COMPUSOFT, An international journal of advanced computer technology, 4 (5), May-2015 (Volume-IV, Issue-V)

ISSN:2320-0790

An International Journal of Advanced Computer Technology

Research on Solving Travelling Salesman Problem using Rank Based Ant System on GPU

Khushbu Khatri [1], Prof. Vinit Kumar Gupta[2]
[1] Student in ME (CSE), [2]Professor
Computer Dept, Hasmukh Goswami College Of Engineering, Ahmedabad, India

Abstract: Ant Colony Optimization (ACO) is meta-heuristic algorithm inspired from nature to solve many combinatorial optimization problems such as Travelling Salesman Problem (TSP). There are many versions of ACO used to solve TSP like, Ant System, Elitist Ant System, Max-Min Ant System, Rank based Ant System algorithm. For improved performance, these methods can be implemented in parallel architecture like GPU, CUDA architecture. Graphics Processing Unit (GPU) provides highly parallel and fully programmable platform. GPUs which have many processing units with an off-chip global memory can be used for general purpose parallel computation. This paper presents a parallel Rank Based Ant System algorithm to solve TSP by use of Pre Roulette Wheel Selection Method.

Keywords: Ant Colony Optimization, Rank Based Ant System, TSP, GPU, CUDA Architecture

I. INTRODUCTION

Travelling Salesman Problem is NP- hard problem in combinatorial optimization, important in operation research and theoretical computer science.[1] A meta-heuristic method, Ant Colony Optimization(ACO) can be used to solve TSP problem. ACO is population based search method inspired from real ants in which ant find shortest path between foods to nest. In our approach artificial ants will work and perform task based on behavior of real ants. In reality, real ants will follow chemical substances known as Pheromone and dropped by earlier ants. Same way artificial ant will perform two steps: Tour Construction and Pheromone updation. Each artificial ant will construct tour independently that's why this algorithm is very suitable for parallel execution.

Graphics Processing Unit (GPU) is very popular from recent years for the parallel execution of programs. Compute Unified Device Architecture (CUDA) was presented for parallel processing using GPU since 2006. GPUs are designed to host thousands of low frequency and efficient cores for handling multiple data parallel task by running thousands of lightweight threads concurrently.

The objective of this survey is to get knowledge about the different methods of ACO to solve TSP problem on parallel platform.

The rest of this paper is organized as follows. Section 2 Gives background knowledge about TSP, ACO and GPU. Section 3 presents proposed methodology Rank Based Ant

System for parallel architecture to solve TSP. section 4 shows performance evaluation of proposed work. Conclusion of this research is given in Section 5.

II. BASICS OF ANT COLONY OPTIMIZATION ALGORITHM AND GPU COMPUTING

A. TRAVELLING SALESMAN PROBLEM

Travelling Salesman Problem can be represented by complete graph G=(V,E,d) with V represents nodes (city), E represents edge, path between two nodes and d_{ij} represents distance between node i and j. TSP is used to finding complete closed shortest path which by visiting all the cities.

B. ANT COLONY OPTIMIZATION

The ACO algorithm introduced as nature inspired meta heuristic for solution of many combinatorial optimization problems [2], [3]. Here nature inspired is in the mean about real ants exploring path between nest and food. Ants will always follow the path whereprobability of pheromones are greater. There are two main stages in algorithm (Fig. 1*): Tour Construction and Pheromone Updation.* These steps are repeated until some conditional criteria met. There are many versions of ACO like Ant System, Elitist Ant, MAX-MINAnt System and Rank based Ant System algorithm. In

ACO, the characteristics of real ant colony are exploited in simulated ant colonies to solve problem. The basic algorithm is presented below:

```
Procedure ACOMetaheuristuc
    Set parameters, initialize pheromones level
While( termination condition not met) do
        Construct ant's solution
        Update Pheromones
    End
End
```

Figure.1. Overview of ACO algorithm [4]

As shown in above fig. 1 algorithm presents three steps: *(i) Initialization, (ii) Tour Construction and (iii) Pheromone Update.*
Given n cities, distance between the cities and m ants, the details of steps are given below.
(i) Initialization
In the initialization step, the initial parameters for pheromone trail are determined using greedy manner [5]

$$\tau(i,j) = \frac{n}{Cg} \quad \forall(i,j) \in L \tag{1}$$

where L denotes all edges between cities and Cg is total length of a tour obtained by greedy algorithm such that starting from an arbitrary city as current city, the shortest edge that connects current city and unvisited city is selected.

