IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-6, June 2019 E-ISSN: 2347-2693

Finding the best network for laying renewable energy based solar panel roads: A GPU parallel algorithm implemented on CUDA

Aayush Kapur[1*], Nirut Gupta[2]

[1,2]B Tech CSE, VIT Vellore, India

[*]*Corresponding Author aayush.kapur2016@vitstudent.ac.in*

DOI: https://doi.org/10.26438/ijcse/v7i6.255260 | Available online at: www.ijcseonline.org

Accepted: 13/Jun/2019, Published: 30/Jun/2019

Abstract— The objective of this paper is implementation of ACO algorithm on GPU to combat real life problems of road network identification along with an application focusing on renewable energy. GPUs are specialized microprocessors that accelerates graphics operation. Parallel processing is required when we consider a heavy code with so much of similar iterations. CUDA is NVIDIA's architecture for parallel computing that is used for extensive parallel computing and increases the performance by employing the GPU (Graphical Processing Unit). We have Ant colony optimisation algorithm implementation that is a bit different than others. Also, we compare it with the sequential code and the results are that it is very fast as compared to sequential code. To deal with the execution of optimisation we will propose two different approaches, one will be the serial approach of the ACO algorithm to generate the network and other will be GPU / CUDA based approach. We will compare the execution time in both the cases and then find out the speed up. An applicability of this approach is for generating the best possible road network for city coordinates where we try to get the network with least cost. This is of immense applicability for developing countries where road networks are upcoming.

Keywords— CUDA, GPU, Parallel Processing, travelling salesman problem, Road network identification

I. INTRODUCTION

With growing population and ever-increasing energy demands it is very important that we start generating enough energy for our needs without dependency on the fossil fuels to preserve our future, thus there is a lot of scope for the renewable energy. Large expanse of land available for China and India along with abundance of sunlight throughout the year in the subcontinent, there is opportunity for developing a road network for all the cities and small towns especially in India, with these roads being of special solar panels which have been tested successfully on one mile stretch in China. Thus these roads can be a source of electricity for us.

The scope of the paper is limited to the computational and algorithmic aspects of this idea. For finding the best route possible between the cities as explained above for the road network of solar panels which will allow the transportation over it, we need to find the route joining all the points of interest. Thus, the problem can be reduced to that of a basic case of travelling sales man problem. Now there can be various approaches to solve the travelling salesman problem, one such approach is Ant colony optimisation.
Using this optimisation, we will be able to find the network we desire.

To deal with the execution of the optimisation we will propose two different approaches, one will be the serial approach of the ACO algorithm to generate the network and other will be GPU / CUDA based approach. We will compare the execution time in both the cases and then find out the speed up for both the cases.

Section II of this paper gives details about the related work which has been done on this topic, Section III explains the methodology we adopted and the algorithm used, Section IV mentions the results and the sections V gives the conclusion and also the future work which can be done on this topic.

II. RELATED WORK

AN EFFICIENT GPU IMPLEMENTATION OF ANT COLONY OPTIMISATION FOR THE TRAVELLING SALESMAN PEROBLEM

This paper talks about the sophisticated and clean ACO approached on GPU. They have presented an efficient method for selecting random cities by number of ants. The method uses iterative random trial which is useful to find next city in highly efficient and with low computation cost. They have considered many programming issues of the architecture of GPU such as shared memory conflicts. Also, they have introduced a new method with stochastic trial in the roulette-wheel selection.

