www.ijsrnsc.org

Available online at www.ijsrnsc.org

IJSRNSC

Volume-5, Issue-3, June 2017
Research Paper

Int. J. Sc. Res. in Network Security and Communication

ISSN: 2321-3256

Optimizes NP Problem with Integration of GPU Based Parallel Computing

Santosh Kumar[1*], S.S. Dhable[2], Amol D. Potgantwar[3]

[1*]Dept. Computer Engineering, Sandip Institute of Technology and Research Centre, SPPU, Nashik, India
[2] Dept. Computer Engineering, Sandip Institute of Technology and Research Centre, SPPU, Nashik, India
[3] Dept. Computer Engineering, Sandip Institute of Technology and Research Centre, SPPU, Nashik, India

[*]*Corresponding Author: santosh.kumar@sitrc.org, Tel.: 8554016999*

Received 12th May 2017, Revised 28th May 2017, Accepted 14th Jun 2017, Online 30th Jun 2017

Abstract— There are different number of optimization problems are present those are NP problems. Graph theories are most commonly studied combinational problems. An NP-hard problem takes exponential time for computation to get best solution. Heuristics algorithms provides great alternative to tackle NP hard problems with optimized solution in a limited time. GPU architecture boosts computation of heuristics algorithms with single program multiple data approach. By applying solution level parallelism to traveling salesman problem with global and shared memory level optimizations. Inside this paper given that the innovative move towards to resolve these combinational troubles through GPU based parallel computing via CUDA architecture. Evaluating those problem with significant to the transfer rate, effectual memory utilization and speedup etc. to attain the supreme achievable solution. The algorithm for the optimization problem is in consideration is 2-opt algorithm which is used to grasp the capable memory management, coordinated completing, saving time and growing speedup of execution. The experimental results are compared with the performance of different number of datasets with the execution time as well as the parameter values such as size of blocks. So that the speedup factor is develop in addition to search out the paramount most favorable solution.

Keywords— Graph Theory, CUDA, GPU, Parallel metaheuristic.

I. INTRODUCTION

In the optimization field, both intellectual and industrialized problems are frequently composite and NP-hard. In practice, their modeling is continuously evolving in terms of constraints and objectives. Thereby, a huge quantity of real-life optimization harms in knowledge, business, finances and industries are multifarious and tricky to solve. Their resolution cannot be approved during a precise behavior within a levelheaded amount of time and their resource requirements are ever increasing. To compact with such a matter, the intend of resolution process must be support on the cooperative exploit of sophisticated approaches from combinatorial optimization, comprehensive parallelism and manufacturing methods. During the most recent decades, metaheuristics which are estimated algorithms have been successfully applied to solve optimization problems. [1, 2, 3]

Indeed, this class of methods allows to producing near-optimal solutions in a realistic time. Metaheuristics may unravel occurrences of tribulations that are alleged to subsist rigid in general, by discovering the typically great key search gap of these instances. These algorithms attain this by reducing the valuable range of the search space and by exploring that space efficiently. Yet, even though metaheuristics tolerate to shrink the earthly complication of troubles decision, they remain unsatisfactory to tackle large problems. Experiments using large problems are often stopped without any convergence being reached. Thereby, in designing metaheuristics, there is habitually a trade-off to be found among the dimension of the problem instance and the computational involvedness to investigate it. As a result, merely the employ of parallelism permits to devise latest methods to tackle large problems. [5, 6]

Over the earlier decades, parallel computing has been exposed as an unavoidable way to covenant with large problem instances of difficult optimization problems. The planning and realization of parallel metaheuristics are robustly inclined by the computing platform. Most personal computers integrated with GPUs are typically distant fewer influential than their incorporate counterparts. That is the motive why it would be extremely remarkable to exploit this enormous capacity of computing to implement parallel metaheuristics. Certainly, GPU computing has materialized in the modern living since an important challenge for the

parallel computing research area. This novel rising technology is supposed to be tremendously useful to accelerate countless multifaceted algorithms. One of the foremost issues for metaheuristics is to alter obtainable analogous models and programming paradigms to allow their deployment on GPU accelerators.[8, 9]

In further terminology, the challenge is to resume the parallel models and paradigms to efficiently take into account the characteristics of GPUs. However, the mistreatments of equivalent molds are not inconsequential, and many issues correlated to the GPU memory hierarchical management of this architecture have to be measured. Generally in words, the main issues to deal those are:[1]The distribution of data processing between CPU and GPU, the thread synchronization, the optimization of data shift between the dissimilar memories, the memory capacity constraints etc.

