2011 14th International IEEE Conference on
Intelligent Transportation Systems
Washington, DC, USA. October 5-7, 2011

Agent-based Traffic Simulation and Traffic Signal Timing Optimization with GPU

Zhen Shen, Kai Wang, Fenghua Zhu

Abstract—**With the advantage of simulating the details of a transportation system, the "microsimulation" of a traffic system has long been a hot topic in the Intelligent Transportation Systems (ITS) research. The Cellular Automata (CA) and the Multi-Agent System (MAS) modeling are two typical methods for the traffic microsimulation. However, the computing burden for the microsimulation and the optimization based on it is usually very heavy. In recent years the Graphics Processing Units (GPUs) have been applied successfully in many areas for parallel computing. Compared with the traditional CPU cluster, GPU has an obvious advantage of low cost of hardware and electricity consumption. In this paper we build an MAS model for a road network of four signalized intersections and we use a Genetic Algorithm (GA) to optimize the traffic signal timing with the objective of maximizing the number of the vehicles leaving the network in a given period of time. Both the simulation and the optimization are accelerated by GPU and a speedup by a factor of 195 is obtained. In the future we will extend the work to large scale road networks.**

Index Terms—**Microsimulation; Multi-Agent Systems; Intelligent Transportation Systems; GPU; Genetic Algorithms**

I. Introduction

Traffic simulation [1-21] is an important tool for control and management of urban traffic systems as the experiments on the real traffic systems are usually very costly. Early traffic simulation systems tended to be macroscopic or mesoscopic based on hydromechanics or statistical physics. Typical methods include the Lighthill_Whitham_Richards (LWR) model [10,11] and Lattice Blotzmann Methods (LBM) method [12]. These models are good at describing the overall properties of the traffic flow but lack the flexibility to describe the complex microscopic behaviors such as lane changing and vehicle overtaking. As the development of computers, the microsimulation methods such as car following model [13], Cellular Automata (CA) [14] and Multi-Agent Systems (MAS) [15, 16] become more and more popular in traffic analysis and forecasting. Many microsimulation systems are developed, for example [1, 9, 17-19], MATSim, VISSIM, TRANSIMS, TransWorld, MITSIM, MITSIMU, CORSIM, SHIVA and UTOBAHN. These systems can describe the microscopic behavior of the vehicles but has to face a great challenge that it is time-consuming to calculate the state evolution of vehicles. The computation time consumed by microscopic traffic flow simulation increases very fast as the road network expands and the number of vehicles increases. Moreover, the optimization of the traffic system with a microsimulation model often involves in algorithms such as the Genetic Algorithms (GA) which need to evaluate the system many times. There is a great challenge in computing when GA is applied to solve the traffic signal timing optimization problem [22-24].

While the demand for computing power in the Intelligent Transportation Systems (ITS) research is growing, the computing hardware is going through a revolution. Relevant graphics device companies propose the concept of general purpose computing based on the innate characteristics of data parallel computing of Graphics Processing Units (GPUs) [25-31]. GPU is a specialized circuit originally designed to offload graphics tasks from the CPU with the intention of performing them faster than the CPU can do. In a personal computer, GPU usually appears on the video card or the mother board. Usually it has excellent floating point performances with many cores working together to draw triangles and polygons on the screen. Because of this, people began to use it for scientific computing. However, people had to map their applications into problems that draw graphs and program with graphs programming languages like Open Graphics Library (OpenGL) and Cg. NVIDIA, a great GPU producer, realized the potential to use GPU for general purpose computing, and developed General-Purpose GPU (GPGPU) and Compute Unified Device Architecture (CUDA). With CUDA, people can program with high-level languages such as C, C++ and Fortran. The GPGPU idea and CUDA make the objects processed by GPU converted from pixels on screen to different kinds of scientific data. There have been many successful applications of GPGPU, such as molecular dynamics [26], computational fluid dynamics [27], bioinformatics [28] and scheduling [29]. GPU can make considerable speedups compared with CPU and is much affordable than other kinds of computer hardware that have the same computing performance. Lately some efforts have already been made in microscopic traffic simulation using CUDA [20,21] that opened up a new way for implementation and parallelization of microscopic traffic simulation.

