# Parallel Approach for Optimized Travelling Salesman Problem using GPU

Mita A. Landge
Department of Computer Engineering
Pimpri Chinchwad College of Engineering,
Pune, India.

K. Rajeswari, PhD
Department of Computer Engineering
Pimpri Chinchwad College of Engineering,
Pune, India.

## ABSTRACT
The Travelling Salesman Problem (TSP) is the most widely studied optimization problem used in many practical and real time applications. The TSP needs large computational power to be optimally solved by exact algorithms. In recent years, the increased development of general-purpose Graphics Processing Unit (GPUs) has led to huge improvement in decreasing the execution time of algorithm. An Optimization algorithm to solve Graphic TSP instance with parallel approach using GPU is proposed. The new approximation algorithm using GPU can be implemented to optimize the results upto 3/2 - $\mathcal{E}$ approximation ratio. This paper also enlists different approaches that have been proposed to solve various instances of TSP using GPU.

## General Terms
Parallel Computing, GPU Computing

## Keywords
Travelling Salesman Problem (TSP), Optimization Algorithms, Graphics Processing Units GPU, Approximation Algorithms.

## 1. INTRODUCTION
### 1.1.Travelling Salesman Problem
Travelling Salesman Problem is also referred in representing set of problem called combinatorial optimization problem [1]. In Traveling Salesman Problem a salesman has to visit all the cities exactly once and has to return back with the minimum cost length tour from all the possible tour included in that map to the starting city. Hence with n vertices there can be total (n-1)! number of possible tour in a graph. There are tremendous approaches to solve TSP and various instances of TSP. The best classical approaches to solve TSP are dynamic programming, branch and bound that uses exact and heuristic resulting to exact solution. But since, TSP is an NP-hard problem so the time complexity of these algorithms is of exponential time. Hence the small problem can be solved in optimal time but as compared to the problem with large instance take huge time to execute. The solution to this type of problem in reasonable time has No classical approach as the size of the problem increases complexity increases exponentially. Several alternate approaches are used to solve TSP which may not give the exact solution but an optimal solution in reasonable time. To solve such type of problems with small size methods based on the greedy approach like nearest neighbor, spanning tree is efficiently used. To overcome this different other approaches and techniques such as stimulated annealing, genetic algorithm, particle swarm optimization, bee colony optimization are based on natural and population techniques.

TSP is an NP-hard problem as it cannot be solved optimally in polynomial time [2]. Instead, one can hope to find an approximate solution, one that is not optimal, but is guaranteed to be within a small factor from the optimal solution. For a minimization problem to which a polynomial time algorithm is an approximation problem, if the output of the algorithm is within a factor of the optimum in the worst case.

### 1.2. GPU Accelerated TSP
The TSP needs huge computational power and takes huge time to solve. To solve such large problems single handedly it takes lots of time for a single processor. In the Era of Parallel computing the new paradigm to solve such type of problems is using General Purpose Graphical Processing Unit. This parallel programing solution GPUs are meant to do graphical processing such as arithmetic operations in the form of matrices also on the graphics. Hence we can speed up the computational time by utilizing GPUs processor to solve this problem. GPU consist of large no. of processors embedded together on a chip to perform a specific type of operations. OpenCL (OPEN Computing Language) is also an parallel programming framework used implementing programs that can be executed on heterogeneous platforms.

This paper presents a high performance GPU accelerated implementation algorithm for the Traveling Salesman Problem (TSP). The use of GPU significantly reduces execution time required for tour optimization; however it also requires a well-tuned and complicated implementation. The time required for local optimization comparing the graph edges grows significantly with the increase problem size.

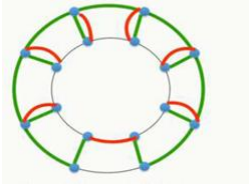
## 2. LITERATURE REVIEW
The Traveling Salesman Problem (TSP) considered as one of the most complex and optimization problem with numerous real-time applications. There is massive amount of literature on TSP solutions. Following are a few GPU accelerated solutions:

The Christofides 3/2 approximation for symmetric TSP was one of the first approximation algorithms proposed [2], it always find a solution that is at most 50% longer than optimum. Oveis Gharan, Saberi and Singh [1] conjectured that this algorithm has better approximation ratio than 3/2 but they could prove this only for the Graphic TSP, and only for a slight variant of this algorithm. Here, first a random solution is taken and then for building a neighborhood the pairwise exchange operations and evaluation of candidate solution is done. And results are putted back into fitness structure and then are copied back on CPU and repeat the steps select optimized solution. Here, in this approach high data transfer rate arises from CPU to GPU.