Rest of the paper is ordered as follows. Review of literature is explained in section II. Section III contains system overview, section IV contains problem formulation, section V contains experimental analysis and Conclusion is given in section VI.

II. REVIEW OF LITERATURE

Rafal Skinderowicz proposed, the GPU-based Parallel Ant Colony System. Paper provides the three narrative comparable adaptation of the ACS for the GPUs. Discovering the shortest lane between nest and food source. Communication is done by between ants using pheromone chemical substance. [1]

Laurence Dawson & Iain Stewart proposed, an improving Ant Colony Optimization performance on the GPU using CUDA. This implement mutually the tour assembly and pheromone revise stages of Ant Colony Optimization resting on the GPU by means of the data corresponding loom. This drastically reduces the organization instant of tour construction. [2]

Wojciech Czech & David A. Yuen proposed, Efficient Graph Comparison and Visualization using GPU. This introduced the numerous graph algorithms for comparison and revelation of real world networks. To obtain interactive and robust framework. [3]

Maida Arnautovic, Maida Curic, Emina Dolamic & Novica Nosovic proposed, Parallelization of the Ant Colony Optimization for the Shortest Path Problem using OpenMP and CUDA. That including to find the preeminent vehicle route amid the elected points. Also finding the straight pathway in several oriented graphs. [4]

Tomohiro Okuyama, FumihikoIno, Kenichi Hagihara proposed, A Task Parallel Algorithm for Computing the Costs of All-Pairs Shortest Paths on the CUDA compatible GPU. Involving the speedy routine for adding the outlay of all pairs direct lane on the GPU. [5]

Kamil Rocki & Reiji Suda proposed, High Performance GPU Accelerated Local Optimization in TSP. Paper presents the lofty concert GPU accelerated accomplishment of confined search heuristic algorithm intended for the TSP. [6]

Marco Dorigo & Luca Maria Gambardella proposed the, Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem. In the ACS, a set of working mediator called ants help out to ascertain the superior solutions to TSP's. Ants help using an indirect form of communication umpired by a pheromone they situate on the margins of the TSP grid while building solutions. [7]

Marco Dorigo, Vittorio Maniezzo, & Alberto Colorni proposed, Ant System: Optimization by a Colony of Cooperating Agents. The main distinctiveness of this mold is sanguine response, widen calculation, and the invent exercise of a rational envious heuristic. Sanguine feedback accounts for hurried recognition of high-quality solutions, enlarge computation evade impulsive convergence, and the avaricious heuristic facilitate to find adequate resolutions into the unfortunate period of the search process. [8]

Ugur Cekmez, Mustafa Ozsiginan, & Ozgur Koray Sahingoz proposed, A Uav Path Planning With Parallel Aco Algorithm On Cuda Platform. The paper involved passageway is amassed for transmitted keys and gather data from a Wireless Sensor Network. Due to its ease and usefulness. [9]

Ying Tan & Ke Ding proposed the, A Survey on GPU-Based Implementation of Swarm Intelligence Algorithms. This follows the widespread appraisal of GPU-based equivalent SIAs in agreement with a recently proposed catalog. Serious concerns for the capable parallel completion of SIAs are also described in detail. [10]

Rafal Skinderowicz proposed, Ant Colony System with Selective Pheromone Memory for TSP. Presents, every part of trails are hoard within a pheromone recollection, which in the casing of the Travelling Salesman Problem wants $O(n^2)$ memory storeroom, wherever n is the coverage of the problem occurrence. [11]

Rafal Skinderowicz proposed, Ant Colony System with Selective Pheromone Memory for SOP. This paper lengthen the preceding work lying on a inventive

discerning pheromone recollection imitation for the ACS in which pheromone principles are lay-up barely for the preferred detachment of trails.[12]