Just as the development of CPU makes the iterative algorithms prevailing, we believe that the development of GPU can make the parallel algorithms in an iterative fashion prevailing. In this paper, we report some preliminary work on

This work is supported in part by NSFC 60921061, 70890084, 90920305, 90924302, 60904057 and 60974095; CAS 2F09N05, 2F09N06, 2F10E08, and 2F10E10.

Dr. Zhen Shen, Kai Wang and Dr. Fenghua Wang are with the State Key Laboratory for Intelligent Control and Management of Complex Systems, Beijing Engineering Research Center for Intelligent Systems and Technology, Institute of Automation, Chinese Academy of Sciences, No. 95 Zhongguancun East Road, Haidian District, Beijing 100190, China. (phone: +86-10-82615422, fax: +86-10-82615087, e-mail: zhen.shen@ia.ac.cn, kai.wang_nudt@hotmail.com, fenghua.zhu@ia.ac.cn.) Kai Wang is also with Center for Military Computational Experiments and Parallel Systems Technology, and College of Mechatronics Engineering and Automation, the National University of Defense Technology (NUDT), Changsha, Hunan, China.

using GPU for the traffic microsimulation and optimization. We build a parallel MAS model for a road network with four signalized intersections. This model builds a GPU based MAS model of traffic simulation environment following the work by David Strippgen [20,21], and then we apply GPU-adapted parallel Genetic Algorithm (GA) [22] to optimize the traffic signal timing configurations. By using NVIDIA GTX 470 we obtain a speedup by a factor of 195 compared with a mainstream CPU of AMD Athlon^{TM} 64 X2 Dual Core processor 4000+. The contribution of the paper is that we use GPU to parallelize the microsimulation and optimization of a traffic system and show that this integration of GPU with MAS and parallel iterative algorithms can help solve real problems more practically.

The remaining parts of the paper are organized as follows. In Section II, we give a review on GPU and the microsimulation and optimization of the traffic systems. In Section III, we give the formulation of the problem and show how to implement the parallel traffic simulation and optimization model with GA on GPU. In Section IV we show the experiment results. In Section V we conclude the paper and discuss the future research.

II. Review

A. GPU, Fermi and CUDA

Currently, millions of personal computer users are using NVIDIA GPUs for various different purposes, most of which are related to acceleration of graphics rendering or scientific computation. GPU is the core of the display card and is controlled by CPU. In hardware, a GPU has many cores working together. The cores are called Streaming Processors (SP), and several cores (8 or 32 typically) are organized into a Streaming Multi-processor (SM). In software, a typical GPU program consists of two parts: one part is the CPU codes that control the process of the whole program and does the sequential work, and the other is the GPU part that does the parallel work. With CUDA, the programmers can use C style codes to use the computing resources provided by GPU and the programming on GPU has no much difference from using Application Programming Interfaces (APIs).

Since 2006, NVIDIA has introduced three generations of hardware architectures of GPU: G80, GT200, and Fermi. The latest Fermi architecture makes great innovations and offers dramatically increased programmability and compute efficiency. Compared to GT200, some key features of Fermi are:

1) Each SM has 32 SPs that is 4 times over GT200.
2) The peak double precision floating point performance is 8 times over GT200.
3) Each SM could have a 64KB shared memory that is 4 times over GT200.

We give some key performance indicators of one Fermi GPU named GeForce GTX 470 in Table I.

A function that executes on the GPU is typically called a "kernel" [25]. When a kernel is launched, multiple threads on GPU organized by two levels are activated. The top level is called "grid" and the other is called "block". One grid can consist of at most 65535×65535 blocks and each block can consist of at most 512 threads. Then the grid is allocated to GPU for parallel computing with blocks allocated to different SMs in the GPU and threads allocated to different SPs in the SM. Each SM has its own memory called shared memory that all threads in it can access simultaneously, and all SMs share the global memory, constant memory and texture memory of GPU. Among all kinds of memories accessed by threads in the grid, the shared memory has the lowest memory access latency while the global memory has the highest latency. Fig. 1 illustrates the GPU parallel computing.

