Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems (ITSC 2013), The Hague, The Netherlands, October 6-9, 2013

GPU based Non-dominated Sorting Genetic Algorithm-II for Multi-objective Traffic Light Signaling Optimization with Agent Based Modeling*

Z. Shen, *Member, IEEE*, K. Wang and F.-Y. Wang, *Fellow, IEEE*

Abstract— **Micro-simulation becomes more and more important in the Intelligent Transportation Systems (ITS) research, because it can provide detailed descriptions of the system. For a multi-agent systems (MAS) modeling of an ITS, the computation burden is large, as it involves the computation of the state changing of all the agents. And, there are many multi-objective optimization problems in the ITS research. In this paper, we solve the traffic light signaling optimization problem and we take the average delay time and the average stop times as two objectives. We use a famous method of Non-dominated Sorting Genetic Algorithm II (NSGA-II). As NSGA-II can be viewed as an intelligent way of running a number of micro-simulations, usually the computation burden is huge. Graphics Processing Units (GPUs) have been a popular tool for parallel computing. The real transportation system runs in parallel and we think that a parallel tool is more suitable for the simulation and optimization of the system. We test GPU based NSGA-II method on a 4 intersection lattice road network, and on the 18 intersection road network of the Zhongguancun area of Beijing. Compared with the CPU version, the GPU version implementation achieves a speedup factor of 21.46 and 27.64 respectively.**

I. Introduction

Traffic congestion is a notorious problem for almost all the big cities in the world. The transportation system is complex in that it does not only involve in vehicles, roads, pedestrians, traffic lights, but also infrastructure, logistics, weather and environmental, legal and regulatory, social and economic, ecological and resource factors [1, 2]. The Intelligent Transportation Systems (ITS) [1-14] is initialized, developed and deployed, and it greatly improves the transportation safety and efficiency, and travel reliability.

One of the essential problems in ITS is the modeling. Early traffic simulation systems tended to use macro- and meso-models based on hydromechanics or statistical physics. Typical macro-methods include the Lighthill Whitham Richards (LWR) model which uses the hydrodynamic equations to describe the temporal and spatial characteristics of traffic flow, and typical meso-methods include Lattice Boltzman Method (LBM) method which describes the dynamic changes of traffic flow velocity distribution function with time. These models may have good performance on describing the overall properties of the traffic flows, but both of them lack the capability of describing the details of the system and the complex microscopic behaviors such as vehicle overtaking and lane changing. In recent years, as the computer technologies become developed, the micro-simulation methods become more and more popular such as Cellular Automata (CA), and Multi-Agent Systems (MAS) [1, 3, 8-10]. The MAS model is flexible in that almost everything can be viewed as an agent and the agent can react to the changes in an autonomous way. In this way, an "Artificial Transportation System (ATS)" can be "grown up" in a bottom up fashion [1, 2]. A lot of commercial and academic traffic simulation software such as TRANSIMS, PARAMICS, MATSIM and TransWorld [1] provide micro-simulation modules. And more and more Metropolitan Planning Organizations (MPOs) have already used micro-simulation to help make decisions.

The problem with MAS model is that it involves in complex and large computations. For big cities, there are usually several million cars, not to mention other factors such as pedestrians, bicycles, roads, traffic lights. Moreover, there are many optimization problems, such as traffic light signaling optimization, which usually can be seen as an intelligent way of running the micro-simulation systems for many times. The computation burden is huge.

Genetic Algorithm (GA) is an efficient method for the optimization problems in complex systems research. It is a kind of meta-heuristic method inspired by the laws of evolution of the biosphere. It was proposed by Professor J. Holland in 1975. There are three main features for GA. The first is to operate on the structural objects directly. There is no limit to the derivative and the function continuity. The second is that GA owns the inherent implicit parallelism and better capability of the global optimization. The third is the usage of the probabilistic optimization method. GA can guide the search and adjust the search direction adaptively. There is no need to determine the rules.

