Ant Colony Optimization for Travelling Salesman Problem over CUDA with Saturation Point Termination

Chinmay Sudheendra Kulkarni
16BCE1019
Computer Science And Engineering
Vellore Institute of Technology
Chennai,India
chinss.kulk@gmail.com

Piyush Bamel
16BCE1221
Computer Science And Engineering
Vellore Institute of Technology
Chennai,India
piyushbamel1@gmail.com

Kaustubh Nagar
16BCE1338
Computer Science And Engineering
Vellore Institute of Technology
Chennai,India
nagarkaustubh98@gmail.com

***Abstract*—** Application of the Ant Colony Optimization algorithm (a probabilistic technique for solving computational problems) to solve the travelling salesman problem is discoursed further by appending general approach with saturation point termination. In ACO, the artificial ants incrementally build solutions by moving on the graph. The solution construction process is based by a pheromone model, that is, a set of parameters associated with graph components (either nodes or edges) whose values are modified at runtime by the artificial ants. And secondly, we shall use CUDA parallel computing platform to implement the same and analyze the variations in the time required for computation which includes the analyses of the computation time taken while changing the parameters included in mathematical model and arriving at the saturation point of iterations which helps in understanding the scalability of the algorithm. CUDA is NVIDIA's parallel computing architecture that enables dramatic increases in computing performance by harnessing the power of the GPU (graphics processing unit). It enables a very high level of task parallelism. By doing this project, we aspire to learn the importance of using probabilistic approaches to solve complex problems and enhance the understanding of parallel programming.

***Keywords*—** Ant Colony Optimization, Travelling Salesman Problem, CUDA.

I. INTRODUCTION

Travelling Salesman Problem (TSP) –The travelling salesman problem (TSP) which inquires- "In a list of given cities and the distances between each pair of cities, what is the shortest possible tour that visits each city and returns to the origin city?" It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science.

Ant Colony Optimization (ACO) -Ant colony optimization (ACO) is a population-based metaheuristic that can be used to find approximate solutions to difficult optimization problems. In ACO, a set of artificial ants search for good solutions to a given optimization problem. The artificial ants incrementally build solutions by moving on the graph. The solution construction process is stochastic and is biased by a pheromone model, that is, a set of parameters associated with graph components (edges) whose values are modified over the course of multiple iterations by the artificial ants. The probability to select city j by an ant k, sitting at city i is given by:

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}.[n_{ij}]^{\beta}}{\sum_{s \epsilon allowed_k} [\tau_{is}]^{\alpha}.[n_{is}]^{\beta}} & j \epsilon\ allowed_k \\ 0 & Otherwise \end{cases}$$

- Where τ_{ij} is the intensity of pheromone trail between cities i and j
- the α parameter to regulate the influence of τ_{ij}
- n_{ij} the visibility of city j from city i = $\frac{1}{d_{ij}}$ (d_{ij} is the distance between city i and j)
- β the parameter to regulate the influence of n_{ij}
- $Allowed_k$ the set of cities that have not been visited by k yet
- After each ant completes n iteration, the complete tour is obtained
- The pheromones are updated in such a way that shorter path gets more pheromone than longer path
- The pheromone update formula is given by

$$\tau_{ij}(t+1) = p.\tau_{ij}(t) + \Delta\tau_{ij}$$

$$\Delta\tau_{ij} = \sum_{k=1}^{l} \Delta\tau_{ij}^{k}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k} & if\ ant\ k\ travels\ on\ edge(i,j) \\ 0 & otherwise \end{cases}$$

- Where L_k the length of its tour
- T is the iteration counter
- $p \in [0,1]$ is the evaporation factor which regulates the reduction the pheromone.

- $\Delta\tau_{ij}$ the total increase of trail level on edge (i,j) by all ants
- $\Delta\tau_{ij}^{k}$ the increase of trail level on edge (i,j) caused by ant k

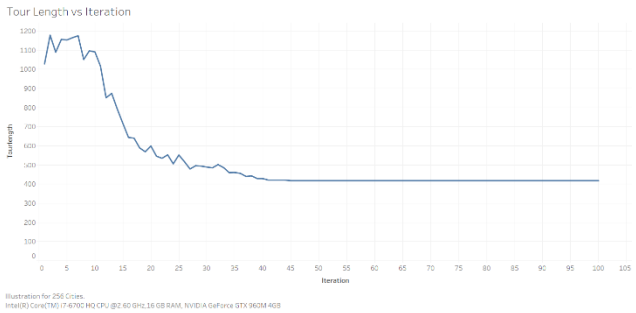
CUDA (Compute Unified Device Architecture) -CUDA is a parallel computing platform and application programming interface (API) model created by Nvidia. It allows software developers and software engineers to use a CUDA enabled graphics processing unit (GPU) for general purpose processing. This approach is termed GPGPU (General-Purpose computing on Graphics Processing Units). The CUDA platform is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements, for the execution of compute kernels. Function qualifiers- global, host and device are used to manage computations over GPU and CPU. CUDA API is used for device memory management. Kernel calls allow us to define grids, blocks and threads based on compute compatibility of hardware.