TABLE I KEY SPECIFICATIONS OF GEFORCE GTX 470

Number of SPs (cores)	448
Processor Clock (*MHz*)	1215
Memory Bandwidth (*GB/s*)	133.9
Video Memory (*MB*)	1,280

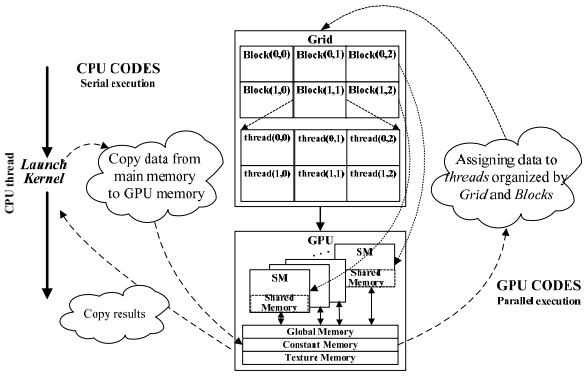

Fig. 1. GPU parallel computing

One big achievement of the GPU is that Tianhe-1A, the second fastest computer in the world (overtaken by the K computer in Jun. 2011), uses 7,168 NVIDIA Tesla M2050 GPUs and 14, 336 CPUs. It would require more than 50,000 CPUs to deliver the same performance, and the power consumption would increase from 4.04 megawatts to more than 12 megawatts [30]. This shows clearly the advantage of GPU.

B. Traffic mircosimulation and optimization

In the microsimulation, the road network and the vehicle travel are two elementary parts. Usually a graph is used to describe the topology of the road network. The nodes and the links represent the intersections and roads. And a queue simulation algorithm can be used to simulate the behaviors of cars travelling along the links. When not considering the lane changing or vehicle overtaking, the relationship between two sequential vehicles can be described by a GM-type car-following model [32],

$$a_n(t+\tau_n) = \alpha \frac{v_n(t+\tau_n)^{\beta}}{[x_{n-1}(t)-x_n(t)]^{\gamma}}[v_{n-1}(t)-v_n(t)], \qquad (1)$$

where n is used to index vehicles. The $(n-1)$-th vehicle is in

front of the n-th vehicle. $a_n(t)$, $v_n(t)$ and $x_n(t)$ are the acceleration, speed and position of vehicles at the time t , τ_n is the driver's reaction time of the vehicles, α , β and γ are constant parameters that need to be estimated from the real car-following data.

Many microsimulation systems are based on the above methods of describing the road network and the vehicle travel. As the computer develops, the performance of microsimulation systems becomes better and better. For example, MATSim has a multi-core version of its so-called Deterministic, Event-based Queue-Simulation (DEQSim), and by using a Beowulf Cluster (with 8 nodes) it can simulate the traffic flow of Switzerland consisting of about 20,000 nodes and 60,000 links for one whole day (24 hours) within 7 min [17]. However, the Ethernet network latencies still make it difficult to gain speedup by adding more clusters [20,21]. The GPU can be used to accelerate the microsimulation. It is reported in [20] that a speedup of up to 67 times compared against the highly optimized java version MATSim on a CPU was achieved.

Besides traffic simulation, there are many optimization problems in the ITS research. In [23] GA is used to solve the problem of traffic signal timing optimization with a 34% improvement over the mutually consistent solution of the problem. It has been challenging for the microsimulation, not to mention the optimization based on the microsimulation. This is why we believe that GPU is a necessary and promising tool for the traffic microsimulation and optimization. Fortunately, GA has been modified into a parallel version adapted to GPU. In [29], the authors use the parallel GA algorithm to solve the Quadratic Assignment Problem (QAP) and achieve a speedup factor of 3 to 12 by GTX 285 compared to the Intel i7 965 processor. This parallel GA is the method that we use in this paper for solving the traffic signal timing optimization problem.

III. Formulation and Implementation

A. Problem Formulation and the Overall Implementation

In this paper we maximize the number of vehicles leaving a road network in a given time period. This problem is important in that the performance is improved by software without upgrading the hardware. This problem is a typical problem in the ITS research [23].