Graphics Processing Unit (GPU) is now a powerful parallel computing tool. 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. Usually it has excellent floating point performances with

*This work is supported in part by NSFC 60921061, 61174172, 60904057, 31170670, 61101220, 70890084, 90920305, and CAS 2F11D03, 2F11N06, 2F11D01, 2F11N07, and by the Early Career Development Award of SKLMCCS, and by Zhejiant Tengtou Landscape Co. Ltd..

Prof. Z. Shen and Prof. F.-Y. Wang are with the State Key Laboratory of Management and Control for Complex Systems, Beijing Engineering Research Center of Intelligent Systems and Technology, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with Dongguan Research Institute of CASIA, Cloud Computing Center, Chinese Academy of Sciences. (e-mail: zhen.shen@ia.ac.cn, feiyue.wang@ia.ac.cn.)

Dr. K. Wang is with Center for Military Computational Experiments and Automation, the National University of Defense Technology(NUDT), Changsha, Hunan, China.(e-mail: kai.wang_nudt@hotmail.com.)

978-1-4799-2914-613/$31.00 ©2013 IEEE

many cores working together to draw triangles and polygons on the screen. NVIDIA 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. GPU has been used in traffic micro-simulation and good results are reported. In [13-14], the authors used GTX 285 and achieved a speedup factor of more than 67, compared with a highly optimized Java version MATSIM CPU program. We have used GPU based GA for the traffic signal timing optimization problem with the throughput as the single objective. Compared with CPU version, we achieve a speedup factor of 195 [8].

In this paper, based on GPU, we try to solve the multi-objective traffic lights signaling optimization problem with Non-dominated Sorting Genetic Algorithm-II, which is the GA for multi-objective optimization version. The main difference between NSGA and the simple genetic algorithm is that NSGA will perform the stratified operation according to the relations of domination between individuals before the selection operator. The selection operator, crossover operator and mutation operator are the same with the simple GA. The non-dominated stratified method makes good individuals have a greater chance to be passed on to next generation. Fitness sharing strategy makes the individuals evenly distributed. NSGA also maintains the diversity of population. Compared with NSGA, NSGA-II is the extension with the low computational requirements, elitist approach, and parameter-less sharing approach. There are two objectives: delay time and stop time. The two objectives are both affected by the period of the traffic lights. If the period is short, the number of stops of a vehicle may increase. If the period is long, the waiting time for the green light may increase. The two objectives should be balanced and this is a well-known problem in the ITS research. We test the method on a 4 intersection lattice road network, and on the 18 intersection road network of the Zhongguancun area of Beijing. The results show that it is promising to use NSGA-II to solve the multi-objective optimization problems of a large and more real road network.

II. Literature Review

A. Traffic Micro-simulation and optimization

We give a very short review here. For more information, please refer to [11], which is a paper dedicated to the GPU based MAS model by the same authors.

In the traffic micro-simulation, we usually describe the topology of the road network with a graph, in which the nodes and links represent the intersections and roads respectively. There are many car following models, among which GM-type model [8, 11] is a popular one,

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

where n stands for the indexes of vehicles, and the (n–1)-th vehicle is in front of the n-th vehicle. $a_n(t)$ is the acceleration of vehicles at the time t, $v_n(t)$ is the speed at the time t, and $x_n(t)$ is the position at the time t. τ_n is the reaction time of the agents, α, β and γ are constant parameters that need to be estimated from the real car-following data.