Pavel Kromer, Jan Platos, Vaclav Snasel & Ajith Abraham proposed, A Comparison of Many-threaded Differential Evolution and Genetic Algorithms on CUDA. In this paper, compare incongruity expansion and intrinsic algorithms employ on CUDA while work out the self-governing errands scheduling problems.[13]

Byunghyun Jang, Dana, Perhaad Mistry & David Kaeli proposed the, Exploiting Memory Access Patterns to Improve Memory Performance in Data-Parallel Architectures. Intend to encircle techniques for ornamental reminiscence proficiency of employment, stand on the scrutiny and tagging of memory admittance samples in round bodies, blotch vectorization via records alteration headed for the benefit of the vector-based structure and algorithmic memory collection meant for scalar-based architectures. [14]

Kai-Cheng Wei, Chao-Chin Wu & Chien-Ju Wu., proposed Using CUDA GPU to Accelerate the Ant Colony Optimization Algorithm. This paper subsequent a new-fangled comparable scheme, which is labeled the Transition Condition Method. The provisional upshot have confirmed that the authority of solutions does not be sacrificed in the foundation of speed-up. [15]

III. SYSTEM OVERVIEW

A. *Problem Statement*

To develop a parallel environment framework which will be practical intended for resolve the optimization troubles very easily and effectively. Optimization problems which are NP tribulations can be cracked by means of the various heuristic algorithms. Appropriate in the track of this the solution for a difficulty is gained quickly.

Use of a mixture of heuristic algorithms that will exist functional for implementation of system. Getting the approximate key to the different optimization problem within little quantity of instance with successful memory employment furthermore increase the speedup of the operations. [13]

B. *Proposed System Architecture*

Proposed system is mainly classified into two segments such that CPU and GPU. Each segment performs its tasks independently.

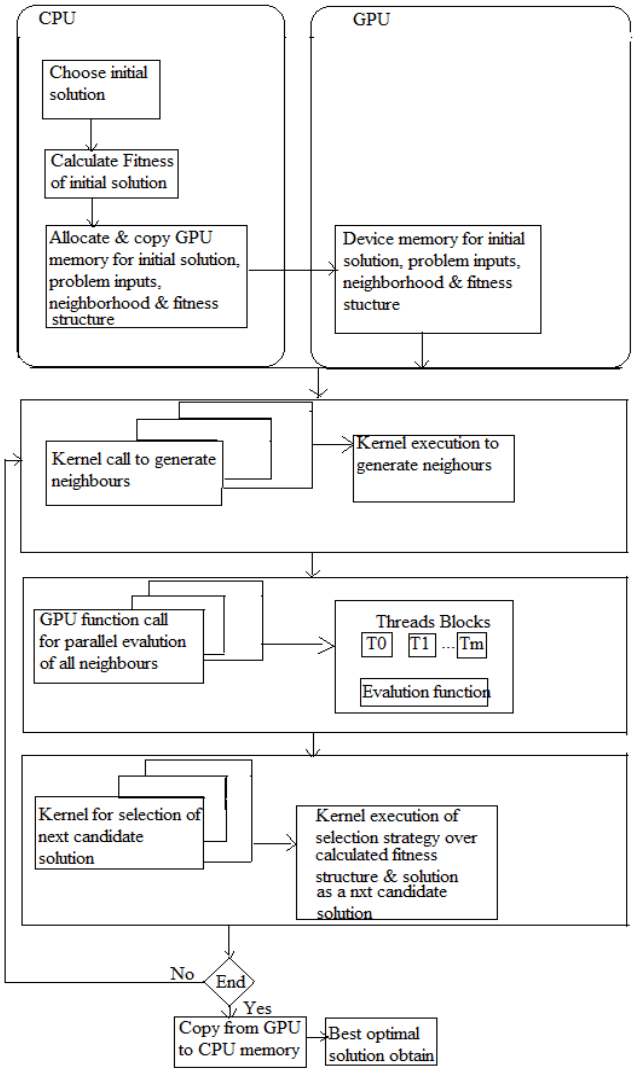

Figure 1. Proposed system Architecture

Figure.1 indicates the flow of proposed system. Initial solution is chosen to evaluate on CPU. GPU memory is to be paid for initial solution, problem inputs (for example, lookup table for TSP problem), and neighborhood. As results of each neighbor valuation need to be hoard for judgment, fitness constitution is billed for storing results on GPU. Initial solution and problem inputs are copied on GPU memory. To produce neighborhood of candidate clarification, kernel function is instigated. Every neighbor is stored in a mapped memory locality. Next kernel function is initiated to evaluate every neighbor in parallel by mapping a thread to it. Another kernel is started to select best solution among all evaluated neighbors. Reduction techniques can be engaged for this if minimum fitness needs to be computed. Chosen solution is configured a next candidate for evaluation. Lead of this approach in excess of iteration-level is lessening