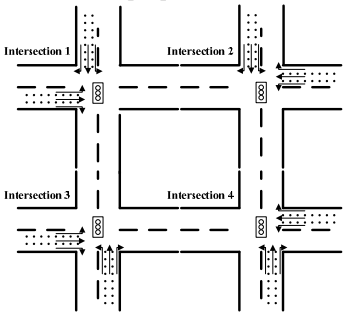

Fig. 2. The road network

We index the intersections in the road network arbitrarily from 1 to N. The road network used in this paper is shown in Fig. 2. For every road there are several lanes. We assume that there are vehicles from both directions on a road and we do not consider vehicle overtaking or lane changing in this paper. For every intersection, there are several phases of the traffic lights, shown in Fig. 3. We assume that there are $M = 4$ phases and that the sequences of phases are the same for all the intersections. The M phases constitute a "cycle" and we assume that all intersections share the same cycle time, denoted by c. We take the 1-st intersection as the reference intersection and the offset time of cycles of the intersections is denoted by a vector $\vec{\psi} = [\psi_1, \psi_2, \cdots, \psi_M]^T$. Obviously $\psi_1 = 0$. We denote $\vec{\theta}_m = [\theta_{1m}, \theta_{2m}, \cdots, \theta_{Nm}]^T$ ($m = 1, 2, ..., M$) with θ_{nm} ($n = 1, 2, \ldots, N$) as the green time of the m-th phase of the n-th intersection. The problem can be formulated as follows,

$$\begin{aligned} &\max f(c, \vec{\psi}, \vec{\theta}_1, \vec{\theta}_2, \cdots, \vec{\theta}_M) \\ &s.t. \quad c_{\min} \le c \le c_{\max} \\ &\qquad 0 \le \vec{\psi} \le c e_N, \\ &\qquad \vec{\theta}_m \ge \theta_{\min,m} e_N, m = 1, 2, \cdots, M, \end{aligned} \tag{2}$$

where $f(c, \vec{\psi}, \vec{\theta}_1, \vec{\theta}_2, \cdots, \vec{\theta}_M)$ is the function of the number of vehicles that leave the road network during the given time, $c_{\min}$ and $c_{\max}$ are the minimum and maximum values of the cycle time c, e_N is a column vector of N components all being ones, $\vec{\theta}_{\min,m} (m = 1, 2, \cdots, M)$ is the minimum green time for the m-th phase.

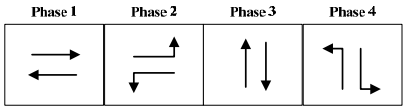

Fig. 3. Phase sequence of the traffic lights

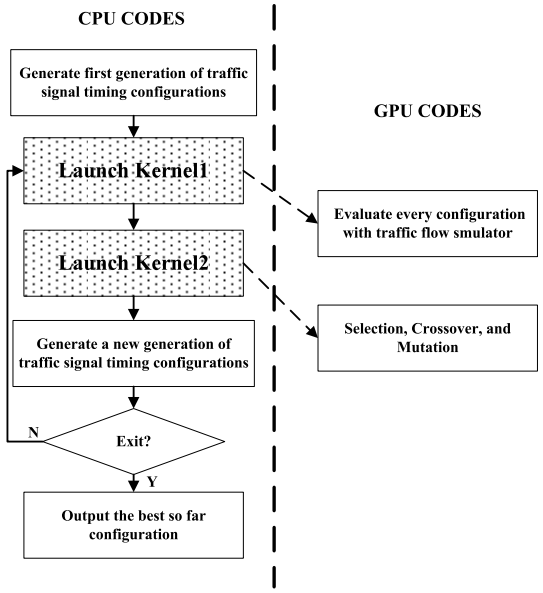

Fig. 4. Overall implementation

There are two main parts in the implementation: one is the

traffic flow simulator, and the other is the traffic signal timing optimizer. Every vehicle is viewed as an agent. The operations for the agents are similar. Also, the operations for the chromosomes in GA are also similar. These make the parallel implement on GPU possible, please see Fig. 4.

B. Traffic Flow Simulator

The road is divided into several lanes. We use the GM-type car-following model [32] to describe the movements of the vehicles as it has a clear physical meaning, please see (1). In this model, vehicles are queued in a lane and the state of a vehicle at a simulation time step can be computed by the states of itself and the vehicle in front of it at the previous simulation time step.

We assume that the arrival of the vehicles is subject to Poisson distribution with the arrival rate λ for each lane of the road network. The random numbers $u_1, u_2, ..., u_n$ which obeys the uniform distribution are generated by a Mersenne Twister (MT) generator [33]. This method can be applied easily on GPU. Each lane of the roads in our simulator has its own vehicle generator.

We use a "queue" to represent the vehicles in a lane. We assume that when the distance between a vehicle and the one at its front is less than 125 meters, the behind one moves in a car-following mode [34]; otherwise, it travels with a desired speed of its driver.