For the lane changing model, the Gipps model [11] is a well-known one. Generally speaking, the whole process of Gipps model can be divided into three steps: the first is to generate lane-changing intentions, then to judge the feasibility of lane changing and choose a destination lane, finally to perform lane changing. Here we classify the model into two types, based on the different lane-changing intentions: discretionary lane changing and compulsory lane changing. The aim of discretionary lane changing is to travel with desired speed, while the aim of compulsory lane changing is to follow the planned path by travelling in one of the allowed lanes for the desired turning movement. For the agent of a specific vehicle which is travelling in a lane, when there is a long distance between the agent and the intersection, the discretionary lane changing can be performed. And when there is a not enough long distance between them, compulsory lane changing can be performed. If the required deceleration for the new follower to allow the subject agent to move into the destination lane and the required deceleration for the subject agent to move behind the new leader in the destination lane are both acceptable, the lane changing is feasible. The acceptable deceleration b_n is given by (2). It is physically possible and safe for the vehicle agents to change lanes without an unacceptable risk of collision when the actual required deceleration is larger than b_n.

$$b_n = [2-(D-x_n(t))/(10V_n)]\cdot b_{LC}\cdot\theta, \qquad (2)$$

where b_n is the acceptable deceleration of agent n, D is the location of the intended turn or lane blockage, $x_n(t)$ is the location of n at time t, V_n is the desired speed of n, b_{LC} is the average deceleration an agent is willing to accept in lane changing, θ is the agent's aggressive parameter which models the drivers' different driving habits.

In the ITS research, there are many optimization problems besides traffic simulation. In [15], GA is used to solve the problem of single objective traffic signal timing optimization and obtains a 34% improvement over the mutually consistent solution of the problem. In [10], the authors use GA to generate more reasonable daily activity plans. However, the problem with such methods is that the computation burden is very heavy. This is why usually clusters and GPUs are employed. Besides the hardware improvement, we aim to choose more suitable and efficient algorithms to solve the optimization problems with MAS models.

B. GA and NSGA-II

The GA method was invented by Holland in 1975 and now is an efficient method for complex systems research. For simulation based optimization, usually we formulate the problem as follows,

$$\min_{\theta\in\Theta} J(\theta) = E[L(x(t;\theta,\xi))], \qquad (3)$$

where $J(\theta)$ is a suitable performance function with the objective of minimizing its expected value. θ stands for the parameters or variables of the traffic system that may subject to choices and Θ stands for the whole search space for the variables or parameters which is very large but finite. Usually θ and Θ are called "design" and "search space" respectively. ξ stands for the randomness of the system and x is a trajectory for θ generated according to ξ.

The traditional mathematical programming usually starts from a single initial value, and obtains the optimal solution by iteration. Therefore it is easy to fall into local optima. But a population is the unit for searching in GA, so it is conducive to achieve global optimization. Also the assessment of the individuals in the population can be parallelized. This improves the efficiency of the iterative calculations.

NSGA-II extends GA to solve multi-objective problems. Without loss of generality, we consider a problem to minimize n objective functions $J_1, J_2, \ldots, J_n$. A design θ_1 is said to dominate θ_2 denoted by $\theta_1 \prec \theta_2$, if $J_i(\theta_1) \le J_i(\theta_2)$, for i = 1, 2, …, n, with at least one inequality being strict. A set of designs L_1 is said to be in the Pareto frontier in terms of the objective functions $J_1, J_2, \ldots, J_n$ if it contains all the designs that are not dominated by other designs Θ, i.e.,

$$L_1 \equiv \{\theta \mid \theta \in \Theta, not\ \exists \theta' \in \Theta,\ \text{s.t. } \theta' \prec \theta\} \quad (4)$$

All the designs in the Pareto frontier are called Pareto optimal. Further, a series of designs given by

$$L_{s+1} = \Omega(\Theta \setminus \cup_{i=1,\ldots,s} L_i), s = 1, 2, \ldots \quad (5)$$

are called layers. Designs in L_s are called s designs. They are the successive Pareto frontier after the previous layers have been removed from consideration.

NSGA-II is a powerful tool for multi-objective optimization. It is proposed on the basis of the NSGA by Srinivas and Deb [16]. It has more advantages than the NSGA.

