2009 Sixth International Conference on Computer Graphics, Imaging and Visualization

Implementation of Ant Colony Algorithm based on GPU

Wang Jiening
Air Traffic Management Research Base
Civil Aviation University of China
Tianjin, China
wang_jiening@yahoo.com.cn

Dong Jiankang
Air Traffic Management Research Base
Civil Aviation University of China
Tianjin, China
jkdong@cauc.edu.cn

Zhang Chunfeng
Air Traffic Management Research Base
Civil Aviation University of China
Tianjin, China
cfZhang@cauc.edu.cn

***Abstract*—Ant colony algorithm is an efficient intelligent algorithm to solve NP hard problem. This paper presents a parallel computing solution based on General Purpose GPU (GPGPU) to solve Traveling Salesman Problem (TSP) with Max-Min Ant System (MMAS). The experimental result shows it is more efficient than pure CPU computing.**

Keywords-Ant colony algorithm; Max-Min Ant System (MMAS); General Purpose GPU (GPGPU); Traveling Salesman Problem (TSP)

I. Introduction

Ant algorithm was an intelligent algorithm as a multi-agent approach to difficult combinatorial optimization problems like the traveling salesman problem (TSP) and the quadratic assignment problem (QAP) [1]. This algorithm was inspired by the observation of real ant colonies, which can find shortest paths between food source and their nest. According to the natural phenomena one can mimic ant's behavior to construct an ant colony optimization which is a meta-heuristic colony of artificial ants cooperate in finding good solutions to difficult discrete optimization problems. Artificial ants have two good features, which are parallel mechanisms based on positive feedback and distributed calculation. It is therefore reasonable to give artificial ants some capabilities that make them more effective and efficient in using parallel calculation resources such as GPU.

With in-depth studies on ant colony optimization, some improved schemes have been suggested which enriched with extra capabilities like lookahead, local optimization, backtracking, and so on. The Ant Colony System (ACS) algorithm introduced by Dorigo and Cambardella [2] was able to find good solutions within a reasonable time only for small problems. Further more, Stutzle and Hoos proposed Max-Min Ant System (MMAS) which was the same as ant system but improved pheromone trail updating features [3]. This helped avoiding stagnation, which was one of the reasons that made ant system poor performance when an elitist strategy. In this paper, a novel MMAS implementation method using modern GPU resource was proposed. When applied to the TSP, the experimental results showed it was more efficient than only use CPU.

II. Algorithm

In this paper, the ACS was applied to solve the typical combinatorial optimization problem—TSP using modern GPU parallel calculation features. This section will give brief introduction about the algorithm.

A. Traveling Salesman Problem (TSP)

TSP is one of the most studied NP-hard problems in combinatorial optimization. Consider a set $V=\{v_1,v_2,...,v_n\}$ of nodes, representing cities, and a set of E of arcs fully connecting the different nodes V. Let d be the weight value of arc $(i,j)\in E$, usually d_{ij} or d_{ji} is the distance between cities i and j, with $(i,j)\in V$. The TSP is the problem of finding a minimal length Hamiltonian circuit on the graph $G=\{V,E,d\}$, where an Hamiltonian circuit of G is a closed tour visiting once and only once all the nodes of G, and its length is given by the sum of the lengths of all the arcs of which it is composed. The TSP is a typical NP-hard problem.

B. Ant Colony System

In ACS artificial ants make combinatorial optimization solutions of the TSP by moving on the problem graph from one city to another. The system can be described as follow:

Consider an ants' set Ω, there are *m* ants build a tour executing *n* steps in which a probabilistic decision rule is applied. At the beginning, these ants are located on a defined city randomly and let them to find their way independently. In practice, when in city the $k_{th}(k=1...m)$ artificial ant chooses the city *j* to move to according to its bionic features that ants deposit pheromone while building a solution or after they have built a complete tour. That is the k_{th} ant movement is determined by the amount of pheromone on the path. Now we defined some feature values to control the ant moving behavior, η_{ij} is the heuristic value of move from city *i* to city *j* (generally $\eta_{ij} = 1/d_{ij}$ where d_{ij} is the distance between *i* and *j*), τ_{ij} represents the amount of pheromone trail on *path(i,j)* at time *t*, α and β are two parameters to control the relative weight of pheromone trail and heuristic value. Therefore the desire moving probability of k_{th} ant from city *i* to *j* at time *t* according to the local pheromone trail values and the local heuristic values can be represented as follows:

$$p_{ij,k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum\limits_{s \in allow_k}[\tau_{is}(t)]^{\alpha}[\eta_{is}]^{\beta}} & j \in allow_k \\ 0 & j \notin allow_k \end{cases} \quad (1)$$

Where $allow_k = \{V - tabu_k\}$ is the set of cities in the neighborhood of city *i* that k_{th} ant can visit at time *t*, $tabu_k$ is a city list that the ant have already visited.

In ACS one of the important strategies is how to update the on pheromone the path to avoid redundant pheromone overcoming heuristic value. After all ants have completed their tour, pheromone evaporation on all paths is triggered, and then each k_{th} ant deposits a quantity of pheromone on each path it has used. Generally the pheromone updating equations applied to all the paths are:

$$\tau_{\mathrm{ij}}(t+n) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \quad (2)$$

$$\Delta\tau_{\mathrm{ij}}(t) = \sum_{k=1}^{m}\Delta\tau_{ij,k}(t) \quad (3)$$

Where $\rho \in [0,1)$ is the pheromone trail decay coefficient, *1-p* is the residual coefficient of pheromone, $\Delta\tau_{\mathrm{ij}}(t)$ is the increased amount of pheromone after one loop, $\Delta\tau_{ij,k}(t)$ is the deposits of pheromone that k_{th} ant left after one loop. Here

$$\Delta\tau_{ij,k}(t) = \begin{cases} Q/L_k & \textit{if the } k_{th} \textit{ ant pass through the path (i,j)} \\ 0 & \textit{otherwise} \end{cases}$$

Where L_k is the length of the path the k_{th} ant toured, *Q* is a defined constant.

C. Ant Colony System

MMAS is an improved ant system algorithm. Its improved features are (1) the path which were used by the best ant in the current iteration received additional pheromone, (2) pheromone trail values are restricted to the interval $\tau_{\mathrm{ij}}(t) \in [\tau_{\min}, \tau_{\max}]$, and (3) trails are initialized to their maximum value $\tau_{\max}$. This helps avoiding stagnation, which may occur in case some pheromone trails are close to $\tau_{\max}$ while most others are close to $\tau_{\min}$; that is, pheromone trails are update using a proportional mechanism. Following rule is applied to update pheromone values:

$$\tau_{\mathrm{ij}}(t+n) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}^{\min} \quad (4)$$

$$\Delta\tau_{\mathrm{ij}}^{\min} = Q/L, L = \min(L_k), k = 1,2,...,m \quad (5)$$

III. Implement of MMAS based on GUP

The GPU has become an integral part of today's mainstream computing system. It is not only a powerful graphics engine but also a highly-parallel programmable processor featuring peak arithmetic and memory bandwidth that substantially outpaces its CPU counterpart. A large fraction of the GPU's resources exist within programmable processing cores responsible for executing shader functions. GPUs maintain high efficiency through the use of multicore designs that employ both hardware multithreading and SIMD processing. The transform and lighting computation integrate into one step in which position, color and lighting of scene objects are determined by vertex program written in a custom shader program. Similarly, the texturing and color composing stages are also collapse into a single abstract stage where the outputs are determined by shader programs [4]. Figure 1 shows the simple pipeline of computing process using GUP resource.

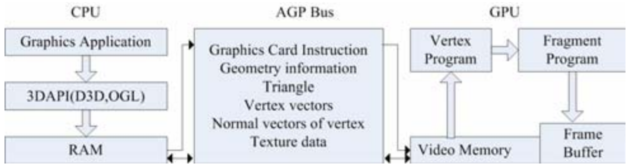

Figure 1. Simple pipeline of computing process with GPU.