(ii) Tour Construction
In tour constriction, *m* ants independently visit each city exactly once. Each ant starts at a city decided randomly, and selects which city to visit probabilistically. A probability *pk*(*i*, *j*) to visit city *j* from city *i* for ant *k* is computed by Eq. (2).

$$pk(i,k) = \begin{cases} \dfrac{f(i,j)}{\sum_{l \in Nk(i)} f(i,j)} \text{if j } \in \text{ Nk(i)} \\ 0 otherwise \end{cases} \tag{2}$$

Where *Nk*(*i*) is a set of unvisited adjacent cities for ant *k* in city *i*, and *f*(*i*, *j*) is a fitness between cities *i* and *j*

$$f(i,j) = [\tau(i,j)]^{\alpha} . [\eta(i,j)]^{\beta} \tag{3}$$

These equations mean that when the quantity of pheromone between cities *i* and *j* is large and the distance between cities *i* and *j* is short, the probability to visit city *j* becomes large. Using this probability, each ant visits each city exactly once, ending up back at the starting city.

(iii) Pheromone Update
When all the ants complete tour construction, the Pheromone assigned between cities is updated using information of each tour. The update consists of pheromone evaporation and pheromone deposit.
Pheromone evaporation is utilized to avoid falling into local optima. Every quantity of pheromone is reduced with the following equation:

$$\tau(i,j) = (1-\rho)\tau(i,j) \quad \forall(i,j) \in L \tag{4}$$

Where ρ is an evaporation rate of pheromone.
After the pheromone evaporation, for every pheromone between cities, pheromone deposit is performed with the results of the tour construction as follows

$$\tau(i,j) = \tau(i,j) + \sum_{k=1}^{m} \Delta\tau k(i,j) \quad \forall(i,j) \in L \tag{5}$$

Where $\Delta\tau_k(i, j)$ is a quantity of pheromone between cities *i* and *j* which is deposited by ant *k*. The quantity is computed by

$$\Delta\tau k(i,j) = \begin{cases} \dfrac{1}{CK} \text{if e j } \in \text{ e (i,j) } \in \text{Tk} \\ 0 otherwise \end{cases} \tag{6}$$

Where *Ck* is the tour length of ant *k*, and *Tk*is the tour of ant *k*. This equation means that when an edge is included in shorter tours and is selected by more ants in the tour construction, the quantity of additional pheromone is larger.

C. GRAPHICS PROCESSING UNIT (GPU)

GPU-accelerated computing is the use of a graphics processing unit (GPU) together with a CPU to accelerate scientific, analytics, engineering, consumer, and enterprise applications. GPU has a massively parallel architecture consisting of thousands of smaller, more efficient cores designed for handling multiple tasks simultaneously [7].

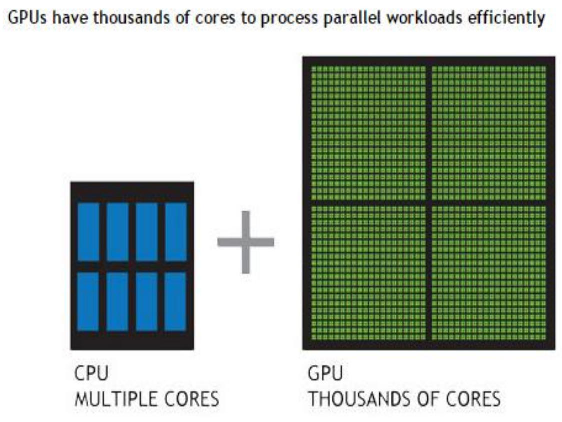

GPUs have thousands of cores to process parallel workloads efficiently
CPU
MULTIPLE CORES
GPU
THOUSANDS OF CORES

Figure.2. GPU with thousands of core [7]

As shown in above fig. 2, CPU contain multiple cores where GPU having thousands of cores and handling millions of lightweight threads significantly accelerate the data parallel approach. In contract to ACO algorithm each thread will perform task as of independent ant. Independently tour construction can b done by each thread.

III. LITEREATUE SURVEY

A. A Parallel Ant Colony Optimization Algorithm with GPU-Acceleration based on All-In-Roulette Selection[11]

In this paper, Jie Fu, Lin Lei Guphuo Zhou [11] presents a parallel MAX-MIN Ant System (MMAS) based on GPU + CPU to solve Travelling Salesman Problem. They adopt the fully matrix based data parallel computing in GPU within MATLAB. In this method they creates matrix for all three components TAU for pheromones table, TABOO an integer matrix for keeping data for city is visited or not, and one probability matrix which will maintain the fitness function between all cities. There is another new proposed matrix AIR (All-In-Roulette), used for selection approach based on all previous matrix. Such way, proposed methods will fully utilize the GPU to speed these computations up.