Table 1: Papers on ACO and CUDA

Title	Author	Journal and DATE	Key concepts	Advantages	Disadvantages	Future enhancement
Improving Ant Colony Optimization performance on the GPU using CUDA	Lawrence Dawson Iain Stewart	IEEE congress 2013	Implementation of tour construction using roulette wheel selection. Proposed a new parallel implementation of roulette wheel which DS roulette which reduces running time of tour construction	New parallel implementation executed upto 82x faster while not changing the quality of tour constructed. And almost 8.5 times more than GPU existing parallel implementation	Roulette wheel selection method cannot be used on minimization problems.	A more compact and enhanced version of roulette.
The GPU-based Parallel Ant Colony System	Rafal Skinderowicz	2016	Proposal of three parallel versions of Ant colony system (which is similar to ACO and MMAS). The two of them uses standard pheromone memory and the third one uses selective approach.	Parallel ACS on Nvidia Kepler GK104 is able to obtain speed up of 25x vs sequential ACS while in case of selective pheromone it comes about to be 17x.	Selective pheromone is costly and complex.	Algo should be tested by using new generation of GPU's.
Accelerating ant colony optimization-based edge detection on the GPU using CUDA	Laurence Dawson, Iain A Stewart	IEEE congress 2014	Implements a novel data parallel approach that maps individual ants to thread wraps. GPU is used to reduce number of iterations	First parallel implementation of an ACO based edge detection.	Result in some drawbacks like broken edges	Parallel implementation of other such thing based on ACO
Novel Method to Improve ACO Performance on the GPU Using CUDA for Nurse Roster Scheduling Problem	Mr. A. P. Pande1,Mr. B. S. Patil, Mr. A.U. Patil	International Journal of Innovative Research in Computer and Communication Engineering March 2016	Implementation of data-parallel GPU execution of the ACO algorithm to solve nurse roster scheduling problem.	parallel accomplishment executes up to 8-12x faster than sequential execution at the same time as preserving the quality of the Schedules formation.	Implemented both the construction of Schedule and pheromone update phases on the GPU	Improvement in SS required since Schedule construction phase uses a new efficient execution of roulette wheel selection.

A REVIEW ON SOLAR ROADWAYS: THE FUTURE OF ROADS	Stephy Johny and Keerthi Susan John	International Journal of Recent Innovation in Engineering and Research Volume: 02 Issue: 03 March– 2017 (IJRIER)	idea is to replace the asphalt roads with solar roadways on our streets, highways parking lots and sidewalks that collect solar energy to be used by our homes and businesses.	The renewable energy generated by solar road panels will replace the current need for fossil fuel which is used for generation of electricity which in turn can reduce the greenhouse gases nearly to half.	As it requires huge initial investment, it would be difficult to install solar roadways in developing countries.	Solar roadways will solve the problems of usage of fossil fuels and energy consumption. The future work involves making it possible in real life.

COMPUTE UNIFIED DEVICE ARCHITECTURE (CUDA)

CUDA is NVIDIA's architecture for parallel computing that is used for extensive parallel computing and increases the performance by employing the GPU (Graphical Processing Unit). CUDA is identifying its use in various branches which includes image and video processing, simulation of fluid dynamics, seismic analysis, ray tracing and many more.

CUDA programming has a hierarchy of thread groups called block, grid and thread. A grid is divided into some number of blocks and in each block, there consist an equal number of threads.

CUDA C extends C language by allowing the developer to declare C function. They are called as kernels. When kernel is involved all blocks, which are there in the grid are allocated to the running processor and threads in each block is executed by the cores present in the running processor.

III. METHODOLOGY

Steps:

1. Initialise the ants and cities, get the coordinates of the cities.
2. Initialise required matrices
3. Random assignment of cities to ants

For (pheromone update < max iterations)
{
4. Tour construction for each ant
5. While (tour construction == TRUE)
{
Calculate order of cities visited
Calculate the distance of tour
}
6. Minimum distance travelled by an ant taken
7. Update the delta matrix taking in to account the optimisation for each ant
8. More ants on route, higher delta value, higher pheromone value
9. Update pheromone definite times
}

Steps explanation:

1. Initialise the ants and cities, get the coordinates of the cities.
2. Initialise the distance matrix as distance between the cities, pheromone matrix as a highly negative value, delta matrix as zero, visited array and tour array for each ant as null/zero/empty.
3. Randomly assign a city to each ant as starting point, tour array will have that city as first element as well as mark that city as visited in the visited array
4. Now construct tour for each ant from the starting city selected above, each city to be visited next is selected randomly, no city can be visited more than once.
5. While tour construction (true)
for all cities
for each ant
5.1 make next city as visited
5.2 save the order in which cities are visited in tour array
5.3 calculate the total distance traversed in the tour till that point
5.4 make next city as current city
6. Identify the Ant which traversed minimum distance while covering all the cities and reached back to the starting city.
7. Delta array will be updated by the rule
"for all ants
for two consecutive cities in the tour
Add
(CONSTANT/distance between consecutive cities in the tour) to the array elements in the delta matrix depicting the consecutive cities in the tour of a particular ant"

8. This will make sure that if more ants take similar route to cities, higher the value of delta.

9. Pheromone will be updated definite number of times with each successive update affecting the pheromone value to a lesser extent.