For vehicle agents in a queue, the most front is different from the others as there is no vehicle at its front. Its movement is totally determined by the states of traffic light and its distance from the stop line. We assume that the most front vehicle has an initial speed which obeys a normal distribution. When a vehicle arrives at an intersection, it may go forward, or turn left or turn right. No matter which action it takes, we assume that it chooses the new lanes with an equal probability.

C. Traffic Signal Timing Optimizer

1) Parallel GA model

GA is an iterative searching algorithm that mimics the process of natural evolution. Its main idea is that the better individuals are more likely to pass the genes to next generations. Recently, there has been a growing interest in implementing parallel GA with GPU [29, 31]. Here we employ the parallel GA algorithm with the GPU designed in [30] which can solve the quadratic assignment problem (QAP) efficiently. In this parallel GA, two pools P and W of the same size are used to store current and newly generated offspring individuals. The algorithm flow is as follows,

Step 1: Generate an initial population of individuals of P.

Step 2: Evaluate the individuals in P.

Step 3: For each individual I_i in P, select another individual $I_j (i \neq j)$ in P randomly. Apply crossover to I_i and I_j with the probability P_c. If the crossover happens, put the child I_i' into W; otherwise, copy I_i into W.

Step 4: For each individual I_i' in W, apply the mutation with the probability P_m.

Step 5: Evaluate individuals in W.

Step 6: For each individual I_i in P and its corresponding child or copy I_i' in W, compare the fitness values. If I_i' has a higher fitness, replace I_i in P with I_i' in W.

Step 7: If the termination criteria are met, terminate the algorithm. Otherwise, go to Step3.

The steps 2 to 6 all have the same operations on chromosomes. This is why this GA can be parallelized.

2) Fitness function

The fitness function is used to measure how good a solution is. In this paper we choose the number of vehicles leaving the road network in the given time period as the fitness measure [23]. It is just the objective function in (2).

3) Chromosome encoding and decoding

We use a binary encoding that the cycle time, the offset and the duration of green time of the phases can be encoded in one chromosome that is shown in Table II.

TABLE II CHROMOSOME ENCODING

Traffic signal timing for the intersections							
No. 1					No. 2	…	No. *N*
Cycle time, c	Offset ψ_1	duration of green time of phase 1, θ_{11}	…	duration of green time of phase *M*, θ_{1M}			
8 bits	8 bits	8 bits		8 bits			

We need (M + 2) 8-bit codes for an intersection with the first two for the cycle time and the offset and the remaining M for the M phases. For the decoding, for any intersection, we define a mapping factor ϕ_i as follows,

$$\phi_m = \sum_{k=0}^{7} 2^k b_{mk}, m = 1, 2, \cdots, M+2, \quad (3)$$

where b_{mk} is the value of the k-th bit of the m-th 8-bit binary code. The decoding method is shown in Table III.

TABLE III MAPPING BINARY CODES TO DECIMAL

Cycle time	$c = c_{\min} + \phi_1 \frac{c_{\max} - c_{\min}}{255}$
Offset	$\psi_n = \phi_2 \frac{c}{255}$
Duration of the green time for m-th phase	$\begin{cases} p_{nm} = p_{\min} + \phi_{m+2} \frac{p_{\max} - p_{\min}}{255}, m = 1, 2, ..., M \\ \theta_{nm} = \theta_{\min,m} + \frac{p_{nm}}{\sum_{i=1}^{M} p_{ni}} (c - \sum_{i=1}^{M} \theta_{\min,i}), m = 1, 2, ..., M \end{cases}$

In Table III, n is used as the index for the intersection, and p_{nm} is the parameter to determine the green time for the m-th phase of the n-th intersection, with $p_{\min}$ and $p_{\max}$ as its minimum and maximum values.

4) Crossover and mutation

The one-point crossover operator and the one-point mutation operators are with the probabilities of crossover P_c and mutation P_m respectively.

D. Data Structures and Computing Resources Allocation

The data of vehicle agents is organized from top to bottom in five levels: the traffic lights configurations, intersections, roads, lanes, and vehicles. Please see Fig. 5. Before we launch the kernel function on GPU, the data of vehicle agents and traffic lights are copied to memories on GPU. The vehicle data is copied to the global memory of GPU and then copied to the shared memory of the SMs as the access to the shared memory is faster. Traffic light data are copied to the constant memory as the vehicles controlled by the same traffic lights configuration need it and the constant memory can be read by all the threads in the grid with lower memory access latency than the global memory.

