International Journal of Computer Applications Technology and Research
Volume 3– Issue 6, 385 - 389, 2014, ISSN: 2319–8656

Parallel Implementation of Travelling Salesman Problem using Ant Colony Optimization

Gaurav Bhardwaj
Department of Computer Science and Engineering
Maulana Azad National Institute of Technology
Bhopal, India

Manish Pandey
Department of Computer Science and Engineering
Maulana Azad National Institute of Technology
Bhopal, India

Abstract: In this paper we have proposed parallel implementation of Ant colony optimization Ant System algorithm on GPU using OpenCL. We have done comparison on different parameters of the ACO which directly or indirectly affect the result. Parallel comparison of speedup between CPU and GPU implementation is done with a speed up of 3.11x in CPU and 7.21x in GPU. The control parameters α, β, ρ is done with a result of best solution at 1, 5 and 0.5 respectively.

Keywords: Travelling Salesman Problem, Ant colony optimization, parallel, OpenCL, GPU.

1. INTRODUCTION

Travelling salesman problem [1] is an NP-hard problem in a set of combinatorial optimization problem. In travelling salesman problem we have to found a Hamiltonian circuit having minimum total edge weight. TSP has various applications such as JOB Scheduling, DNA sequencing, designing and testing VLSI circuits, graph coloring, vehicle routing etc. There are various methods to solve such type problems such as ANT colony optimization, neural network, Genetic algorithm etc.

ACO [2] is a heuristic algorithm for solving combinatorial optimization problem. ACO imitates the behavior of real ants to search food. Ants communicate indirectly to the agents of their colony with a trail of a chemical substance called pheromone. Pheromone is a chemical substance that shows the trace of an ant. Other ants follow the smell of the food and the trace of the pheromone to find out the minimum distance to the food.

Complex problem such as TSP needs huge computational power as well as time to solve. It takes lots of time for a single processor to solve such large problems single handedly.ACO can be implemented parallel with high efficiency.[4] Parallel computing is the new paradigm to solve such type of problems using General Purpose Graphical Processing Unit. GPUs are meant to do graphical processing such as simple arithmetic operations also on graphics in the form of matrices. So we can utilize GPUs processor to solve our problem to speed up the computational time.

GPU [9] consist of large no. of processors embedded together in a chip to perform a specific type of operations. Open CL [10] (OPEN Computing Language) is the framework used to write programs that can be executed on heterogeneous platforms.

This paper applies ACO to the Travelling Salesman Problem in heterogeneous platform using OpenCL framework to achieve parallelism in ACO. We have compared the time taken in sequential as well as the parallel program used to solve this problem with some standard graphs. [11]

In section 2 previous sequential approaches for ACO has been discussed with travelling salesman problem. In section 3 Travelling salesman problem with different approaches to solve this problem is discussed. Section 4 gives the briefing of the Ant Colony Optimization algorithm. Section 5 gives the parallel approach to solve TSP using ACO on GPU. Section 6 gives the experimental comparison with different parameters.

2. RELATED WORK

Travelling salesman problem is one of the oldest mathematical problems in history. Scientist had a great interest to solve such type of problem using different approaches. M.Dorigo and T.stizzle in 1992 [6] has designed an biological approach to solve such type of combinatorial optimization problem such as Travelling Salesman Problem called ACO. The first ACO algorithm was proposed by them called Ant System basic approach on ACO. Then many other algorithm were proposed based on it such Max-Min approach, Ant colony System. All these approaches are successors of Ant system. M.Dorigo has given the basic parallel approach to solve ACO parallel as he has discussed the basic parallel behavior of ants in real life. There after many parallel approaches has been delivered with the parallel strategies. This paper describes the parallel implementation of ACO on heterogeneous platform using OpenCL and comparing their parameters.

3. TRAVELLING SALESMAN PROBLEM

Travelling Salesman Problem represents a set of problem called combinatorial optimization problem. In TSP a salesman is given a map of cities and he has to visit all the cities exactly once and return back to the starting city with the minimum cost length tour of all the possible tour present in that map. Hence the total no. of possible tour in a graph with n vertices is (n-1)! .