- NSGA-II uses a fast non-dominated sorting algorithm, and greatly reduces the computational complexity compared with NSGA.
- Instead of the shared radius "σ_{share}" that needs to be specified, NSGA-II adopts the virtual fitness which guides the selection operation at the various stages of the algorithm. It maintains the diversity of population and gets a uniformly spreading out Pareto-optimal front.
- NSGA-II introduces the elitist strategy which is useful for preventing the loss of good solutions and speeding up the performance significantly.

III. Problem Description and Application of NSGA-II

Except the multi-objective, the formulation is the same with the one presented in [8, 11, 12], which are also our papers. For readers' convenience, we give a concise description here.

A. Road Network Modeling

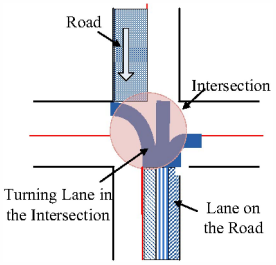

Figure 1. An intersection

For every road there are usually several lanes. Please see Figure 1. The vehicles in a lane can be modeled as a queue with a fixed capacity. Once the capacity is exhausted no more vehicles can enter the queue. The right most and left most are dedicated lanes for turning. We assume that the vehicles follow given paths when turning.

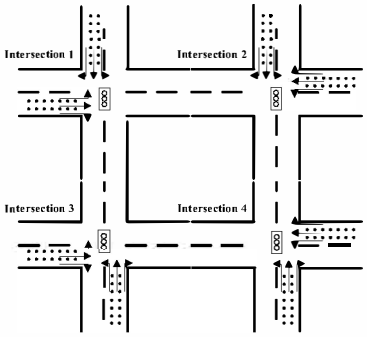

Figure 2. A 4 intersection road network

We index the intersections in the road network arbitrarily from 1 to I. Please see Figure 2. We use the GM-type car-following model and the Gipps lane changing model, as we described in Section II.

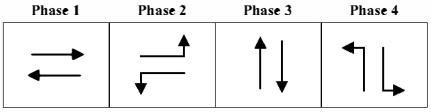

Figure 3. Phase sequence of the traffic lights

For every intersection, there are several phases of the traffic lights, shown in Figure 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. For T shape intersections there are only M = 3 phases.

B. Routing

Every agent tends to travel from the origin to the destination along the fastest path. The well-known Dijkstra algorithm [17] is used for routing the vehicles. The algorithm uses the free travel time as the weights for roads if the vehicle travels only once, while uses the average travel time for weights if there have been multiple travels in the road network.

C. Problem Formulation

We index the intersections arbitrarily and take the 1-st intersection as the reference intersection. The offset time of cycles of the intersections is denoted by a vector $\vec{\psi} = [\psi_1, \psi_2, \cdots, \psi_I]^T$. Obviously $\psi_1 = 0$. We denote $\vec{\theta}_m = [\theta_{1m}, \theta_{2m}, \cdots, \theta_{Im}]^T$ ($m = 1, 2, ..., M$) with θ_{im} ($i = 1, 2, \ldots, I$) as the green time of the m-th phase of the i-th intersection. The problem can be formulated as follows,

$$\begin{aligned} &\min F = [f_1(c, \vec{\psi}, \vec{\theta}_1, \vec{\theta}_2, \cdots, \vec{\theta}_M), f_2(c, \vec{\psi}, \vec{\theta}_1, \vec{\theta}_2, \cdots, \vec{\theta}_M)]^T \\ &s.t. \quad c_{\min} \le c \le c_{\max} \\ &\qquad 0 \le \vec{\psi} \le c e_I \\ &\qquad \vec{\theta}_m \ge \theta_{\min,m} e_N, m = 1, 2, \cdots, M \end{aligned} \tag{6}$$

where $f_1(c, \vec{\psi}, \vec{\theta}_1, \vec{\theta}_2, \cdots, \vec{\theta}_M)$ is the average delay and $f_2(c, \vec{\psi}, \vec{\theta}_1, \vec{\theta}_2, \cdots, \vec{\theta}_M)$ is the average stop times, $c_{\min}$ and $c_{\max}$ are the minimum and maximum values of the cycle time c, e_I is a column vector of I components all being ones, $\vec{\theta}_{\min,m} (m = 1, 2, \cdots, M)$ is the minimum green time for the m-th phase.