As Fig. 5 shows, a block handles the vehicles in the same lane, and each thread in it handles a single vehicle on the lane. Each row of blocks in the grid handles a replication of the computing of the traffic lights configuration. For each configuration, we simulate for T independent replications to reduce the uncertainty.

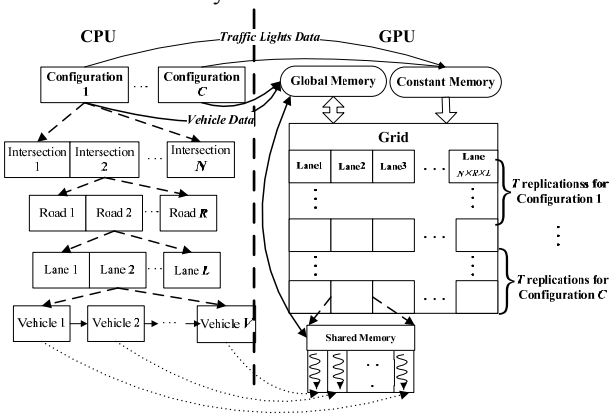

Fig. 5. Data structures and computing resources allocations

IV. EXPERIMENTS

In this paper, we use the road network shown in Fig. 2. The number of intersections is $N = 4$. For each road, there are 6 lanes with 3 at either side. The right side lane is used for turning right, the left for turning left and the middle is used for going forward. When going past the stop line, the vehicle chooses new lanes with equal probabilities, i.e., 1/3.

The distance between two neighboring intersections is $256 \times 4 = 1024m$, with $4m$ as the average length of a vehicle. The vehicle arrival rate for each entry lane in the road network shown in Fig. 2 (24 entry lanes altogether) is $\lambda = 0.2$ vehicle per second. The initial speed of the car obeys the normal distribution $N(60 \text{ km/h}, 10)$, and the desired speed is 80 km/h. The parameters in the car-following model are $\alpha=\beta=\gamma=1$. The number of phases is $M = 4$.

The maximum and the minimum of the cycle time are $c_{\max} = 240s$ and $c_{\min} = 60s$. The parameters for phases are $p_{\min} = 0, p_{\max} = 100$ and $\theta_{\min,m} = 6s$ (m = 1, 2, …, M). The time step for the simulation is $1s$. We simulate for $3{,}600s$.

For GA, the number of individuals for each generation is $C = 500$. The replications for evaluating a configuration is $T = 10$. The probabilities for the crossover and the mutation are $P_c = 0.99$ and $P_m = 0.05$. We set GA for 1,000 generations.

We use a PC with one AMD Athlon™ 64 X2 Dual Core processor 4000+ and an NVIDIA GeForce GTX470 GPU. We use CUDA driver and SDK with the version 3.2.

The results of a typical run of GA are shown in Fig. 6. Table IV shows the output of the GA. The number of vehicles that leave the road network is corresponding to Fig. 6 is 8835.

We give the time consumption for one iteration of GA for the CPU + GPU implementation and compare it with the implementation of CPU only in Table V. The results are based on 30 runs independently for both implementations. The whole process of the CPU + GPU implementation takes about $19{,}044s$ and it would be $3{,}719{,}976s$ (about 43.1 days) on CPU with a speedup factor of 195. The data of one generation transferred from the main memory to the GPU memory takes about 120 *MB* video memory.

Please note that the road network used in this paper (Fig. 2) is symmetric, and the results in Table IV verify the symmetry.

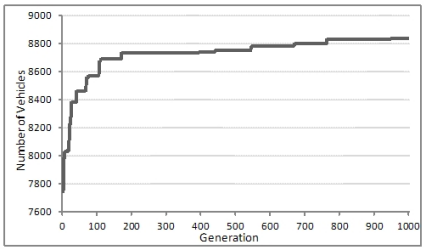

Fig.6. Results of a typical run of GA

TABLE IV TRAFFIC SIGNAL TIMING CONFIGURATION OUTPUT by GA

Cycle time *c/s*	Inter-section No.	Offset/*s*	Green time of Phase 1 /*s*	Green time of Phase 2 /*s*	Green time of Phase 3 /*s*	Green time of Phase 4 /*s*
61	1	0	10	10	19	22
	2	46	19	22	13	7
	3	32	19	22	11	9
	4	60	11	10	18	22