10. For particular city pair

Pheromone update =pheromone current + delta value

11. Initialise the tour and visited array for each ant as empty, to account for successive iterations.

12. Use the maximum distance traversed in the tour and the pheromone matrix in the successive iterations.

IV. RESULTS AND DISCUSSION

Through this work we have tried to implement Ant colony optimisation in the sequential mode and in the parallel mode using CUDA. Then we applied this implementation on the travelling salesman problem to find the best route, we have taken the 'tsp' file of the coordinates of cities and applied it to the above scenario to obtain the best network for those points. The said points are the cities in developing country such as India and the said network is the road network which we want to develop.

Initially the algorithm was run in sequential fashion on different data points to get the results, however the time taken for the cities increases exponentially even though the increase in the number of cities is low.

As can be seen from the table above the time taken for 438 cities is 160 seconds, for 1002 cities its higher and it goes on increasing.

This results in a prohibitive picture for execution for dataset as large as 70009 cities which we want to consider. The time taken will be very large. Thus, we tried to implement the same on the parallel CUDA, where we found that the time taken was very much lower compared to the sequential execution.

In the parallel implementation first, we have equal number of ants and cities, the number of cities increased as 29, 48, 100, 200, 318, 438, 1000, 4000

The execution cities and parameters were kept the same as for sequential execution. The speedup results were found and the comparative graph has been shown above. In the case of 70009 cities, since we had no serial data due to prohibitive time requirements, we tried to test it on different number of ants considering different number of cities each time.

V. CONCLUSION AND FUTURE SCOPE

In this paper we have proposed an implementation of a parallel algorithm for finding best network for laying energy based solar panel roads. In our implementation we randomly assign city to each ant and start the tour and find the best tour possible by multiple iterations. This heavy task with so much iterations needs to be done in parallel for which CUDA is the best solution. We have successfully implemented the algorithm and find good results.

For 200 cities the parallel execution time was found to be 1.7389 seconds whereas serial execution time for the same was 36.172. Thus, GPU implementation attains the speedup factor of 20.80

For our future work, we are mainly going to focus on the accuracy and efficiency of the algorithm and will find some ways to optimise the solution.

Second it will be of some practical implication of the results we acquired more aligned with the real life scenarios with more parameters considered.

It can also be applied on different applications which we haven't considered in this paper requiring heavy computational tasks.

ACKNOWLEDGMENT

We would also like to thank VIT University management for giving us this opportunity to undergo this subject and providing us facilities to do this project.

REFERENCES

[1] Akihiro Uchida, Yasuaki Ito, Koji Nakano, "An Efficient GPU Implementation of Ant Colony Optimization for the Traveling Salesman Problem", IEEE
[2] Zhou Y, He F Z, Qiu Y M. June 2017, Vol. 60 068102:1–068102:3. SCIENCE CHINA Information Sciences
[3] Dawson, Stewart. 2013. IEEE
[4] Zhoua, Hea, Houa, Qiub . 14 october 2017. ELSEVIER , future generation computer systems
[5] Akihiro, Ito, Nakano. 2012. IEEE
[6] Ceciliaa , Llanesa, Abellána, Gómez-Lunab, Changc, Hwud .15December2017. ELSEVIER, Journal of parallel Distributed computing
[7] Dawson, Stewart. 2014. IEEE
[8] Johny and John. 25 May 2018. ELSEVIER Computer Languages,Systems & Structures
[9] Ermiş , Çatay. May 2017. Transportation Research Procedia
[10] Souza, Pozo. 2014. IEEE
[11] Patil, Pandel. March 2016. International Journal of Innovative Research in Computer and Communication Engineering
[12] Skinderowicz. 2016. ACM
[13] Papenhausen, Mueller. 25 May 2018. ELSEVIER Computer Languages,Systems & Structures
[14] Khatri, Gupta. 2014. IEEE
[15] Johny, John. March– 2017. International Journal of Recent Innovation in Engineering and Research
[16] Alimi, Bali, Elloumi, Abraham. 2017. Springer
[17] Kulkarni. May-Jun 2013. International Journal of Engineering Research and Applications
[18] Michaël Krajeck, Gravel, Delévacqa.2012.IEEE
[19] Shingate. 16 November 2017

