Parallel Ant System for Traveling Salesman Problem on GPUs

Ying-Shiuan You
Taiwan Evolutionary Intelligence Laboratory (TEIL)
Department of Electrical Engineering
National Taiwan University
No.1, Sec.4, Roosevelt Rd., Taipei, Taiwan
r97921039@ntu.edu.tw

1 Introduction

Ant Colony Optimization(ACO) is a meta-heuristic introduced in 1991 by Dorigo *et al.* on TSP problem(Dorigo, 1992). This alorithm is inspired by the natural behavior of real ants. Ants usually communicate via pheromone trail, i.e. an ant would lay down some mount of pheromone on the passed path. An ant's tendency to choose a specific path is positively correlated to the intensity of trail. The pheromone trail evaporates over time, if on pheromone laid down by other ants. If many ants lay down pheromone on specific path, the intensity would attract more ants forward this path.

Although ACO has outstanding performance on TSP problem, it spends huge execution time in large scale TSP problem. However, ACO has highly parallelizable structure(Talbi, Roux, Fonlupt, & Robillard, 1999; Stützle, 1998). In this work, we choose NVIDIA's CUDA programming model and Tesla C1060 as platform to implement our Parallel ACO.

2 Ant Colony Optimization

While adapting to the computer system which is operated in discrete time domain, there are some difference between an artificial ant and a real ant. An artificial ant would not visit same location in their travel and ACO needs to track every artificial ant's path in order to find better paths. As the result, artificial ant uses tabu list to record its path, i.e. an artificial ant has memory. To enhanced the efficiency, the ACO adds local search procedure when ants decide next location, therefore an artificial ant would not be completely blind.

The simplified flow of ACO is as follow:

Step 1: Set counter and terminating condition.

Step 2: Reset Tabu list.

Step 3: Travel by probability equation.

Step 4: Evaluate tour length by Tabu list.

Step 5: Estimate trail intensity by intensity formula.

Step 6: Check for termination, if not return to step 2.

3 Parallel Implementation of ACO on CUDA

Reviewing the algorithm above, ACO has a natural feature for SIMD parallel computation, i.e. the probability computation for next location of each ant is independent in one travel because

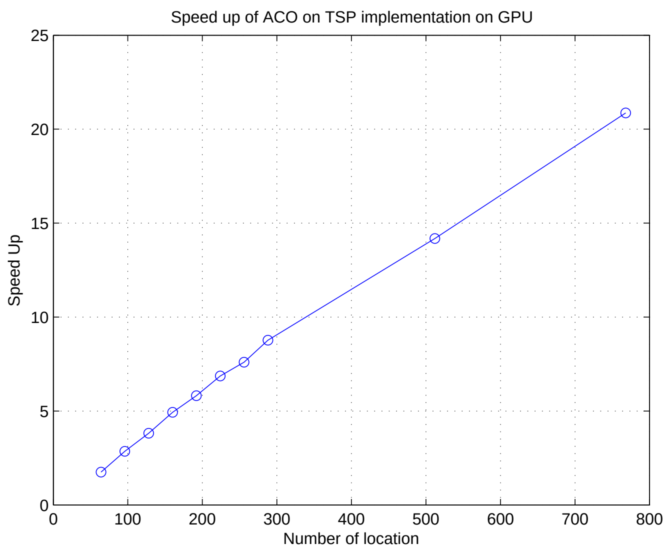

Figure 1: Speed-up of running time of ACO on TSP implemeting on CPU and GPU

ants communicate only via pheromone trail. Pheromone trail updating and path evaluation are implemented after ants complete their own travel. The basic model of parallel ACO is: let each ant travels simultaneously in one cycle. On CUDA, the travel is putted into the kernel, taking the advantage of GPU computing power. Each thread is responsible for a complete travel of single ant. When a new cycle starts, the kernel will be invoked.

When implementing on CUDA, the memory usage should be considered carefully, or would bring a large impact on performance due to large access latency difference between memory architectures. In this work, the memory arrangement of GPU is as follows:

1. For frequently accessed data and can only be accessed by single ant, such as the legal flag of each location, is stored on share memory in order to take the advantage of fast accessing speed.

2. For write-once data, such as path information in tabu list, is stored on global memory.

3. For read-only data, such as pheromone trail and distance between cities, is stored on texture cache because the smaller latency than global memory.

4 Experiment Result

The device used in this work is NVIDIA Tesla C1060. The version of CUDA version is 1.3. The operating system is Linux 64-bit. The compiler of ANSI-C is gcc with version 4.1.3, and the compiler of kernel code is nvcc with version 2.1. TSP is used as test problem in this work. The number of ants is equal to the number of cities. Each ant starts from different city.

Figure 1 which contains the performance speed-up when travel of each ant is running on GPU. The improvement of running time grows as the complexity of problem grows.

References

Dorigo, M. (1992). *Optimization, learning and natural algorithm.* Doctoral dissertation, Dip. Elettronica e Informazione, Politecnico di Milano, Italy.

Stützle, T. (1998). Parallelization strategies for ant colony optimization. Springer-Verlag.

Talbi, E.-G., Roux, O., Fonlupt, C., & Robillard, D. (1999). Parallel ant colonies for combinatorial optimization problems. In *Proceedings of the 11 IPPS/SPDP'99 Workshops Held in Conjunction with the 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing* (pp. 239–247). London, UK: Springer-Verlag.