Parallel Genetic Algorithm for Traveling Salesman Problem on Graphic Processing Unit

Fitriyani
School of Computing Telkom University, Bandung, Indonesia

Erqy Sara Quartian
School of Computing Telkom University, Bandung, Indonesia

Yuliant Sibaroni
School of Computing Telkom University, Bandung, Indonesia

Fhira Nhita
School of Computing Telkom University, Bandung, Indonesia

Abstract
Travelling Salesman Problem (TSP) Algorithm is classified as high complexity case that frequently occurred in real life. On the other hand, Genetic Algorithm is classified as a robust algorithm to solve complex cases. While having the ability to solve complex cases, Genetic Algorithm has high time complexity. Parallel computation can be applied to increase the performance of a processing unit. In this research, the TSP case will be solved by using the Genetic Algorithm on Graphical Processing Unit (GPU). The experimental result showed that how a genetic algorithm needs lower execution time, larger fitness and optimal mileage. The speedup is 2.65, the shortest distance of parallel implementation is 4739.34, better than serial implementation, and the best fitness is not much different respectively 0.000210 and 0.000211 for serial and parallel.

Keywords: parallel computing, genetic algorithm, TSP, CUDA
1 Introduction

Nowadays, the needs of high performance computing are increasingly fast. Most of the computational problems need an excessive amount of computing time, which is inversely proportional with the need to be fast. There are several cases that require the support of high-performance system, for example: combinatorial problem (NP-complete), complex modeling and simulation both micro (molecular, nano material, dynamic fluid, etc) and macro scale, a very detailed animations with special effects, weather forecast systems that covers whole regions of the world, etc. The servers of some internet giants like Google and Yahoo also requires the support of a high-performance systems.

Travelling Salesman Problem (TSP) is considered as a problem with high complexity or NP-complete. The main problem of TSP is to find the shortest route from a collection of city, visiting a city exactly once and return to the initial city. Until now, TSP is still considered as an interesting study in order to find the fastest time and the best result.

The utilization of GPU in high-complexity computational problem offer an interesting prospect given the price of GPU is relatively inexpensive while the numbers of the processors are relatively high. With this condition, the utilization of GPU in computation becomes very feasible to be developed in the future. In this research, the TSP case will be solved by implementing the Genetic Algorithm on GPU with CUDA architecture.

2 Related Work

In its early emergence, the GPU is known as a graphics processor. But the numbers of processor cores in it attract the researchers to implement it data computing. A number of studies have been carried are related to GPU utilization in numeric data computing. Lefonn et al. developed a library to access the GPU data structures in generic and efficient way [13]. Mendez-Lojo et al. use GPU to run irregular algorithms that operate on pointer-based data structures as graphs[14], resulting an average speedup of 7x compared to a sequential CPU implementation and outperforms a parallel implementation of the same algorithm running on 16 CPU cores.

The TSP case is frequently studied by the researchers of Yang and Nygard to observe the impact of genetic algorithm initial population in order to solve the TSP case by using the approach of time windows by using regular CPU[15]. Some researchers also investigate the TSP case in mathematical pint of view. Bartal et al. shown the algorithmic tractability ofmetric TSP depends on the dimensionality of the space and not on its specific geometry [16]. In another TSP study, Fekete et al. can solve the Fermat-Weber-Problem (FWP) with high accuracy in order to find a good heuristic solution for the MWMP [17]. Various algorithms to solve the TSP case can also be applied the way Ant Algorithm is utilized [5].
Parallel genetic algorithm for traveling salesman

Genetic Algorithm is also known of its robustness in solving high-complexity problems. Some of the application of the Genetic Algorithm can be seen in weather forecasting [6, 7] and the prediction dengue fever dissemination.

3 Parallel Genetic Algorithm for TSP

TSP is a high-complexity problem or NP-Complete. The idea of TSP is to find the shortest route from numbers of city, visiting a city exactly once and return to the initial city. When the numbers of cities are very large, the time complexity of finding each cost of the route is very high.