B. Improving Ant Colony Optimization performance On the GPU using CUDA [12]

In this paper, Laurence Dawson, Iain Stewart [12] , presents a new parallel implementation of roulette wheel selection method called Double-Spin Roulette (DS-Roulette) which significantly reduces implementation of pheromone update. This paper mainly focuses on both performance of algorithm and quality of solutions by implementing on GPU using NVIDIA CUDA.

DS-Roulette striking the modern hardware architecture, increases parallelism and decreases the execution time and still enable to construct high quality tours. The authors adopts pheromone updation phase of MAX-MIN Ant System to achieve significant speed ups the proposed algorithm.

C. Using CUDA GPU to Accelerate the Ant Colony Optimization Algorithm [13]

In this paper, Kai-Cheng Wei, Chao-chin Wu, Chien-Ju Wu, proposed a new parallel method called Transition Condition Method. In this paper adopted strategy, assigns each ant to a CUDA thread block and also apply independent Roulette Wheel Selection mechanism. The proposed parallel method works as follows:
Step 1 : each CUDA thread independently calculates transition condition and checks city has been visited or not. Finally transition condition and Tabu list value is multiplied per city and stored as Product Of Previous array.
Step 2: Transition probability can be calculated by adding all Product Of Previous (POP) values. Now each city's POP value will be divided by sum of all Product Of Previous.

Step 3: Now based on division answer next city will be selected in tour.

But author proposed method in that step 2 will be removed. And based on Product Of Previous Array next city will selected as next city. No need to find sum of Product of Previous which accelerate the performance of algorithm and reduces execution time.

D. High Performance GPU Accelerated Local Optimization in TSP [14]

In this paper KamilRocki, RejiSuda [14], focuses on Local Optimization techniques in TSP. in general phenomena, when problem size growing the time spent on local optimization comparing graph edges grows significantly. In this main focus on problem division scheme exploiting data locality which allows to solve arbitrarily big problem instances using GPU and parallel implementation of algorithm.

The problem repeating series of steps is called 2- opt exchanges. According to proposed results at least 90% of execution time during the Iterated Local Search (ILS) is spent on 2-Opt checks and that no strongly increases with size growing

E. Parallel Implementation of Travelling Salesman Problem using ACO [15]

In this paper, Gaurav Bhardwaj, Manish Pandey [15], presents importance of parameters used in ACO algorithm for solving TSP on OPEN CL. Parameter α shows the dependency of the pheromone to find the next city to visit. If the value of α is too high then it shows the dependency of the algorithm on the pheromone value which may lead to an suboptimal result as the new ant will follow the path followed by the previous ants leads to the initial stagnation. Whereas very low value of α shows the low dependency of the algorithm on the pheromone content which may lead to follow the path with the nearest neighbor.

Parameter β shows the dependency of the algorithm on the heuristic value. Similarly if the value of β is too high than it shows that the algorithm depends upon the heuristic value and it will choose the next city with a minimum distance where as if it is too low than only pheromone amplification is at work

ρ is the evaporation rate of the pheromone. As the initial pheromone update may lead to the suboptimal solution. High pheromone evaporation rate (ρ) doesn't affect the pheromone content as the change is too less. Whereas lower value of ρ leads to the negative affect for the pheromone content as it becomes too low to be recognized.

F. A UAV Path planning with parallel ACO algorithm on CUDA platform [16]

In this paper, UgurCekmez, MustufaOzsiginan [16], has proposed algorithm for calculating UAV path in following four steps:

(a) *Random Number Generation:* In tour construction step, each ant will select the next city using probabilistic model. It produces random number by using RNG seeder taken from current city index. N piece of seeders are stored in 1-D array by N threads.

(b) *Distance Table Calculation:* Distance table is calculated ny N^2 threads where N is equal to the no. of cities.

(c) *Initialization :* Before the tour construction processes of the ants, the initial pheromones must be assigned the vertices among the cities so that an ant can decide which city to visit next.

(d) *Tour Construction:* that an ant is represented by a thread block and each thread block includes number of threads equal to the number of cities. In such a case, assume that each ant has N feet where each feet (thread) makes some computations in parallel and after all, one foot takes the results, compares them all and selects the best choice to visit.