[20] SHI, Zhi. 2012. Springer
[21] Rocki, Suda. 2013. IEEE 27th International Symposium on Parallel & Distributed Processing
[22] Zhao, Cai, Lan. 2012. International Conference
[23] Youness, Ibraheim, Moness, Osama. 2012. IEEE
[24] Diaz, caro. 2012. IEEE
[25] Jain, Vanita & Jain, Aarushi & Jain, Achin & Kumar Dubey, Arun. (2018) Comparative Study between FA, ACO, and PSO, ijsrcse

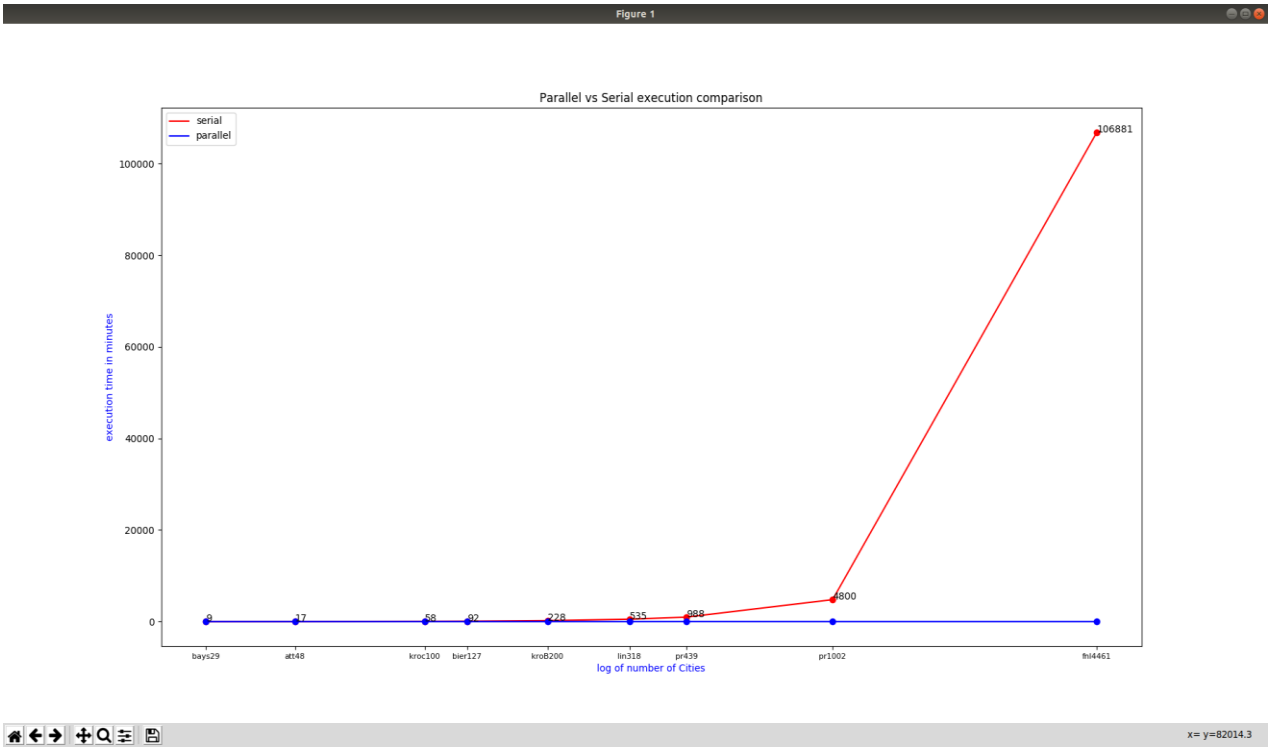

Fig 1. NVIDIA visual profile for the code source: run on dell system nvidia GTX960M