TABLE V TIME CONSUMPTION OF ONE ITERATION IN GA

	CPU ONLY	GPU+CPU	Speedup
Average Time/*s*	3719.976	19.044	195.336
Standard Deviation	12.778	1.496	N/A

V. CONCLUSION AND FUTURE RESEARCH

In this paper, we propose a framework of an agent-based traffic flow simulator and a traffic signal timing optimizer based on GPU using NVIDIA's CUDA. The GM type car-following model and the parallel GA are used. We obtain a speedup by a factor of 195 compared with the implementations on CPU only. Our results show the power of GPU on the transportation simulation and optimization. In the future, we will go on the research on the following points,

1) Improve the microsimulation model to make it more practical. We use a very simple model in this paper to demonstrate and verify the power of GPU. In the future,

we will make the model more practical by learning from mature software such as MATSim.

2) Solve large scale problems. The road network in the paper has only 4 intersections. A real network contains hundreds of intersections of many types. We plan to use GPU clusters to tackle the large scale problems.

3) Improve the optimization method. GA is not the only choice for the traffic signal timing optimization problem. We may consider GA with some local search algorithms to improve the performances.

4) Different criteria should be considered. We chose the number of vehicles leaving the road network as the objective to optimize. Other objective functions can be considered, such as the mean travel time of the vehicles, the mean length of the queues, or the mean stop times of the vehicles. Moreover, we should consider the criteria together and solve the multi-objective optimization problem.

Finally, we conclude the paper that this paper is only part of on-going research and we believe that it is promising to design and use parallel algorithms with GPU to solve the various simulation and optimization problems in the Intelligent Transportation Systems research.

REFERENCES

[1] F.-Y. Wang, "Parallel control and management for intelligent transportation systems: concepts, architectures, and applications," *IEEE Trans. on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 630-638, 2010.

[2] F.-Y. Wang and S. Tang, "Concepts and frameworks of artificial transportation systems," *Complex Systems and Complexity Science*, vol.1, no.2, pp.52-59, 2004. (in Chinese)

[3] N. Zhang, F.-Y. Wang, F. Zhu, D. Zhao and S. Tang, "DynaCAS: computational experiments and decision support for ITS," *IEEE Intelligent Systems*, vol.23, no.6, pp.19-23, 2008.

[4] F.-Y. Wang and S. Tang, "Artificial societies for integrated and sustainable development of metropolitan systems," *IEEE Intelligent Systems*, vol.19, no.4, pp. 82-87, 2004.

[5] N. Zhang, F.-Y. Wang, F. Zhu, D. Zhao and S. Tang, "DynaCAS: computational experiments and decision support for ITS," *IEEE Intelligent Systems*, vol.23, no.6, pp.19-23, 2008.

[6] Q. Miao, F. Zhu, Y. Lv, C.-J. Cheng, C. Chen and X. Qiu, "A game-engine-based platform for modeling and computing of artificial transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 343-353, 2011.

[7] F. Zhu, G. Li, Z. Li, C. Chen and D. Wen, "A case study of evaluating traffic signal control systems using computational experiments," *IEEE transactions on Intelligent Transportation Systems*, 2011, accepted.

[8] F.-Y. Wang, "Agent-based control for networked traffic management systems," *IEEE Intelligent Systems*, vol.20, no.5, pp. 92-96, 2005.

[9] B. Chen and H. H. Chen, "A review of the applications of agent technologies in traffic and transportation systems," *IEEE Trans. on Intelligent Transportation Systems*, vol. 11, no. 2, pp. 485-497, Jun. 2010.

[10] M. J. Lighthill and G. B. Whitham, "On kinematic waves.I. Flood movement in long rivers," in *Proceedings of the Royal Society of London*, London, British,1955, pp. 281-316.

[11] M. J. Lighthill and G. B. Whitham, "On kinematic waves. II. A theory of traffic fow on long crowded roads," in *Proceedings of the Royal Society of London*, London, British, 1955, pp. 317-345.

[12] I. Prigogine, R. Herman and R. S. Schechter, "Kinetic Theory of Vehicular Traffic," *IEEE Trans. Systems, Man and Cybernetics,* vol. 2, pp. 295, April 1972.

[13] D. Chowdhury, L. Santen and A. Schadschneider, "Statistical physics of vehicular traffic and some related systems," *Physics Reports,* vol. 329, pp. 199-329, May 2000.