D. Encoding and Decoding of Designs

TABLE I. BINARY ENCODING

Traffic signal timing for the intersections							
No. 1					No. 2	…	No. *I*
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			

TABLE II. MAPPING BINARY CODES TO DECIMAL

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

In order to apply NSGA-II, 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 binary vector shown in Table I.

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,

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

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

E. Algorithm Flow

The application of NSGA-II is described as follows.

Step 1: Set the generation step $t = 0$. Generate the first generation P_0 with N random individuals in it. Set $P_0' = P_0$, and then use the crossover and mutation operators to create a child population Q_0 of size N. We combine the individuals in P_0' and Q_0 to obtain R_0.

Step 2: Run the micro-simulation to obtain the delay time and stop times for each individual in R_t.

Step 3: According to the delay time and stop times, calculate the non-domination layer p_{rank} of each individual p in R_t, which is constituted by P_t and Q_t. To obtain the non-domination layer, first we need to compare each solution with every other solution to get two important parameters: n_p and S_p. n_p is the number of solutions which dominate the solution p. S_p stands for a set of solutions which the solution p dominates. From the definition, we can know that $p_{rank} = 1$ if $n_p = 0$, and p is much likely to be discarded if $S_p = \emptyset$. In other cases, the individual p in p_{rank} is always dominated by certain member in $p_{rank} - 1$ and dominates some member in $p_{rank} + 1$.

Step 4: According to the non-domination layers, P_{t+1} is formed by adding individuals from the first several layers until the size of P_{t+1} exceeds N.

Step 5: Calculate the virtual fitness $p_{distance}$ of each individual p in the layers selected at step 4. The virtual fitness is used for estimating the density of solutions surrounding a point in the population. We use distances from this point to the neighbor points along each of the objectives as the measurement. It should be noted that there are two solutions which have only one neighbor in the same layer, if there are no less than two solutions in the layer. The $p_{distance}$ of these two points are assumed to be infinite so that the boundary points are always selected. It is trivial if we have less than two solutions in one layer. It is also trivial if we have individuals that are the same. For all other points in the same layer, according to each objective, we index the individual with i from the one with the smallest value to the one with the largest value. Then we can give an equation as follows,

$$p_{i,\text{distance}} = \sum_{m=1}^{M} \left(U_{i+1,m} - U_{i-1,m} \right) \tag{8}$$

Here M stands for the total number of the objectives, and $U_{i,m}$ is the value of m-th objective function for the i-th individual. In other words, for every objective function, we obtain the difference between the function values of two neighbors, and we sum all the differences to obtain the $p_{i,\text{distance}}$.

Step 6: Selection operation: we select two individuals p and q in P_{t+1}, and we choose p and drop q, if $p_{rank} < q_{rank}$, or $p_{rank} = q_{rank}$ and $p_{distance} > q_{distance}$. Repeat the process until the scale of the population reaches N. These N individuals constitute P'_{t+1}. The principle of selection operation is to select solutions near to the Pareto frontier. When the solutions are from the same layer, we select the solution that can make the population spreading.

Step 7: Get the temporary offspring population Q_{t+1} by the crossover and mutation operations. Then we can obtain R_{t+1} by combine P'_{t+1} and Q_{t+1}.

Step 8: Repeat steps 2 to 7 until the exit criteria are met.

F. 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. 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.

IV. EXPERIMENTS

We test NSGA-II on two cases. Case 1 is the 4 intersection road network given in Figure 2. Case 2 is the 18 intersection road network of Zhongguancun area of Beijing in Figure 4.