Genetic Algorithms produce a solution or a model in the form of chromosomes or an individual containing genes. The evolution process in Genetic Algorithms starts the initialization of population or the determination of individual representation. Then each chromosome will be decoded. After that, each chromosomes will be evaluated and each parents will be determined. The parents will produce child from crossover process and mutation. The produced child will be selected to replace the parents’ chromosomes in the next generation. In Genetic Algorithms, elitism process will be carried in order to maintain the best individual that has the best fitness value so that the individual will be included in to the next generation.

a. Individual Representation
   In Genetic Algorithms (GA), biner, real or integer representation can be used to represent individual/chromosomes. The representation that is used in TSP is permutation representation, where the value of each gene has to be different from each other because each gene describes each city that has been visited.

b. Individual Evaluation
   Individual evaluation used invers fitness below:

   \[ f = \frac{1}{h+a} \]

   Where:
   \( f \) = the fitness function
   \( h \) = sum of path length
   \( a \) = the smallest value

c. Parents Selection
   Parents selection or the selection of two chromosomes in GA depends on fitness value and uses Roulette-Wheel algorithms where the chromosomes occupy a portion of the circle corresponding to the value of fitness. The higher the value of fitness of a chromosome, the portion of the circle that it occupied will also be larger, so that the chromosome will be larger and will be selected as parents.

d. Cross-over
   The purpose of the cross-over is to produce child from selected parents. Each
parent will produce two child. Order crossover is used in TSP to prevent one city visited more than once.

![Figure 1 Order Cross-over example](image)

The probability of cross-over in GA is determined by the Probability of cross-over ($P_c$). The interval of the $P_c$ that often used is 0.6 to 0.9[12]. The random value of $P_c$ is smaller than determined $P_c$ resulting cross-over in the selected point.

e. Mutation

In GA, the mutation is independent. Mutation in a gene will not induce a mutation to another gene. As well as cross-over, mutation is also determined by probability of mutation ($P_m$). A random value will be generated for each gene in the chromosomes. If the value of each gene is lower than $P_m$ or higher than $P_m$, or between intervals [0..1], the gene that initially has the value of 0 will be changed to 1 and the other way around. On the other hand, a gene with the value of real numbers will get another random value.

f. Survivor Selection

After going through an evaluation process by using feed forward algorithms that will produce the best individual, and parents selection process by using roulette-wheel that will produce the parents, each pair parents will have to undergo recombination and producing two child. The individual that were produced through recombination will be going through mutation. The amount of individual that were produced after the mutation took place will be the same amount of the amount of initial population. The individuals will replace the previous generation.

Genetic Algorithm is very effective to solve many problems. But this algorithm needs an excessive amount of execution time because in the same time every solution candidate has to be evaluated to the point it reach the desired performance standard [11]. In the aspect of transportation, genetic algorithm has been shown to resolve various problems that are considered complex. [1, 12]. In his research, Chakroborty used genetic algorithms to resolve an Optimal Routing and Scheduling problem in the field of transportation. In the other research, Vidall et al. (2011) used a hybrid genetic algorithm to solve multi depot and periodic vehicle routing problem with a satisfaction result. Genetic algorithm have also been used in parallel Pospichal et al. (2011) to parallelize the genetic algorithm in GPU environment using CUDA library. Thera are several model in parallelizing GA, among others.
4 Experiment and Result

Parallel Genetic Algorithms for TSP are implemented in GPU Geforce GTX 770, Cuda 7.0 with the compute capability 3.0. TSP data that were being used has been taken from city TSPLIB 101, by replacing the following parameter:

a. Maximum Generation: 100
b. Population size: 100, 150, 200 and 250
c. The probability used in cross-over phase: 0.5, 0.7, and 0.9
d. The probability used in mutation phase: 0.5, 0.3, and 0.1

Parallel implementation carried out in individual evaluation phase

By observing the parameters above, will be obtained serial execution results as can be seen in Table 1