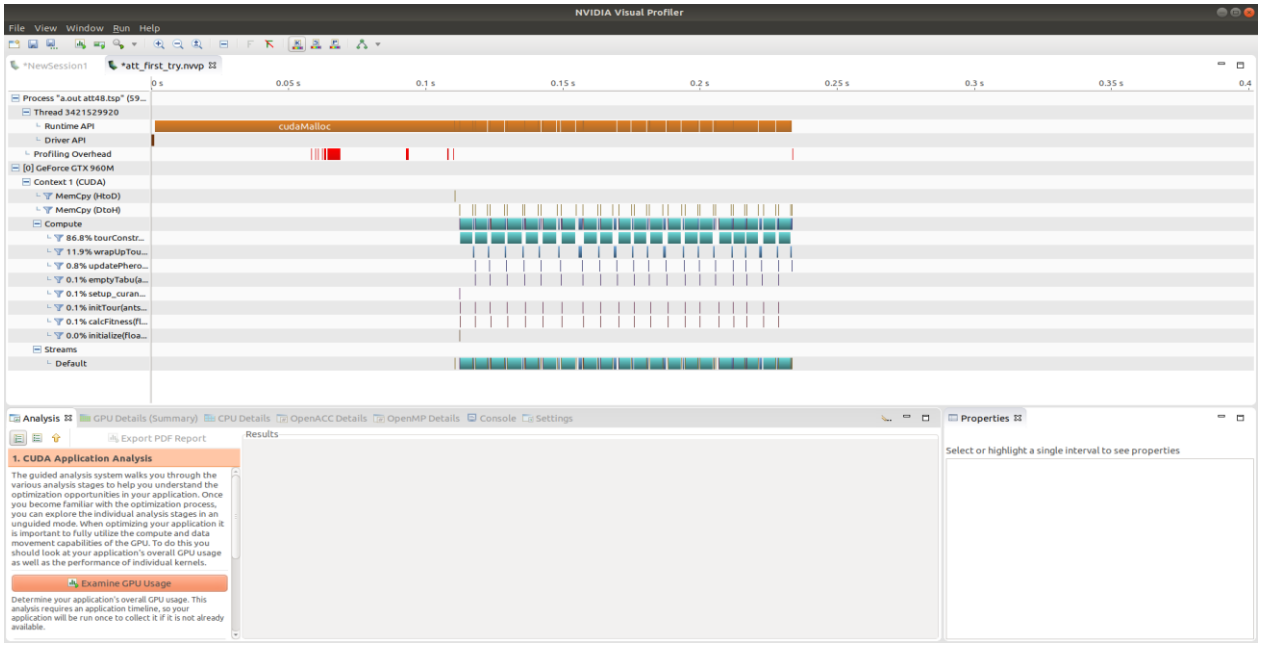

Fig 2 Parallel vs Serial Execution Comparison Source: run on python2.7 plotted using matplotlib.pyplot on dell system nvidia GTX960M

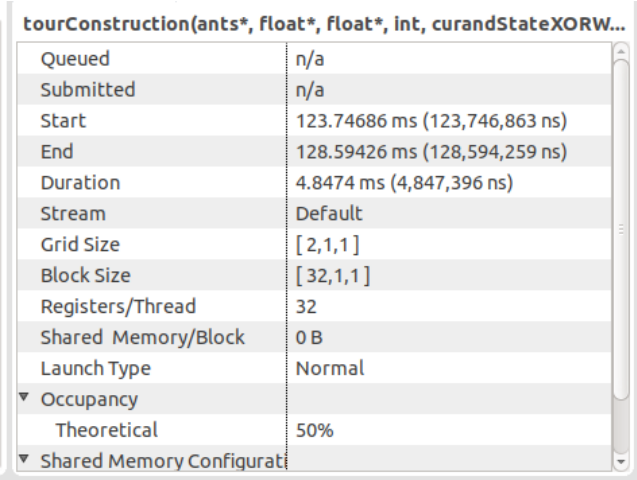

Fig 3. Tour construction measurements from NVIDIA visual profiler source: run on our machine