There are various approaches to solve TSP. Classical approach to solve TSP are dynamic programming, branch and bound which uses heuristic and exact method and results into exact solution. But as we know TSP is an NP-hard problem so the time complexity of these algorithms are of exponential time. So they can solve the small problem in optimal time but as compared to the large problem time taken by these algorithms are quite high. So no classical approach can solve

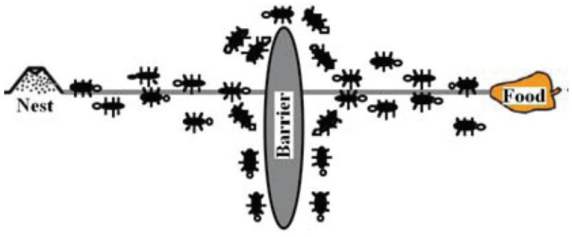
this type of problem in reasonable time as the size of the problem increases complexity increases exponentially.

So many alternate approaches are used to solve TSP which may not give you the exact solution but an optimal solution in reasonable time. Methods like nearest neighbor, spanning tree based on the greedy approach are efficiently used to solve such type of problems with small size. To overcome this different other approaches based on natural and population techniques such as genetic algorithm, stimulated annealing, bee colony optimization, particle swarm optimization etc. are inspired from these techniques.

4. ANT COLONY OPTIMIZATION

ANT colony optimization [5][6] technique introduced by Marco Dorigo in 1991 is based upon the real ant behavior in finding the shortest path between the nest and the food. They achieved this by indirect communication by a substance called pheromone which shows the trail of the ant. Ant uses heuristic information of its own knowledge the smell of the food and the decision of the path travelled by the other ants using the pheromone content on the path. The role of the pheromone is to guide other ants towards the food.

Ant has the capability of finding the food from their nest with the shortest path without having any visual clues. At a given point where there are more than one path to reach to their food then ants distribute themselves on different paths and the path and lay pheromone trace on that path and return with same path. Thus the path with minimum distance will acquire more pheromone as compared to other paths as the ants will return faster from that path comparative to the other path. So the new ants coming in the search of food will move with probability towards the path having higher pheromone content as compared to the path having lower pheromone content and in the end all the ants will move towards the same path with the minimum shortest path to their food. Now figure 1 shows the behavior of ants going from the upward direction will return early as compared to the ants going from the downward direction so the pheromone content in the upward direction is more as compared to the downward direction due to that in the end all the ants will start moving towards the upward direction which is the shortest path to their food.



Group Fig 1: shows the behavior of real ants.

ACO uses the set of artificial ants which co-operate each other to solve the problem and find the optimal solution of the problem. ACO can be used to solve combinatorial optimization problems such as Travelling Salesman Problem, Vehicle Routing, Quadratic Assignment, Graph Coloring, Project Scheduling, Multiple Knapsack etc. maximum of the problems are NP-hard problem [3] i.e. they take exponential time complexity in their worst case.

In the travelling salesman problem we are given with a set of cities and the distance between them. We have to found a shortest tour such that each city should be visited exactly once and then return to the stating city. Formally we can say that we have to found a minimal Hamiltonian circuit in a fully connected graph.

In ACO we stimulate no. of artificial ants on a graph where each vertex represents the city and the edge represents the connection between the two cities [7]. Pheromone is associated with each edge which shows the trace of the ant can be read and modified by the ants. It is an iterative algorithm where no. of ants is used to construct a solution from vertex to vertex without visiting any vertex more than once. At every vertex ant select the next vertex to be visited stochastically that is based upon the pheromone as well as the heuristic information available to it.

ACO algorithm
set parameters and pheromone value
    while termination condition not met do
    construct Ant solutions
    update pheromone
endwhile

in the above algorithm artificial ants will construct a solution. Ants start with an empty partial solution. At each iteration partial solution is modified by adding a set of components and updating the pheromone content. Creating a solution is completely based on a probabilistic stochastic mechanism. Updating pheromone value means increasing the pheromone content on the edges having good solution in order to find the optimal solution.

4.1 Ant System

Ant system was the first algorithm proposed under ACO to solve TSP problem [8]. In this algorithm all the ants update their pheromone values after completing a solution. In the construction of a solution an ant chooses next node to be visited using a stochastic mechanism. An ant k at city i has not visited set of cities S_p then P_{ij} be the probability to visit edge k after edge i.

$$P_{ij}^{k} = \begin{cases} \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{j\epsilon S_p}\tau_{ij}^{\alpha}\eta_{ij}^{\beta}} & if\ j\epsilon S_p \\ 0 \end{cases} \quad (1)$$

S_P represents the set of cities which has not been visited yet and to be visited again so that the probability of the ant visiting a city which has already visited becomes 0. Where τ_{ij} is the pheromone content on the edge joining node i to j . η_{ij} represents the heuristic value which is inverse of the distance between the city i to j, which is given by:

$$\eta_{ij} = \frac{1}{d_{ij}}$$

Where d_{ij} is the distance between the city i to j. α and β represents the dependency of probability on the pheromone content or the heuristic value respectively. Increasing the value of α and β may vary the convergence of ACO.