The traveling salesman problem using Ant Colony Optimization (ACO) [3], a meta-heuristic algorithm to discover, collaboratively, the shortest path between their nest and a food source by depositing pheromones along their traveled paths based on the natural ability of ants. ACO algorithms simulate the behavior of individual ants, which construct a solution and travel independently based on the pheromone trails that are left by the ants in previous iterations. Thus, this algorithm is highly parallelizable [6]. Many CUDA implementations of TSP solvers exist [5][6][12. There are several approaches till date proposed for GPU implementations of genetic algorithms (GAs) applied to TSP [7][8]. Heuristic algorithm initially generates a solution randomly and then attempt to improvise the results using heuristic techniques until it get locally optimal solution. O'Neil [17] also describe solution to TSP by evaluating parallel approach for implementation of iterative hill climbing with random restart for getting high quality solution.

Christofides's Algorithm[2]

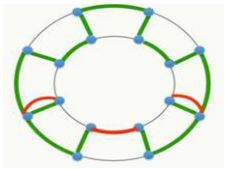
1. Choose a Minimum spanning tree T.
2. Add the minimum cost on odd degree veritces of T.



α= Cost of Computed Solution / Cost of The Optimum $\cong 1.5$

New Approximation Algorithm[1]

1. Choose a Random spanning tree T based on the Solution of LP relaxation
2. Add the minimum cost on odd degree vertices of T.



α= Cost of Computed Solution / Cost of The Optimum $\cong 1.25$

## 3. BACKGROUND

The Parallel approach for TSP using GPU for a graphic TSP instance can be given. The architecture shows the flow of problem solving instance using parallel optimization algorithms on GPU. Graphic TSP is a natural special case of TSP where we are given an underlying connected graph $G_0 = (V, E_0)$ and for all $u, v \in V, c(u,v)$ is the length of the shortest path that connects $u$ to $v$ [3]. Equivalently, we can reformulate graphic TSP as follows: we are given an weighted graph, and we want to find an Euclidian connected sub graph with the minimum number of edges. We recall that a graph is Euclidian if every vertex has an even degree. Similarly, one can also define graphic ATSP problem where $G_0$ is directed graph and $c(u,v)$ is the length of the shortest directed path from $u$ to $v$. The importance of graphic TSP is that all of the known hard instances of TSP are essentially instances of graphic TSP. So, it seems graphic TSP capture the main difficulty of the problem. Also, similar to TSP, graphic TSP is APX-hard, meaning that under the P = NP conjecture there is no PTAS for graphic TSP. The best known approximation algorithm for graphic TSP is also the 3/2 approximation algorithm of Christofides [2].

### 3.1. Parallel GPU Approach for Traveling Salesmen Problem (TSP)

For Parallel GPU approach to TSP we find minimum tour length with less amount of time where all vertices are visited at least once. In the TSP algorithm, thread allocation is based on thread control function because if threads are not used then data transfer rate and allocation of thread is waste hence only require numbers of threads to be created.

Parallel TSP work as follows:

**Input**: graph G(V,E)

**Output**: optimized tour length

**Begin**:

1. Choose an initial random solution.
2. To evaluate the initial solution using LSM initialization.
3. Allocate memory on GPU for problem input, solution, finesses structure and additional structure
4. Copy problem inputs, initial solution and additional structure on GPU memory.
5. Evaluate the neighbor candidate solution and insert the resulting fitness into fitness structure for each neighbor in parallel.
6. The solution selection strategy is applied to the fitness structure and new candidate solution has been selected.
7. Finding neighbor of new candidate solution and repeat step 6 and 7upto stopping criterion satisfied.
8. Copy chosen solution on CPU and display minimum tour length.

### 3.2.The GPU implementation is as

1. Copy the initial ordered coordinates to the GPU global memory.
2. Execute the kernel function.
3. Copy the ordered coordinates sets to the shared memory.
4. To calculate swapping effect of one pair of each coordinate.
5. To find the best value for the TSP instance and store it back to global memory.
6. Read the result from (CPU).