G. Improving Ant Colony Optimization algorithm based on Dynamically Adjusting Ant Number [17]

In this paper, DewenZeng, Qing He, Binheng [13] proposes a Ant Colony Optimization Algorithm by dynamically adjusting ant number. In this approach the only part of the ants passing the shorter path is allowed to release pheromone and update the total ant number randomly or fixedly in algorithm iterative process.

H. Parallelization of Ant Colony Optimization for the shortest path problem using OpenMP and CUDA [18]

In this paper Maida Arnoutoive, Maida Cunic presents very good comparison of parallel algorithm of ACO for shortest path problem in Open MP and CUDA. In OpenMP, the problem is implemented by use of no. of threads in that calculate tour construction and submit pheromone to master thread which update the pheromone to table and the same process repeats for next iteration.

In CUDA architecture, GPU has always been a processor with ample computation resources. CUDA allows execution of multiple threads per block in parallel, each thread in the same block executes the same instruction, each ant represented as single thread.

From this paper results shows up to 50% faster execution when the using of GPU.

I. Research and Improvement of Ant Colony Algorithm based on TSP [19]

In this paper, Ling Lin, Huailin Dong, Qingfeng Wu, elaborates the basic principle and mathematical models of typical ant colony algorithm for solving TSP and analyzes impact of the optimal parameters to performance of algorithm. In this paper parameter related to pheromones and heuristic information and evaporation (α, β, ρ) is adjusted dynamically to improve the performance of algorithm. To improve global search capability, time varying function ρ(t) is used to achieve the adaptively adjust the constant parameter ρ and initialize ρ(0) = 1

$$\rho(t) = \begin{cases} D.\rho(t-1), & where\ D.\rho(t-1) \geq \rho_{min} \\ \rho_{min}, & otherwise \end{cases}$$

D is constant in range 0 <D(),ρ_{min} is minimum pheromone evaporation factor.

IV. PROPOSED METHODOLOGY

In this section we are going to present a new parallel algorithm in following section

A. Algorithm To Solve Tsp Using The Parallel Rank Based Ant System:

```
BEGIN
STEP 1 : //INITIALIZE PARAMETERS
         TAU0 = RANDM();
         ALPHA = 1;
         BETA = 2;
         ROH=0.5;
         //INITIALIZE PHEROMONE TRAILS
         For(every edge i, j)
         {
             TAU(i,j) =TAU0;
         }
STEP 2 : //MAIN LOOP PARALLEL TOUR CONSTRUCTION
         // FIND PROBABILITY TO SELECT NEXT CITY
       -global-
         Parallely Calculate Probability by calcProb() function from each city using
         preRouletteWheelSelection(); method;

         //WAIT FOR EVERY THREAD TO COMPLETE EXECUTION
       -synchronizethreads();
STEP 3: Find The Tour[i][j] For Every Ant K.
STEP 4: Calculate Cost Of Each Tour_k Constructed By Ant K
STEP 5: // ASSIGN RANK TO EACH ANT BASED ON TOUR LENGTH
         Rank[n] = sort Ant in ascending order based on tour_cost
STEP 6: // ALLOW TO UPDATE PHEROMONE ONLY TO TOP ELITIST ANTS
         For (k = 1 to ELITIST from RANK[ ] )
         {
             For ( every edge i, j)
                 UPDATE_PHEROMONE(K);
         }
STEP 7 : Repeat Step- 2 To Step 6 Up To Max_Iteration
END
```

In above algorithm starting from Step 1,initializeS parameters by initial values and all edges by initial pheromones. In Step 2 each ant will select random city for tour construction. In that tour construction is done based on preRouletteWheelSelection method which is shown in Fig. 3.