Table 1. Serial Results

<table>
<thead>
<tr>
<th>Population Size</th>
<th>Crossover Probability</th>
<th>Mutation Probability</th>
<th>Path length</th>
<th>Fitness</th>
<th>Fitness Average</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>0.9</td>
<td>0.5</td>
<td>4852.61</td>
<td>0.000206</td>
<td>0.000194</td>
<td>2.78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.3</td>
<td>4833.26</td>
<td>0.000207</td>
<td>0.000187</td>
<td>2.74</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1</td>
<td>4836.49</td>
<td>0.000207</td>
<td>0.000191</td>
<td>2.77</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>0.5</td>
<td>4856.81</td>
<td>0.000206</td>
<td>0.000193</td>
<td>2.72</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.3</td>
<td>4759.60</td>
<td>0.000210</td>
<td>0.000194</td>
<td>2.76</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1</td>
<td>4902.11</td>
<td>0.000204</td>
<td>0.000185</td>
<td>2.89</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.5</td>
<td>4847.36</td>
<td>0.000207</td>
<td>0.000191</td>
<td>2.82</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.3</td>
<td>4794.60</td>
<td>0.000209</td>
<td>0.000195</td>
<td>2.68</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1</td>
<td>4858.35</td>
<td>0.000206</td>
<td>0.000188</td>
<td>2.91</td>
</tr>
</tbody>
</table>

From the Table 1 it could conclude that the best generation can be found in the population size 200, $P_c = 0.7$ and $P_m = 0.3$, the shortest distance of 4759.60 and the best fitness of 0.000210 with execution time of 2.76 seconds. The parallel execution can be seen in the following table.

Table 2. Parallel Results

<table>
<thead>
<tr>
<th>Population Size</th>
<th>Crossover Probability</th>
<th>Mutation Probability</th>
<th>Path length</th>
<th>Fitness</th>
<th>Fitness Average</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>150</td>
<td>0.9</td>
<td>0.5</td>
<td>4739.34</td>
<td>0.000211</td>
<td>0.000187</td>
<td>1.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.3</td>
<td>4797.78</td>
<td>0.000199</td>
<td>0.000195</td>
<td>1.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1</td>
<td>4917.11</td>
<td>0.000205</td>
<td>0.000197</td>
<td>0.72</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>0.5</td>
<td>4889.35</td>
<td>0.000206</td>
<td>0.000193</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.3</td>
<td>4822.92</td>
<td>0.000204</td>
<td>0.000194</td>
<td>1.19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1</td>
<td>4852.58</td>
<td>0.000208</td>
<td>0.000193</td>
<td>1.03</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.5</td>
<td>4795.98</td>
<td>0.000210</td>
<td>0.000189</td>
<td>1.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.3</td>
<td>4873.57</td>
<td>0.000207</td>
<td>0.000197</td>
<td>1.20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1</td>
<td>4864.20</td>
<td>0.000207</td>
<td>0.000192</td>
<td>1.02</td>
</tr>
</tbody>
</table>
In Table 2, it can be obtained that the shortest distance for the best generation can be found in population 150, \( P_c = 0.9 \) and \( P_m = 0.5 \), shortest distance of 4739.34 and best fitness of 0.000211 with execution time of 1.04 seconds.

![Figure 2 Execution Time Serial vs Parallel](image)

Based on the results of serial and parallel implementation shown that the execution time required for parallel implementation is lower than execution time for serial implementation by speedup 2.65, see Figure 2 above. Speed up depends on the design of parallel algorithms and parallel proportion in an algorithm. Speedup result in this experiment show that the parallel algorithm design is good enough. Speed up can be improved with better algorithms design and enlarge parallel proportion in an algorithm. And the shortest distance of parallel implementation is 4739.34, better than serial implementation. But the best fitness is not much different respectively 0.000210 and 0.000211, see Table 3 below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Serial</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>200</td>
<td>150</td>
</tr>
<tr>
<td>( P_c ) (crossover)</td>
<td>0.7</td>
<td>0.9</td>
</tr>
<tr>
<td>( P_m ) (mutation)</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>the shortest distance</td>
<td>4759.60</td>
<td>4739.34</td>
</tr>
<tr>
<td>the best fitness</td>
<td>0.000210</td>
<td>0.000211</td>
</tr>
</tbody>
</table>

5 Conclusion

The result of the observation in the algorithm implementation in serial and parallel shows that the higher the fitness value, the shorter the mileage of the TSP. Probability of cross-over parameter will also affect the result obtained, because the function of the cross-over is to cross-over between both parents to produce child that are better than the parents. The lower the probabilities, the more likely
the child would not develop into a better individual in the next iteration.

The result of the parallel and serial execution shows that the implementation of parallel resulting less computing time and more optimal distance.

References


Received: November 15, 2015; Published: August 29, 2016