# 4. PROPOSED ARCHITECTURE

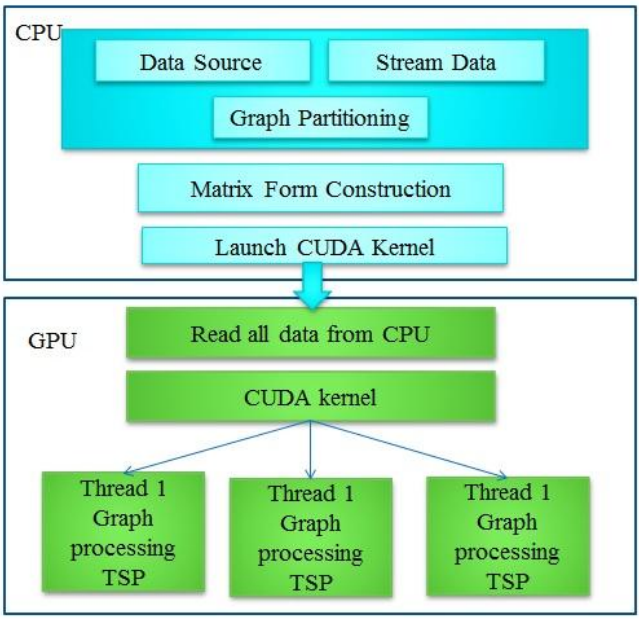



**Fig. 1 TSP-GPU Architecture**

Algorithm is very simple to describe: it chooses a random spanning tree based on the solution of the LP relaxation using the rounding by sampling method using GPU. Then it adds the minimum cost matching on the odd degree vertices of the tree.

Rounding by Sampling Algorithm for TSP [9]:

1. Compute an Optimum solution of LP
2. Write LP solution as a $\lambda$-uniform dist. Of spanning trees.
3. Sample a spanning tree T.
4. Add a minimum cost matching an odd degree vertices of T

# 5. MATHEMATICAL MODEL

## 5.1. Approximation Ratio For The Graphic TSP

1. Let (V, c) be an instance of the Symmetric TSP satisfying the triangle inequality.
2. Let x be an optimum solution of sub-tour elimination.
3. let ( V,S ) be a spanning tree picked at random according to the maximum entropy distribution $(P,S)S \in S$ with

$$\sum S \in S : e \in PS = \frac{n-1}{n} x_e \qquad (1)$$

For all $e \in E.$

4. Let $T_S$ be the set of odd degree vertices Of $(V,S)$.
5. Assign edge e good if e does not belong to any $T_S$-cut $\partial(U)$ with $x(\partial(U)) \le 2+10^{-15}$.

Then at least one of the following holds:

**a. there is a subset $E^*$ of edges with $x(E^*) \ge 10^{-12} n$ such that for each $e \in E^*$ the probability that $e$ is good is at least $10^{-24}$.**

b. there are at least $\frac{19}{20}n$ edges $e$ with $x_e \ge 1-10^{-7}$.

## 5.2.Minimum Distance Entropy Calculation

The weighted uniform distribution on spanning trees[1]

$$\max \sum_T -PT \log(PT) \qquad (2)$$

$$\sum_{T \ni e} PT = z_e \quad \forall e \in E,$$

$$PT \ge 0 \qquad \forall T.$$

z represents a fractional point inside the spanning tree polytope, and PT represents the probability of selecting an extreme point of the spanning tree polytope, which in this case is a spanning tree T.

## 5.3. Improved Approximation Algorithm for Graphic TSP [1]

1. Find an optimal solution $x^*$ of LP and let $z^* = (1-1/n)x^*$ be a fractional spanning tree and let G = (V,E) be the support graph of $x^*$.
2. if $x^*$ contains $(1-\varepsilon_2)n$ edges of fraction greater than $1-\gamma$ then
3. Let S = $\{e : x_e^* \ge 1-\gamma\}$ , and let $T^*$ be a minimum cost spanning tree with respect to the cost function c( ) among all trees T such that $|T \cap S|$ = rank(S).
4. else
5. Use minimum distance entropy using equation (2) to find weights $\lambda = E \to R_+$ such that the $\lambda$-random spanning tree distribution, $\mu$, approximates the marginal probabilities imposed by $z^*$, i.e., for all $e \in E$,

$$P_U[e \in T] \le (1+1/n^3) z_e^* \qquad (3)$$

6. Sample a tree $T^*$ from $\mu$
7. end if
8. Let O denotes the set of odd-degree vertices of $T^*$. Find the minimum cost O-join $J^*$.