After solution construction we have to update the pheromone accordingly, as follows:

$$\tau_{ij} \leftarrow (1-\rho).\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

Where ρ is the evaporation rate, *m* is the number of ants, and $\Delta\tau_{ij}^{k}$ is the quantity of pheromone laid on edge(i,j) by an ant k:

$$\Delta\tau_{ij}^{k} = \begin{cases} {}^{Q}/_{L_k} & if\ ant\ k\ uses\ edge\ (i,j) in\ its\ tour, \\ 0 & otherwise \end{cases}$$

Where Q is a constant and L_k is the length of the tour constructed by an ant k.

5. PARALLEL IMPLEMENTATION OF ACO ON TSP

The main purpose of this section is to show parallel implementation of ant system for TSP. Biologically ants use parallel approach in search of their food. Ants perform task based parallelism to search their food. All the ants search their food parallel simultaneously and synchronize with the help of the pheromone content in the ground similarly we can use this approach in artificial ants in ACO [12]. Parallel model used in ACO is a master/worker paradigm. Where master controls the workers by communicating and capturing the global knowledge where as worker implements the search. In this model same copies of the ant system algorithm are simultaneously and randomly executed using different random source.

ACO is an iterative approach where at each iteration master shares the global knowledge of pheromone to its worker ants to construct a solution. When 1000 of ants perform the search operation then the solution construction becomes comparatively fast as compared to sequential implementations. Parallelism where large number of threads can be executed simultaneously can be done using GPGPU (general purpose graphical processing unit). GPU [9] consist of hierarchy of processing elements and their memory. An AMD GPU consist of more than one SIMD (single instruction multiple data) computation engine. Where each computation engine consists of multiple thread processor which executes same instruction all the time simultaneously but data items may vary. Where each thread processor have their own L1 cache. Each thread processor is a four or five way VLIW (very large instruction word) processor consisting of four or five ALUs respectively. Parallelism can be attained at both the level of thread processor and ALUs.

The two main steps of the ACO solution construction and pheromone updating are thoroughly discussed.

5.1 Solution construction Kernel

In solution construction task based parallelism approach is used as each ant performs their task independent of each other to find the best tour. As we have discussed earlier in this phase ants are allocated the source node randomly and they have to visit each node exactly once and have to reach back to their source node. On each iteration they have to choose their next node using probabilistic stochastic mechanism. This phase has inbuilt parallelism at the level of each ant as the biological ant find their tour. Each ant can be identified as a thread to construct the solution.

SOLUTION_CONSTRUCTION(weight_a, pheromone_p)
Source=get_global_id(0)
Length=0
Initialize all the nodes unvisited;
I=source
Starting from source while all the nodes visited do
    For every unvisited node j from the current node i
        Calculate the node with the max probability using (1)
    Visit the node j with max probability
    Length=length +weight_$a_{ij;}$
    Mark j as the node traversed
    Enter j in the tour
    i=j
Enter source node in to the tour
Length = length + $a_{j\ source}$

Solution construction kernel calculate the heuristic information to visit city j from the city i. computationally it is expensive to construct a solution a with a order of time complexity (n^2). However this kernel has memory related issues to maintain the ant memory for the tour constructed, visited node, weight matrix and pheromone matrix. This type of approach is basically suitable for the problems having large no. of cities. Maximum number of threads can be produced in this problem is equal to the no. of cities. Problem having less no. of cities will have less threads and less parallelism which leads to improper utilization of GPU resources.

5.2 Pheromone Update Kernel

It is a data parallelism approach where all the threads are performing the same task on different data sets. Pheromone update consist of pheromone evaporate and pheromone update. Pheromone evaporation can be done simultaneously to each edge parallel as data parallelism with respect to other edge as there is no relevance in evaporation. For pheromone evaporation we call N no. of thread for each row of our pheromone matrix and the pheromone is evaporated parallel. The work group for pheromone evaporation is of size N. Whereas pheromone is updated using a kernel where the tour of the ant is passed and the length of the tour so for every node the pheromone content is updated. Size of the workgroup for the pheromone update is also of size N.