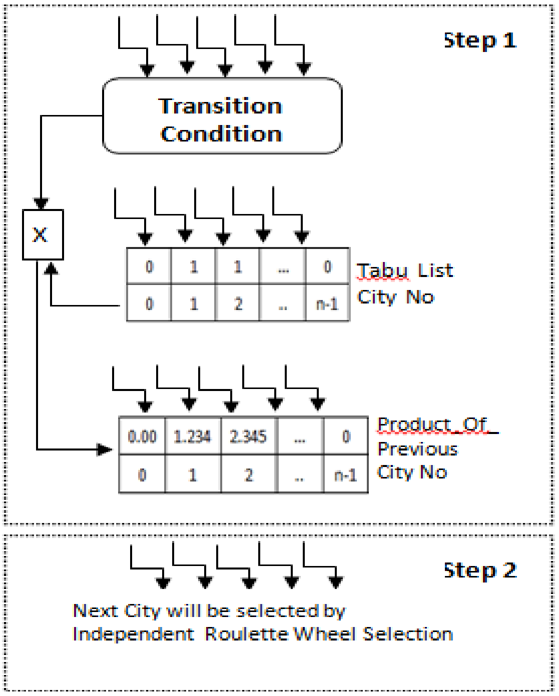


Fig. 3 preRouletteWheelSelection Method

B. PHEROMONE UPDATION FOR RANK BASED ANT SYSTEM

$$\tau(i,j) \;=\; \rho.\tau(i,j)(t) + \Delta\tau(i,j) + \Delta\tau'(i,j)$$

$$\Delta\tau(i,j) \;=\; \sum_{\mu=1}^{\sigma-1} \Delta\tau_{i,j}^{\mu}$$

$$\Delta\tau_{i,j}^{\mu} \;=\; \begin{cases} (\sigma-\mu).\dfrac{Q}{L_\mu}, if\ \mu th\ best\ ant\ on\ edge(i,j) \\ 0, \quad otherwise \end{cases}$$

$$\Delta\tau'(i,j) \;=\; \begin{cases} \sigma.\dfrac{Q}{L^+}, if\ (i,j)\ part\ of\ best\ sol.found \\ 0, \quad otherwise \end{cases}$$

$\Delta\tau_{i,j}^{\mu}$	=	Increase of trail level on edge (i, j) caused by the µ-th best ant
L_μ	=	Tour length of µ-th best ant
$\Delta\tau'(i,j)$	=	Increase of trail level on edge (i, j) caused by the elitist best ant
σ	=	Number of Elitist ants
L^+	=	Tour length of best solution found

Pheromones Updation will be done by only Elitist ants. User can decide Elitist ants based on no of ants. The purpose of this method is to just path which belongs to shortest tour cost will be upgraded. If any ant which travels longer tour that will not allowed to update pheromones.

V. PERFORMANCE EVALUTION

We have implemented our Rank Based Ant System(RAS) for TSP using CUDA C. we have used NVIDIA GeForce GTX630M with 96 cores running in 950 MHz and 2 GB memory. The operating system is Microsoft Windows 7. Its compiler environment is Nsight Visual Studio Edition 4.2. Dram memory size is 4GB. The implementation uses many standard TSP instances from the TSPLIB Library [9]. In our implementation of the ACO algorithm, key parameters such as α, β a pheromone value and so on are given on the basis of the values recommended by [10]. In CUDA, it is important to determine the number of blocks and the number of threads in each block. It greatly influences the performance of the implementation on the GPU. We had shown below the GPU based programme Ant System (AS) and Rank based Ant System (RAS)and the results of overall performance are shown.

The below graph shows the difference between execution time of Ant System (AS) algorithm and Rank Based Ant System (RAS) algorithm. The graph contains three TSPLIB instances. We can see the decreased execution time of RAS compare to AS.

The Distance Comparison Graph shows that difference between optimal distance is less in both technique AS and RAS. Here main focus on execution time which is improved in proposed methodology (RAS)

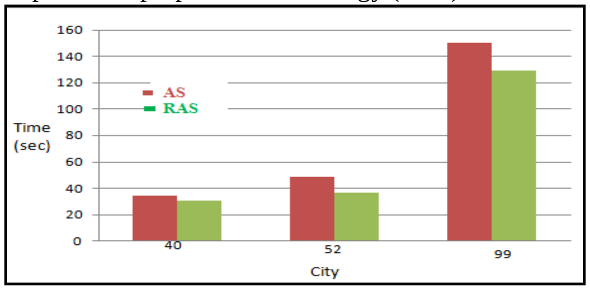
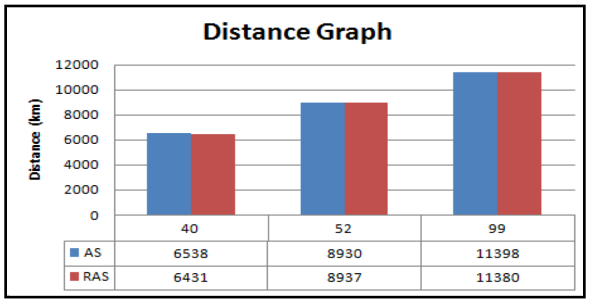


Fig .4 Result Analysis between AS and RAS



	40	52	99
AS	6538	8930	11398
RAS	6431	8937	11380

Fig. 5 Distance Comparison Graph Between AS and RAS

VI. CONCLUSION

A Parallel Rank based Ant System is one of the version of Ant Colony Optimization. In that only top Ranking elitist ants will allow to update Pheromones. This method will used to solve Travelling Salesman Problem on parallel architecture by use of GPU. We also proposed a preRoulette Wheel Selection Method for Tour construction. This method will select next city based on Product_Of_previous.