A. Ideas mapping MMAS to the GPU

The main idea of the novel implementation mapping MMAS to GUP is to make full use of parallel process feature and positive feedback mechanism of the MMAS algorithm. By applying the parallel processing techniques of fragment processor based on the SIMD, one can improve MMAS computing efficiency with GUP's accelerated features. Nowadays, GPU is still deficient on branch and iterative feedback processing, so the GPU computing implements were mainly focus on the calculation flow (equation 1) mentioned in section 2.2.

B. Implement MMAS on GPU

According to section 2 the MMAS includes following computing steps:

- Process for initialization, which includes initial loop count, $\tau_{\min}$ value, $\tau_{\max}$ value, $\tau_{ij} = \tau_{\max}$, $\Delta\tau_{ij} = 0$, heuristic value η_{ij} and $tabu_k = NULL$;

- Process for ant moving, which includes to compute $p_{ij,k}(t)$, to move to next city according to the probability, to update the $tabu_k$;
- Process to update path pheromone, which includes to get the nearest path of the current iteration, to compute $\Delta\tau_{ij}^{\min}$ and τ_{ij}, to update path pheromone.

In this paper, a MMAS calculation flow based on the GPU's SIMD processing feature was proposed (figure 2). Obviously, to make full use of GUP resource, three techniques are applied which are texture construction base on OpenGL, Frame Buffer Object (FBO) technique and Render to Texture technique.

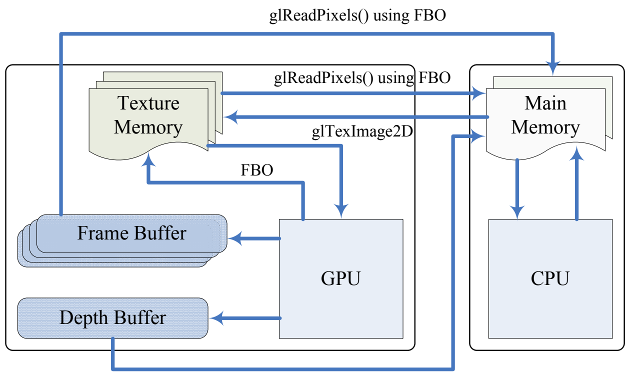

Figure 2. Diagram illustrate GPU computing.

As mentioned before, the computing bottle neck of MMAS is the process for an ant to move from one city to another. We realized the process using GPU resource. Firstly, at the initial process besides assignment of initial parameters, the GUP computing environment based on OpenGL was setup which included OpenGL extension parameters, FBO texture parameters etc. Additionally, there were some kind of FBO textures including tabu list texture, distance between cities texture, path pheromone trail values' texture, initial ant location texture etc. The size of these textures were MAXCITY*MAXANTS. The data format of these textures was RGBA. These textures were bound to GPU processing by Cg Fragment Program API. Next process was ants moving, that was using Cg fragment program to perform parallel computing of moving to next city actions. Considering the features of GUP's pipeline, here statement step(a,x) was used to realize the branch computing. Thirdly, the computing results were retrieved from GPU by FBO texture. Then one could get the shortest path and update path pheromone trail values through CPU processing. The loop steps were listed bellow:

(1) Initialize GPU computing environment, including GLUT initialization, FBO texture initialization, and Cg running environment etc.

(2) Enter main loop, including assignment of MMAS parameters and count of iteration.

(3) Enter iteration process. This process was entirely running in GPU. We adopted multiple rendering techniques to realize the ants' status transferring action. Firstly, render the equation $[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}$ into a texture. Secondly, render ants' tabu list into a texture. Thirdly, compute $p_{ij,k}(t)$ of every ant, and update their tabu list texture. Next is to render ants' tour path and the pheromone trail values into a texture, and then retrieve data to CPU using FBO.

(4) Compute the shortest path with evaluation function and update pheromone quality in CPU.

(5) Update the pheromone values and enter next iteration.