memory relocate operations and deployment of GPU calculating influence for helpful work.

C. Memory organization flow for GPU Architecture

Simplified Memory association flow for GPU configuration plan is illustrated in the Figure 2.

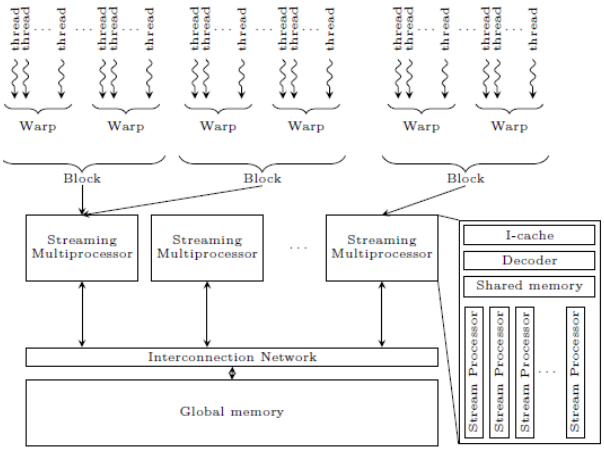

Figure 2. GPU Architecture with Memory Organization flow

Usually the GPU surrounds an infinite quantity of stream processors, each one belonging to solitary of abundant streaming multi-processors(SM). [1] Next to some specific instant a lonely interior realized reckoning for an exacting thread. Threads are accumulated into the obstructs, among each hunk given to a individual SM. Cores belongs to the identical SM share, amid others, the reports , restricted memory, statistics fetch and interpret, and load or store units [15]. By allocating an assortment of secondary units, extra computing cores can be filled into a solo SM at the outlay of a few rating of elasticity of calculations of unit cores. This is time-consuming, repeatedly on regulates of compound series, in accessing on the whole memory is the foremost obstacles to capable analogous computations on GPUs. The GPU memory bus is wider than the memory bus of the CPU and has a relatively huge bandwidth, but it is often still not adequate to provide data for every of the crucial element of the GPU. Designed for this explanation, the GPU programming model assumes that utilized an unwieldy number of threads.

IV. PROBLEM FORMULATION

The system takes sequence of visits between cities in provisions of co-ordinates values.

The projected structure S is definite as given:
S={X, F, N, T, K, R, W}
Where, X: Set of candidate solutions.
F: Fitness of solution.
N: Neighborhood of candidate solutions.
T: Set of threads.
K: Fitness structure.
R: Reduction function.
W: Best solution.
n:Size of solution.
m: Size of neighborhood.

Let, X: $\{x_1 , x_2 ,....,x_n \}$,
N: $\{N_1 , N_2 ,..., N_m \}$,
N1 =$\{x_{11} ,x_{12} ,....,x_{1n} \}$,
N2 =$\{x_{21} ,x_{22} ,....,x_{2n} \}$,
...
Nm= $\{x_{m1} ,x_{m2} ,....,x_{mn} \}$,
T: $\{t_1 , t_2 ,..., t_m \}$,
K: $\{k_1 ,k_2 ,...,k_m \}$,
W: $\{w_1 , w_2 ,..., w_n \}$

Function f1:
It is an evaluation function which calculates fitness of a solution over an operation 'op'.
$f1(X) \rightarrow f1(x_1 \text{ op } x_2 \text{ op....op } x_n) \in F$

Function f2:
It takes candidate solution X as an input and generates neighbors N from the respective encoding.
$f2(X) \rightarrow (N_1 , N_2 ,..., N_m) \in N$