Pheroemene_update(pheromone_p,length,tour)
Q=1/length
for(k=0;k<N;k++)
    i=tour[k]
    j=tour[k+1]
    *pheromone_p_{ij}=pheroemene_p_{ij}*Q*

However this kernel has synchronization related issues which can be handled using barriers such as pheromone can be updated by an ant after the construction of solution only, no ant can update the pheromone before construction of solution.

6. COMPARATIVE ANALYSIS AND PERFORMANCE EVALUATION

We implemented the algorithm sequential as well as parallel to check the speedup of the algorithm to find the solution. We have also parameterized all the parameters using different values so as to find the best parameters for our solution.

6.1 Comparison of different values for the parameters

ACO depends directly or indirectly on differne parameters such as α,β,ρ etc. these parameters affect the probability of the stochastic mechanism in finding the next node to be visited.

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. Acc. experiment we concluded that the value of α should be nearly equivalent to 1 as shown in figure.

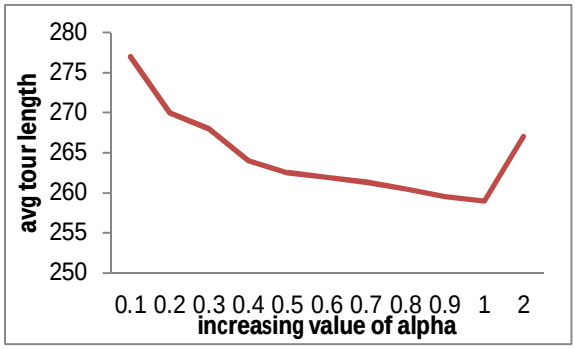


Fig 2 Graph showing the avg. tour length on increasing value of α.

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. Acc. to experiment we concluded that the value of α should be nearly equivalent to 5 as shown in figure as it is giving the best tour length.

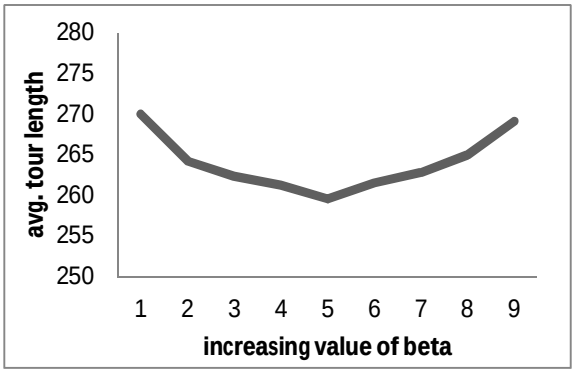


Fig 3 Graph showing the avg. tour length on increasing value of β.

ρ is the evaporation rate of the pheromone. Pheromone evaporation is necessary in ACO as if we will not evaporate the pheromone content than it may lead to the problem of stagnation. 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.

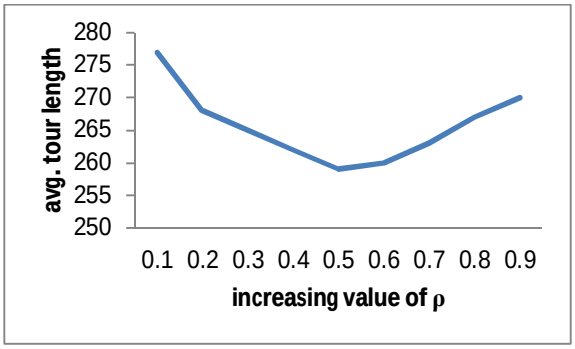


Fig 4 Graph showing the avg. tour length on increasing value of ρ.

6.2 Speedup Comparison

OpenCL parallel implementation on CPU and GPU are tested on the following hardware specifications:-:

AMD Radeon HD 6450(GPU): 2 Compute units, 625 MHz clock, 2048MB Global Mem., 32KB Local Mem., 256 work group size on a system having Intel Core i5 CPU 650 @ 3.2 GHz and 2048MB RAM with AMD APP SDK v2.8.

We have implemented ACO sequentially on the above given hardware specification with a randomly generated graph with different no. of nodes as well as some standard graphs to compare our results. Comparative analysis of the speed up of graph is shown with the sequential, CPU parallel and GPU parallel. In GPU parallel we have considered only the kernel execution time.

Fig 5 shows the speedup between sequential, CPU parallel and GPU parallel. In CPU parallel we have used OpenCL platform to rum the algorithm on CPU parallel. In GPU parallel same program is implemented on GPU. With respect to that we are able to achieve 3.11 times speed up in CPU and up to 7.21 times speed up in GPU.