Experimental result of this study shows that Rank Based Ant System can improve performance by decreasing execution time compare to Ant system on GPU.

ACKNOWLEDGEMENT

I would like to express my deep sense of gratitude to my guide, Asst Prof. Vinit Kumar Gupta for his valuable guidance and useful suggestions. I would like to thank Asst Prof. Lokesh Gagnani also for his precious suggestion

REFERENCES

[1] Wikipedia, “Travelling Salesman Problem [Online]”,http://en.wikipedia.org/wiki/Travelling_salesman problem
[2] M. Dorigo, “Optimization, learning and natural algorithms,” Ph.D. dissertation, Dipartimento di Elettronica, Politecnico di Milano, 1992.
[3] M. Dorigo, V. Maniezzo, and A. Colorni, “The ant system: Optimization by a colony of 11cooperating agents,” IEEE Transactions on Systems, Man, and Cybernetics–Part B, vol. 26, no. 1, pp. 29–41, 1996.
[4] M. Dorigo and T. St¨utzle, Ant Colony Optimization. MIT Press, 2004.
[5] G. Gutin, A. Yeo, and A. Zverovich, “Traveling salesman shoud notbe greedy: domination analysis of greedy-type heuristics for the tsp,”Discrete Applied Mathematics, vol. 117, pp. 81–86, 2002.
[6] Nvidia,C., “ABOUT GPU COMPUTING [Online]” ,http://www.nvidia.com/object/what-is-gpu-computing.html#sthash.VTFrtzbE.dpuf
[7] Wikipedia, “Graphics Processing Unit [Online]”, http://en.wikipedia.org/wiki/Graphics_processing_unit
[8] Ling Lin, Huailin Dong, Qingfeng Wu, “Research and Improvement of Ant Colony Algorithm based on TSP”, IEEE, 978-1-4244-8625-0/11/ - 2011
[9] TSPLIB WebPage August 2008, http://comopt.ifi.uniheidelberg.de/software/TSPLIB95
[10] Dorigo,M., et al., “Ant Colony Optimization ”, A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England, 2004.
[11] Jie Fu, Lin Lei Guphuo Zhou, “A Parallel Ant Colony Optimization Algorithm with GPU-Acceleration based on All-In-Roulette Selection” Third International Workshop on Advanced Computational Intelligence August 25-27, 2010 - Suzhou, Jiangsu, China.
[12] Laurence Dawson, Iain Stewart, “Improving Ant Colony Optimization performance On the GPU using CUDA” IEEE Congress on Evolutionary Computation, Cancún, México, June 2013.
[13] Kai-Cheng Wei, Chao-chin Wu, Chien-Ju Wu, “Using CUDA GPU to Accelerate the Ant Colony Optimization Algorithm” IEEE International Conference on Parallel and Distributed Computing, Applications and Technologies- 2013
[14] KamilRocki, RejiSuda , “High Performance GPU Accelerated Local Optimization in TSP “, IEEE 27th International Symposium on Parallel & Distributed Processing Workshops and PhD Forum- 2013
[15] GauravBhardwaj, Manish Pandey, “Parallel Implemetation of Travillening Salesman Problem using ACO”, International Journal of Computer Applications Technology and Research Volume 3– Issue 6, 385 - 389, 2014
[16] UgurCekmez, MustufaOzsiginan, “A UAV Path planning with parallel ACO algorithm on CUDA platform”, IEEEUnmanned Aircraft Systems (ICUAS), 2014
[17] DewenZeng, Qing He, Binheng, “Improving Ant Colony Optimization algorithm based on Dynamically Adjusting Ant Number”, IEEE International Conference on Robotics and Biomimetics December 11-14, 2012, Guangzhou, China
[18] Maida Arnoutoive, Maida Cunic, “Parallelization of Ant Colony Optimization for the shortest path problem using OpenMP and CUDA”, IEEE MIPRO 2013, MAY 20-24, Opatija.
[19] Ling Lin, Huailin Dong, Qingfeng Wu, “Research and Improvement of Ant Colony Algorithm based on TSP”, IEEE, 978-1-4244-8625-0/11/ - 2011