Function f3:
It maps every thread from set of threads T to corresponding neighbor and executes evaluation function over it to generate fitness K.
$f3(T, N) \rightarrow \{T_1 \rightarrow E(N_1), T_2 \rightarrow E(N_2) ,..., T_m \rightarrow E(N_m)\}$
$\rightarrow (k_1, k_2, ...,k_m) \in K$

Function f4:
It takes fitness structure as an input and returns best solution.
$f4(K) \rightarrow \{w_1 , w_2 ,..., w_n \} \in W$

Function f5:
It sets best chosen solution W as a candidate solution which will be further optimized by iterating algorithm.
$f5(W) \rightarrow X$

V. EXPERIMENTAL ANALYSIS

The results and their analysis done with the help of different parameter, such as Dataset is used, number of cities, amount of blocks and required time for execution. To catch the efficient memory exploitation, synchronized execution, saving time and increasing speedup of execution. So that the speedup factor is enhance and get the best optimal solution. TSPLIB is a library of sample instances for the TSP (and related problems) from various sources and of various types.

Experimental results were evaluated using the system requirements as Intel Pentium i3 (Min), NVIDIA graphics card GT610. The Operating System these experiments are run is Ubuntu 14.04 and the complier is Gcc/Gtt & Nvcc, with Nvprof Profiler. Where the programming language is CUDA 7.0. The factors to be in considerations during the testing are speed-up, execution time & effective utilization of the memory.

TABLE I. PARALLEL COMPUTATION SPECIFICATIONS ON GPU

Sr. No.	Dataset	# cities	# Blocks	Time(sec)
1.	Ch130	130	100	0.0146
			200	0.029
			300	0.0421
			400	0.0421
			500	0.06
2.	Bier127	127	100	0.0139
			10k	1.0561
			20k	1.957
3.	Pr1002	1002	100	1.5762
			200	3.285
			300	4.8314
			2k	29.1702
			10k	144.644
4.	Ch130	130	10k	1.0689
			20k	2.1223
			50k	5.2541
			100k	10.524
			200k	21.0184

The above Table I indicate the different datasets with number of blocks and time in seconds required for computations. That can be clarifying in fine points with the graphs.

A. Performance for Dataset Ch130

Dataset Ch130 defines the cities are 130. That tested with varying number of blocks. As per the results, if increase in quantity of blocks then also increases quality of solution that is minimizes tour length in this case.

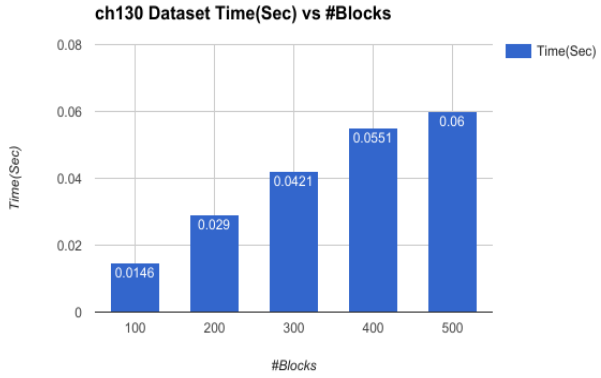

Figure 3. Performance for Dataset Ch130

B. Performance for Dataset Bier127

Dataset Bier127 including cities are 127. That tested with the number of blocks. The block size is initially 100 then goes on 10k, 20k and so no.

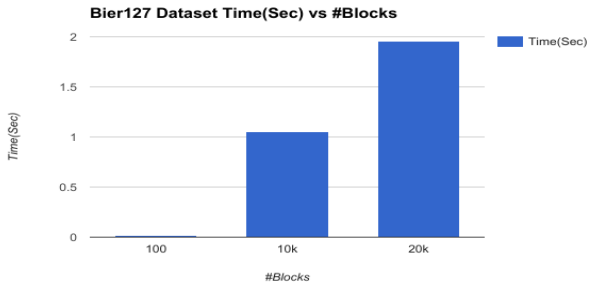

Figure 4. Performance for Dataset Bier127

C. Performance for Dataset Pr1002

Dataset pr1002 contains the 1002 cities with the blocks. It seems that if increase in no. of blocks then also increase in the quality of the solution.