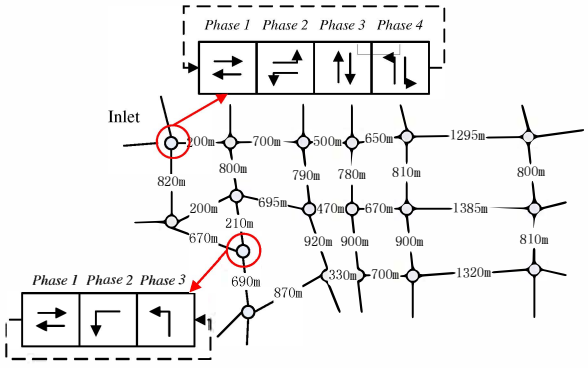

Figure 4. Zhongguancun area road network

For the 4 intersection lattice road network, the distance between neighboring intersections is 1024 m. There are three lanes in one direction and another three in the other. All the roads that are not directing to the intersections in the network are taken as inlets. The rate for every lane is 0.2 vehicle/s. The initial speed of the vehicle obeys N(40 km/h, 10), and the desired speed obeys N(60 km/h, 10). The parameters in the GM-type model is $\alpha=\beta=\gamma=1$ and the average deceleration an agent is willing to accept in lane changing b_{LC} = -4 m/s^2. At the intersections we assume that the number of vehicles going left, straight and right are equal. The step for simulation is 1s and the time for simulation is 3600 s. We set the maximum and minimum of the cycle of the traffic lights as c_{max} = 240 s and c_{min} = 120 s. The parameters for the maximum and minimum of the offset are p_{max} = 100, p_{min} = 0. The minimum green time is $\theta_{min,m} = 30s$ (m = 1, 2, …, M). The iterative time is 500, the size of the population in the configuration is 200, the repeated time of the evaluation in every coordinated control configuration is 10, the probability of crossover and mutation is 0.95 and 0.05 respectively.

We show the convergence process of the iteration calculation from the first iteration to the 500-th iteration. Please see Figure 5.

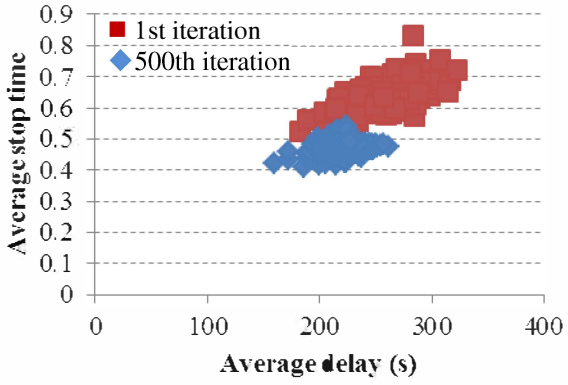

Figure 5. The convergence process of the iteration calculation

Through the iteration calculation, we obtain the Pareto frontier and show it in Figure 6.

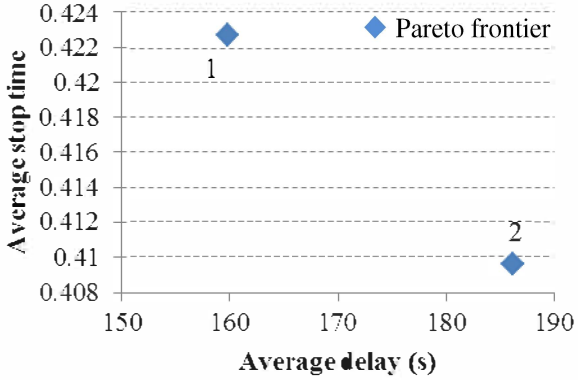

Figure 6. Pareto frontier for the 4-interection lattice road network

When we test NSGA-II with GPUs here, we obtain a speedup factor of 21.46, which is showed in Table III.