ALGORITHM

Procedure ACO Metaheuristic
Set parameters→pheromone trails to 1 for all paths
While (saturation point not reached) *do*
Calculate probability for each path using- - above stated mathematical model
Traverse path for each ant
Construct Ants Solutions
Evaporate pheromones
Update pheromones
End
End
We are finding saturation point after the iteration which produces consistent results following it for 'm' iterations.

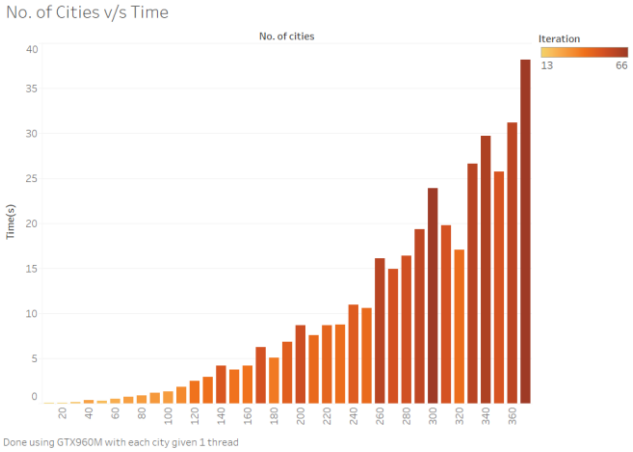
ANALYSIS

Fig.1



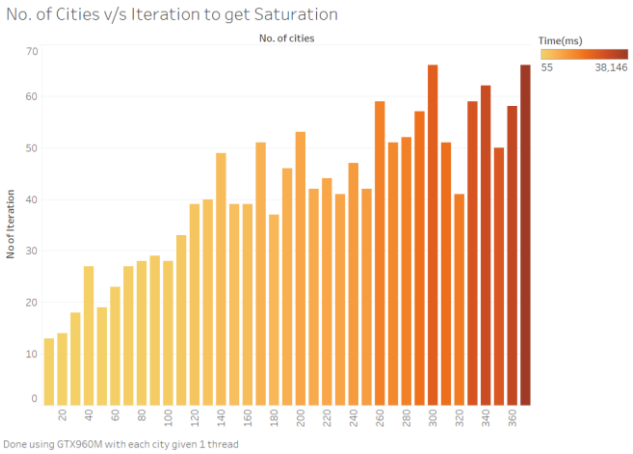
(Fig.1) As execution proceeds to next the iteration, pheromones are added to paths that yield lower tour cost. This leads to more probability of an ant to follow the path with lower cost. After a certain number of iterations (continuous cycle of evaporation and updation) we reach a saturation point where path with lower tour cost has enough concentration of pheromones that lead to prioritization of that path.

Fig.2



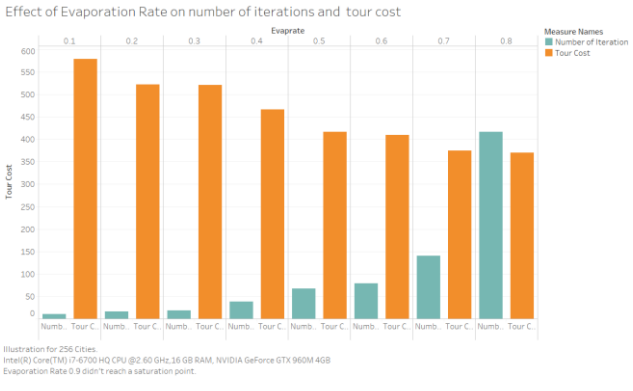
(Fig.2) As the number of cities increase, the time taken for Communication between host and device increases because, various data matrices(tour, choice, pheromone, probability, visited) have to be repeatedly communicated to the device. If the number of cities and ants is n, then the size of the matrices to be communicated shall be n^2.

Fig.3



(Fig.3) Even after we make substantial increase in the number of cities, the iteration in which we obtain the saturation of probability matrix is moderately increasing. Thus, making this approach viable for scalability.

Fig.4



(Fig.4) As we increase the evaporation rate, we get optimized results, but with increase in the iterations to reach saturation. By increasing the evaporation rate, we are equipping the ants with the

ability to explore newer paths by reducing the effect of the previous path traversed, which enables the ants to take a thorough tour of the complete map.

Conclusion

Ant Colony Optimization for TSP which involves heavy computing for a CPU, is easily worked out using GPGPU where each ant acts as a separate thread traversing parallely in each iteration, thus minimizing time taken per iteration by a large extent. Also, this approach is highly scalable as CPU would require a large set of resources and may be constrained due to lack of it. Using the saturation point as termination condition for iterations, the general approach can be modified where number of iterations for computing optimal solution can be hardcoded. This reduces the wastage of computation power.

References

[1] "CUDA C Programming Guide" https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

[2] "Ant Colony Optimization" http://www.scholarpedia.org/article/Ant_colony_optimization

[3] "Professional CUDA C Programming" http://www.hds.bme.hu/~fhegedus/C++/Professional%20CUDA%20C%20Programming.pdf