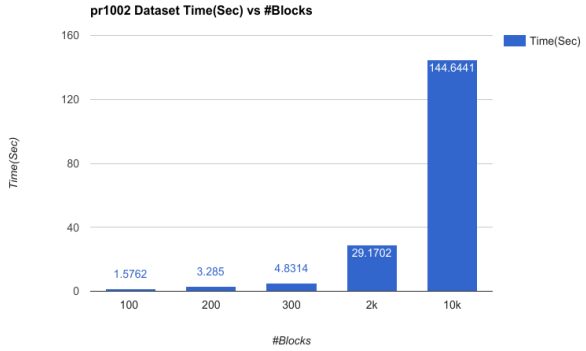

Figure 5. Performance for Dataset Pr1002

D. Performance for Dataset Ch130

Dataset Ch130 together with the 130 cities. So as to tested through the amount of blocks. The block size is primarily 10k after that goes on 20k, 50k and so.

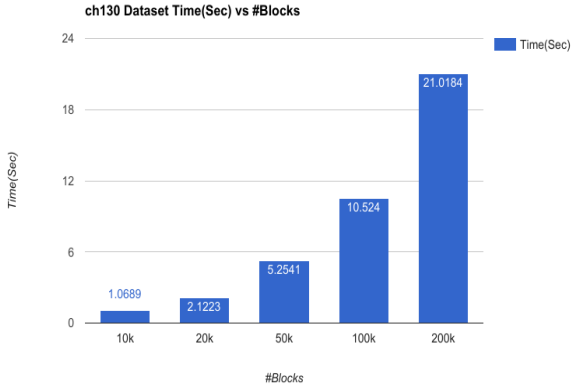

Figure 6. Performance for Ch130

TABLE II. COMPARISON OF OPTIMIZED TOUR LENGTH

Dataset	Cities	Optimized Tour Length	Time (Sec)	Optimized Tour Length, Rocki, K.,,et.al [?]
sw24978	24978	908598	15071.55	949792
usa13509	13509	20984503	2310.117	22090071

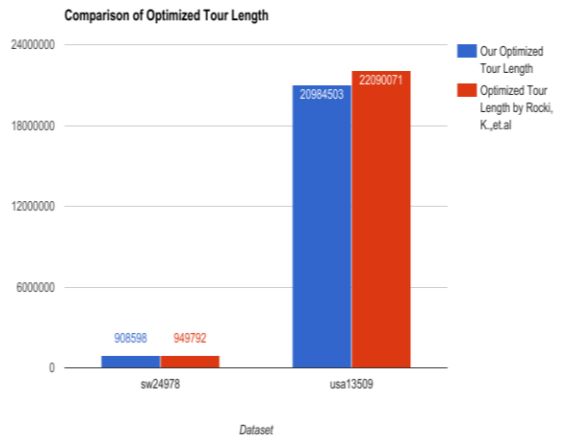

Figure 7. Comparison of Optimized Tour Length

Comparison of total execution time against quantity of blocks and amount of cities. Time required to execute is proportionally increasing as number of blocks of threads increases. It is observed that time is also increases as the no. of cities are increased.

VI. CONCLUSION

The proposed system can provide scalable and capable memory utilization. It is optimized for data transfer between CPU and GPU memory. Local search metaheuristic algorithm is made scalable by assigning an only clarification for each block of threads. This approach also reduces global memory access by threads of block. It is tested by implementation on travelling salesman problem with standard datasets. To achieve the finest optimal solution of good quality. Which belongings to the precision, strength, scalability and so forth. It will be opportune than habitual scrutiny systems.

REFERENCES

[1] Rafal Skinderowicz, "*The GPU-based Parallel Ant Colony System"*, University of Silesia (Institute of computer science), Poland, pp.41-205, 2016.

[2] Laurence Dawson and Iain A. Stewart., *"Improving ant colony optimization performance on the GPU using CUDA"*, In Proceedings of the IEEE Congress on Evolutionary Computation, Mexico, pp.1901-1908, 2013.

[3] Wojciech Czech, David A. Yuen, *"Efficient Graph Comparison and Visualization using GPU"*, The 14th IEEE International Conference on Computational Science and Engineering, China, pp. 561-566, 2011.