IV. EXPERIMENT

In this paper, we chose typical TSP example with 30 cities and 8 ants and realize MMAS to solve TSP with CPU and GPU respectively. Our program run on HP XW9300 graphics workstation with Nvidia Quadro Fx 4500 display card, CPU is AMD 2.79Hz, 2GB RAM. We use standard C++ and Nvidia Cg as program language and our compiler is C++ .NET 2003. The parameters of MMAS are listed below:

City coordinates are shown in TABLE I.

TABLE I. CITY COORDINATES

```
#define MAXCITY 30
int
xPos[MAXCITY]={41,37,54,25,7,2,68,71,54,83,64,1
8,22,83,91,25,24,58,71,74,87,18,13,82,62,58,45,
41,44,4};
int
yPos[MAXCITY]={ 94,84,67,62,64,99,58,44,62,69,6
0,54,60,46,38,38,42,69,71,78,76,40,40,7,32,35,2
1,26,35,50};
```

Other parameters are initialized bellow:

$\alpha = 1.0$, $\beta = 5.0$, $\rho = 0.5$, Q=100, $\tau_{\min} = 0.003$, $\tau_{\max} = 0.34$

The results show that our implements can find the optimized path length that is 423.741. In order to test the GPU computing efficiency, we use different iteration counts in finding the shortest path respectively, see TABLE II.

TABLE II. COMPARISON OF TIME COSTS IN SOLVING MMAS

Count of iteration	GPU costs (ms)	CPU costs (ms)
1000	2781	3390
2000	5218	6766
3000	7656	10141
5000	12500	16922
7000	17344	23656
10000	24531	33766

The comparison of running time cost between GPU and CPU is shown in Figure 3.

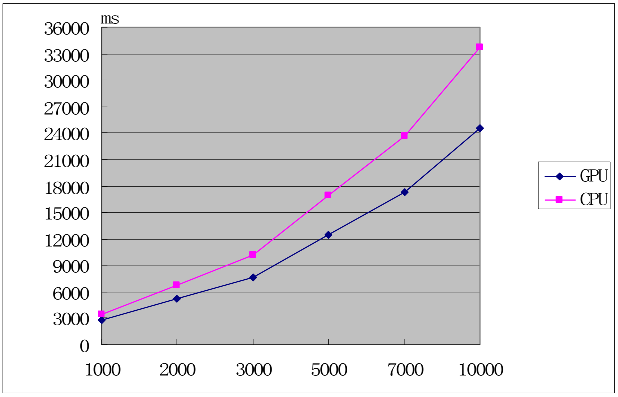

Figure 3. Comparison of computing cost.

V. Conclusions

In this paper, an implement of MMAS to solve TSP based on GPU was presented. The novel method realized fast solution of MMAS algorithm through constructing texture suitable to GUP processing. The experimental results show GPGPU has good accelerated performance on large scale computing. Because the GPU's SIMD features are designed for graphics pipeline, it is more difficult in realization than CPU. Our implement is suitable to GPGPU and can be used on solving other similar computing problem.

Acknowledgment

This paper supported by the National Natural Science Foundation of China under Grant No. 60832011; the Doctor Starting Fund of Civil Aviation University of China under Grant No. 06QD07S; the National High-Tech Research and Development Plan of China under Grant No. 2006AA12A105.

References

[1] Dorigo M., "Tree-Maps: a space-filling approach to the visualization of hierarchical information structures" *Proceeding of IEEE Visualization* '91, IEEE Computer Society Press, 1991, pp. 284-291.

[2] Cambardella L. M., Dorigo M, "Efficient Scheduling Focusing on the Duality of MPL Representatives," Proc. IEEE Symp. Computational Intelligence in Scheduling (SCIS 07), IEEE Press, Dec. 2007, pp. 57-64, doi:10.1109/SCIS.2007.357670.

[3] Stutzle T., Hoos H., "The MAX-MIN Ant System and Local Search for the Traveling Salesman Problem[C]", Proceedings of the IEEE International Conference on Evolutionary Computation, 1997, pp. 309~314.

[4] Oliver Deussen, David Ebert, el, "GPGPU: General-Purpose Computation on Graphics Hardware", In SIGGRAPH'2004 Course Notes 32, 2004.