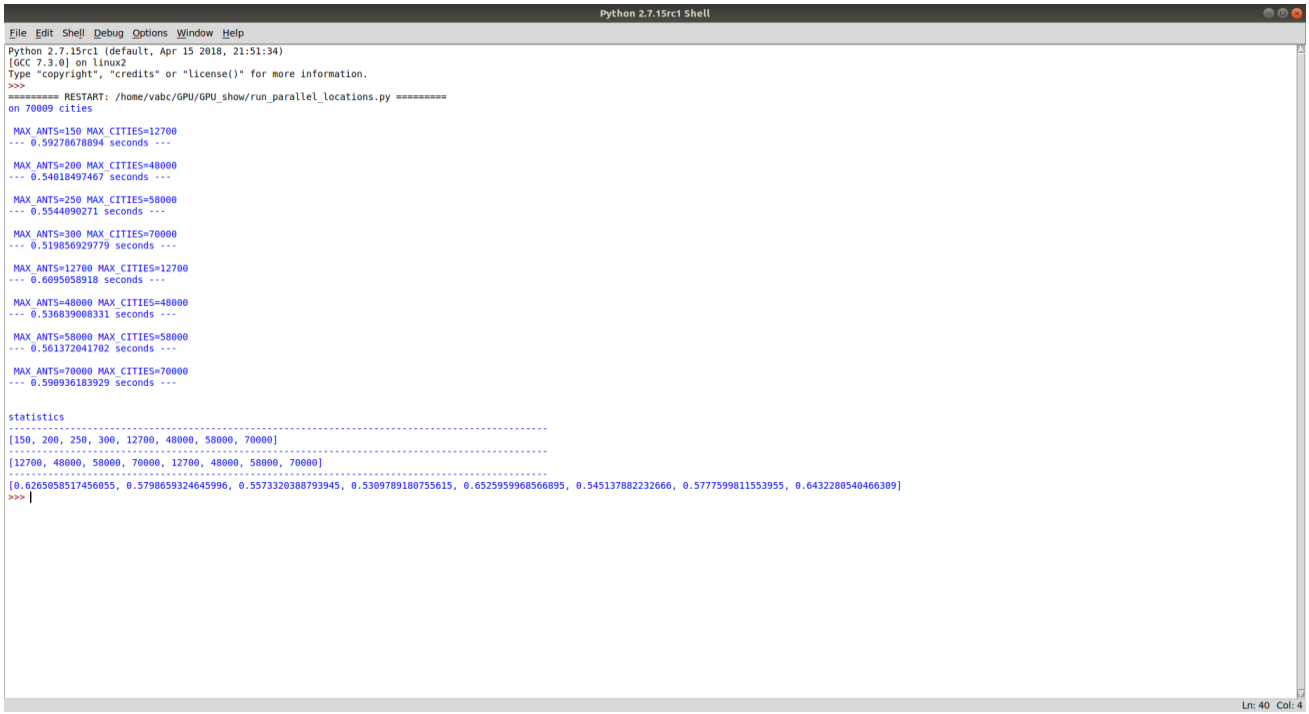

Fig 4. Parallel execution over 70009 cities: time output source: run on python2.7 plotted using matplotlib.pyplot on dell system nvidia GTX960M

Table 2. serial vs parallel time comparison Source: on dell system nvidia GTX960M

Cities	Serial	Parallel
29	0.812s	0.13s
48	2.11s	2.03s
100	8.987s	0.48s
127	13.63s	0.74s
200	36.17s	1.73s
318	92.91s	3.65s
438	160.94s	11.04s
1002	-	0.19s
4461	-	0.07s