7. CONCLUSION AND FUTURE WORK

All the parameters of ACO in ant system is been investigated to their best values as α=1, β=5 and ρ=0.5. parallel implementation is done on CPU and GPU using OpenCL. Where GPU parallelization has given best results with a speed up of . we will look for hybrid implementation of ACO on GPU with overcoming the limitation of GPU by utilizing the resources properly with data fragmentation and to optimize our algorithm to gain more speedup.

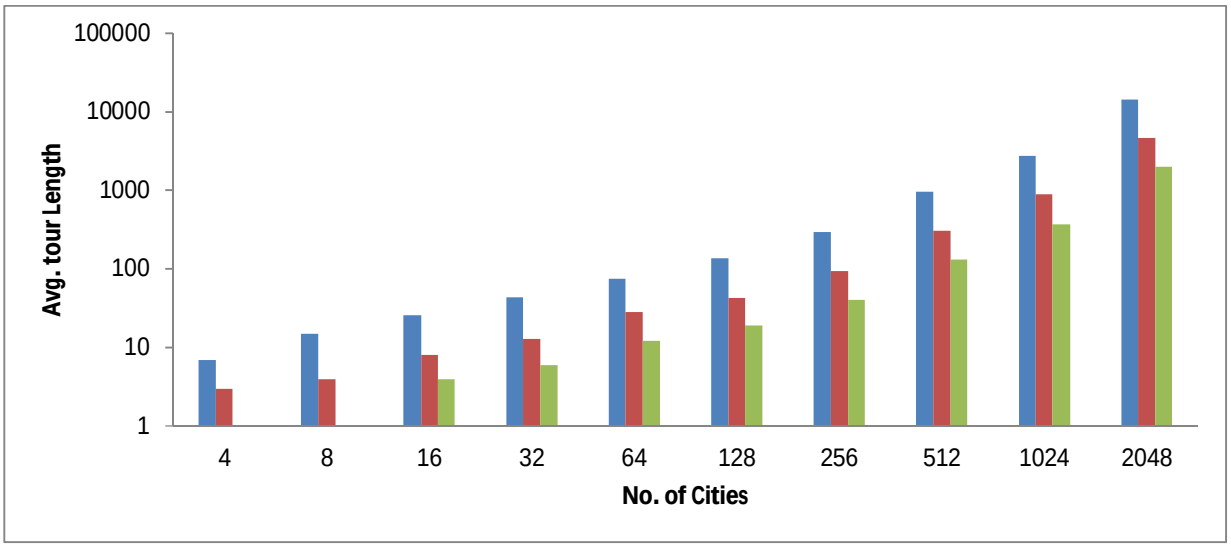


Fig 5 Shows the speed up between sequential, CPU parallel and GPU parallel

8. REFERENCES

[1] E.Lawler, J.Lenstra, A.Kan, and D.Shomsys Wiley New York, 1987 The Travelling Salesman Problem

[2] M.Dorigo and T.Stizzle : Bradford Company 2004. Ant Colony Optimization.

[3] C.Blum. Physics of life reviews, vol. 2, no.4, pp. 353-373, 2005. Ant colony optimization: Introduction and recent trends.

[4] Y-S. You. Genetic and Evolutionary computation, 2009. Parallel ant system for Travelling Salesman Problem.

[5] K.D. boese , A.B. Kahng, and S.Muddu .Operations Research letters ,16:101-113,1994. A new adaptive multistart technique for combinatorial global optimization.

[6] M. Dorigo. PhD thesis, Politecnico di Milano, 1992. Optimization, Learning, and Natural Algorithms

[7] T. Stizzle and H. H. Hoos. Future Generation Computer Systems, vol. 16, no8, pp. 889–914, 2000. MAX–MIN ant system

[8] M. Dorigo and T. Stizzle, A Bradford Book,2004. Ant Colony Optimization.

[9] Ying Zhang. PHD Thesis 2006. Performance and power comparisons between fermi and cypress GPUs.

[10] A. Munshi, B. R. Gaster, T.G. Mattson, J. Fung, D. Ginsburg, Addison-Wesley pub., 2011. OpenCL Programming Guide.

[11]] G. Reinelt, ORSA Journal on Computing, vol. 3, pp. 376–384, 1991. Tsplib–a traveling salesman problem library.

[12] M. Manfrin, M. Birattari, T. Stizzle, and M. Dorigo. 5th International Workshop on Ant Colony Optimization and Swarm Intelligence, vol. LNCS 4150. Springer-Verlag, 2006, pp. 224–234. Parallel ant colony optimization for the traveling salesman problem.