbrock	3.15	11.55	24.52	40.77	59.91	78.24	93.91	112.49	131.78	149.59