[4] Maida Arnautovic, Maida Curic, Emina Dolamic and Novica Nosovic, *"Parallelization of the Ant Colony Optimization for the Shortest Path Problem using OpenMP and CUDA"*, MIPRO, Croatia, pp.7-12, 2013.

[5] Tomohiro Okuyama, FumihikoIno, Kenichi Hagihara, *"A Task Parallel Algorithm for Computing the Costs of All-Pairs Shortest Paths on the CUDA compatible GPU"*, International Symposium on Parallel & Distributed Processing with Applications, Australia, pp.284-291 2008.

[6] Kamil Rocki & Reiji Suda, "*High Performance GPU Accelerated Local Optimization in TSP"*, IEEE 27th International Symposium on Parallel & Distributed Processing Workshops, USA, pp. pp.1788-1796, 2013.

[7] M. Dorigo, LM. Gambardella., *"Ant colony system: a cooperative learning approach to the traveling salesman problem"*, IEEE Trans. Evolutionary Computation, Vol.1, Issue.1, pp.53-66, 1997.

[8] Marco Dorigo, Vittorio Maniezzo, Alberto Colorni., *"Ant system: optimization by a colony of cooperating agents"*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 26, Issue.1, pp.29-41, 1996.

[9] U. Cekmez, M. Ozsiginan, O.K. Sahingoz., *"A uav path planning with parallel aco algorithm on cuda platform"*, International Conference on in Unmanned Aircraft Systems (ICUAS), USA, pp.347-354, 2014.

[10] Y. Tan and K. Ding, *"A survey on gpu-based implementation of swarm intelligence algorithms",* IEEE Transactions on Cybernetics, Vol.46, Issue.9, pp.1-14, 2015.

[11] Rafal Skinderowicz, *"Ant colony system with selective pheromone memory for TSP"*, International Conference ICCCI 2012, Vietnam, pp. 483-492, 2012.

[12] Rafal Skinderowicz. *"Ant colony system with selective pheromone memory for SOP."*, 5th International Conference ICCCI 2013, Romania, pp. 711-720, 2013.

[13] Pavel Kromer, Jan Platos, Vaclav Snasel, *"Nature-inspired meta-heuristics on modern gpus: State of the art and brief survey of selected algorithms"*, International Journal of Parallel Programming, Vol.42, Issue.5, pp.681-709, 2014.

[14] Byunghyun Jang, Dana Schaa, Perhaad Mistry, David R. Kaeli. *"Exploiting memory access patterns to improve memory performance in data-parallel architectures"*, IEEE Trans. Parallel Distrib. Syst., Vol.22, Issue.1, pp.105-118, 2011.

[15] Kai-Cheng Wei, Chao-Chin Wu, Chien-Ju Wu, *"Using CUDA GPU to accelerate the ant colony optimization algorithm"* , International Conference on Parallel and Distributed Computing Applications and Technologies, China, pp.90-95, 2013.

Authors Profile

Prof. Santosh Kumar, currently working as Assistant Professor, in Computer Engineering Department, SITRC, Nashik affiliated to Savitribai Phule Pune University. He has Postgraduate (M.Tech) in Computer science & Engineering in 2010. His research areas are, High Performance Computing, Big data & analytics and GPU. He has published more than 25 papers in international Journal, National Journal and conferences,

Attended Various Workshop and seminar. He has deliver expert lecture on system programming and Latex.

Miss. Swati S. Dhable, currently working as a PG student, in Computer Engineering Department, SITRC, Nashik affiliated to Savitribai Phule Pune University. She completed her B.E (Computer science & Engineering) in 2014 at SIEM, Nashik. Her research interest includes Parallel Computing, Big data and GPU.

Prof. (Dr) Amol D. Potgantwar, currently working as a HOD, in Computer Engineering Department, SITRC, Nashik affiliated to Savitribai Phule Pune University. He received Ph.D. degree in Computer science & Engineering. His research areas are, Mobile computing, wireless technology, near field communication, Image Processing and Parallel Computing. He has registered five (05) patents on Indoor Localization System for Mobile Device Using RFID & Wireless Technology, RFID Based Vehicle Identification System and Access Control into Parking. A Standalone RFID and NFC Based Healthcare System.