Return: Shortcut multi-graph $J^* \cup T^*$ and output the resulting Hamiltonian cycle.

## 6. RESULT

Solution quality achieved by the GPU implementation for five 100-city inputs from TSPLIB[11]

**Table 1 : TSP optimal tour cost results**

| **TSPLIB Database** | | **CUDA GPU Solution Quality** | | |
|---|---|---|---|---|
| **Name** | **Optimal Cost** | **Min.Tour Cost** | **Min Tour #** | **Runtime (s)** |
| kroA100 | 21,282 | 21,282 | 33,188 | 2.540 |
| kroB100 | 22,141 | 22,141 | 5,969 | 2.499 |
| kroC100 | 20,749 | 20,749 | 23,092 | 2.543 |
| kroD100 | 21,294 | 21,294 | 32,142 | 2.499 |
| kroE100 | 22,068 | 22,084 | 16,941 | 2499 |
| | | 22,068 | 117,583 | 4.952 |

Table 1 addresses the solution quality and shows the number of the shortest tour and cost by the GPU implementation of approximation algorithm for five 100-city inputs from the TSPLIB library when using 100,000 random restarts. The the runtime for each input and the optimal tour cost [11]. The GPU code finds the optimal tour in all, on kroE200, where the tour is 0.08% longer. Doubling the number of iterations to 200,000 allows the GPU code to find the optimal tour in the last case as well.

## 7. CONCLUSION

This paper provides a parallel approach for graphic instance of Traveling Salesman Problem. The optimized approximation algorithms achieve parallelism using GPU implementation of approximation algorithm. Thus, the GPU accelerated approximation algorithm can be implemented to optimize the results upto 3/2 - $\mathcal{E}$ approximation ratio. The experimental result shows that algorithm gives an optimized tour length.

## 8. REFERENCES

[1] Shayan Oveis Gharan "New Approximation Algorithms for Traveling Salesman Problem"2013.

[2] Nicos Christofides. Worst case analysis of a new heuristic for the traveling salesman problem. Report 388, Graduate School of Industrial Administration, Carnegie-Mellon.

[3] Dorigo, M. 1992. Optimization, Learning and Natural Algorithms, Ph.D. thesis, Politecnico di Milano, Italy, 1992.

[4] Dorigo, M. and Gambardella, L.M. 1997. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation 1, 1 (Apr. 1997),

[5] Bai, H., OuYang, D., Li, X., He, L., and Yu, H. 2009. MAX-MIN,Ant System on GPU with CUDA. Proceedings of the Fourth International Conference on Innovative Computing, Information and Control (Dec. 2009), 801-804.

[6] Cecilia, J.M., García, J.M., Nisbet, A., Amos, M., and Ujaldón, M.2013. Enhancing Data Parallelism for Ant Colony Optimization on GPUs. J. Parallel Distrib. Comput. 73, 1 (Jan. 2013), 42-51.

[7] Chen, S., Davis, S., Jiang, H., and Novobilski, A 2011.CUDABased Genetic Algorithm on Traveling Salesman Problem. Computer and Information Science 2011, R. Lee, Ed. Springer Berlin, Heidelberg.

[8] Fujimoto, N. and Tsutsui, S. 2010. A Highly-Parallel TSP Solver for a GPU Computing Platform. Proceedings of the Seventh International Conference on Numerical Methods and Applications (Aug. 2010), 264-271.

[9] Shayan Oveis Gharan, "A Randomized Rounding Approach to the Traveling Salesman Problem"2013.

[10] Molly A. O'Neil, "Rethinking the Parallelization of Random - Restart Hill Climbing A Case Study in Optimizing a 2- Opt TSP Solver for GPU Execution", GPGPU-15 , February 07 2015, San Francisco, CA, USA

[11] Molly A. O'Neil, "A Parallel GPU Version of the Traveling Salesman Problem", February 07 2015, San Francisco, CA, USA.

[12] Murilo Zangari de Souza, A GPU Implementation of MOEA/D-ACO for the Multiobjective Traveling Salesman Problem, 2014 Brazilian Conference on Intelligent Systems.