[14] K. Nagel and S. Michael, "A cellular automaton model for freeway traffic," *Journal de Physique I,* vol.2, pp. 2221-2229, Dec. 1992.

[15] P. Davidsson, L. Henesey, L. Ramstedt et al., "An analysis of agent-based approaches to transport logistics," *Transportation Research Part C: Emerging Technologies,* vol. 13, pp. 255-271, Aug. 2005.

[16] R. Schleiffer, "Intelligent agents in traffic and transportation," *Transportation Research Part C: Emerging Technologies,* vol.10, pp. 325-329, Oct.-Dec. 2002.

[17] M. Balmer, M. Rieser and K. M. et.al., "MATSim-T: Architecture and Simulation Times," in A..Bazzan and F. Klügl (eds.) *Multi-Agent Systems for Traffic and Transportation Engineering,* New York: Hershey, 2009.

[18] M. Fellendorf, "VISSIM: A microscopic simulation tool to evaluate actuated signal control including bus priority," in *64th Institute Transportation Engineers (ITE) Annu. Meeting*, Dallas, TX, USA, 1994, pp. 1-9.

[19] K. Nagel and M. Rickert, "Parallel implementation of the TRANSIMS micro-simulation," *Parallel Computing,* vol.27, pp. 1611-1639, Nov. 2001.

[20] D. Strippgen and K. Nagel, "Multi-agent traffic simulation with CUDA," in *International Conference on High Performance Computing & Simulation,* Leipzig, Germany, 2009, pp. 106-114.

[21] D. Strippgen and K. Nagel, "Using common graphics hardware for multi-agent traffic simulation with CUDA," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, Brussels, 2009, pp.1-8.

[22] M. D. Foy, R. F. Benekohal and D. E. Goldberg, "Signal timing determination using genetic algorithms," *Transportation Research Record 1365 TRB,* pp. 108-115, Apr. 1993.

[23] H. Ceylan and M. G. H. Bell, "Traffic signal timing optimisation based on genetic algorithm approach, including drivers' routing," *Transportation Research Part B: Methodological,* vol.38, pp. 329-342, May 2004.

[24] J. J. Sánchez-Medina, M. J. Galan-Moreno, E. Rubio-Royo, "Traffic signal optimization in 'La Almozara' district in Saragossa under congestion conditions, using genetic algorithms, traffic microsimulation, and cluster computing," *IEEE Intelli. Syst.,* vol. 11, pp.132-141, Mar. 2010.

[25] J. Sanders and E. Kandrot, *CUDA by example, an introduction to General-Purpose GPU Programming*, p. 23, Addison-Wesley, MA, USA, 2011.

[26] T. Narumi, R. Sakamaki and S. Kameoka, "Overheads in accelerating molecular dynamics simulations with GPUs," in *Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies*, Otago 2008, pp. 143 - 150.

[27] J. Tölke, "Implementation of a Lattice Boltzmann kernel using the Compute Unified Device Architecture developed by nVIDIA," *Comput Visual Sci,* vol.13, pp. 29-39, Nov. 2009.

[28] A. Benso, S. Di Carlo, G. Politano et.al, "GPU acceleration for statistical gene classification," in *2010 IEEE International Conference on Automation Quality and Testing Robotics*, Cluj-Napoca, 2010, pp. 1-6.

[29] S. Tsutsui and N. Fujimoto, "Solving quadratic assignment problems by genetic algorithms with GPU computation: a case study," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference*, Montreal Quebec, Canada, 2009, pp. 2523-2530.

[30] NVIDIA Tesla GPUs power world's fastest supercomputer, http://www.nvidia.com/object/tesla_computing_solutions.html.

[31] R. Arora, R. Tulshyan, K. Deb, "Parallelization of binary and real-coded genetic algorithms on GPU using CUDA," in *2010 IEEE Congress on Evolutionary Computation*, Barcelona, Spain, 2010, pp. 1-8.

[32] X. Ma and I. Andréasson, "Driver reaction time estimation from real car following data and application in GM-type model evaluation," in *Proceedings of the 85th TRB annual meeting*, Washington D.C., 2006, pp.1-19.

[33] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation,* vol. 8, pp. 4-30, Jan. 1998.

[34] D. L. Gerlough and M. J. Huber, "Traffic flow theory," in *Transportation Research Board Special Report 165,* Washington D.C., 1975.