TABLE III. Comparison for the 4-intersection lattice road network

	CPU	GPU with CPU	Speedup
Average time/s	1111.24	51.78	**21.46**
Standard deviation	9.45	2.37	N/A

For the Zhongguancun area road network, we also use the roads that direct into the network as inlets. We keep all the parameters the same. The convergence process of the iteration calculation and the distribution of Pareto frontier are shown in Figure 7 and Figure 8 respectively. And the speedup factor is showed in Table IV.

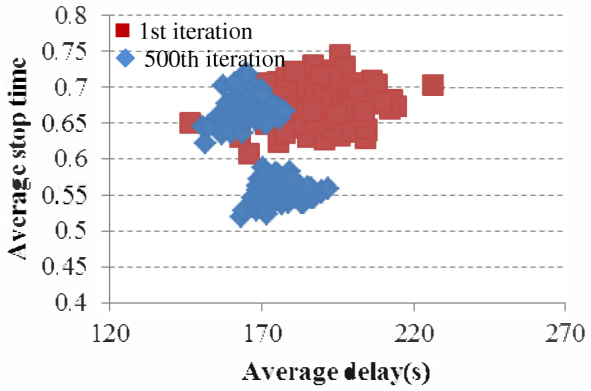

Figure 7. The convergence process for the Zhongguancun area

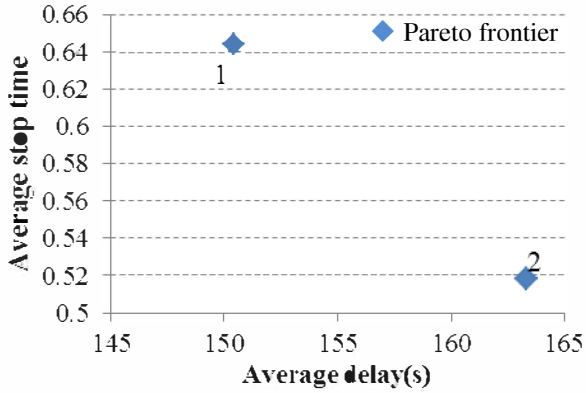

Figure 8. Pareto frontier for the Zhongguancun area road network

TABLE IV. COMPARISON FOR THE 18-INTERSECTION ZHONGGUANCUN AREA ROAD NETWORK

	CPU	GPU with CPU	Speedup
Average time/s	3886.67	140.62	**27.64**
Standard deviation	27.35	2.97	N/A

From the simulation above, we can see that NSGA-II has a good performance for the multi-objective problem. GPUs have a good effect on reducing the computational burden.

V. CONCLUSION

In this paper we applied the Non-dominated Sorting Genetic Algorithm II method to solve a two-objective optimization problem. The two objectives are the average delay time and the average stop times. We test the method on a 4 intersection lattice road network, and the 18 intersection Zhongguancun area road network. We achieve speedup factors of 21.46 and 27.64 respectively. NSGA-II is more like enumerating for the small case. For a larger problem, the computation burden goes heavier. In the future, we aim to improve NSGA-II to solve larger and more practical problems based on GPU clusters, and provide guidance for the transportation management and control.

REFERENCES

[1] F. Y. Wang, "Parallel Control and Management for Intelligent Transportation Systems: Concepts, Architectures, and Applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, 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] 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.

[4] L. Li, K. Yang, Z. Li, and Z. Zhang, "The optimality condition of the multiple-cycle smoothed curve signal timing model," *Transportation Research Part C: Emerging Technologies*, vol. 27, pp. 46-57, 2013.

[5] A. T. Chronopoulos, G. Wang, "Parallel solution of a traffic flow simulation problem," *Parallel Computing*, vol. 22, pp. 1965-1983, 1997.

[6] A. T. Chronopoulos, C. M. Johnston, "A real-time traffic simulation using a communication latency hiding parallelization," *IEEE Transactions on Vehicular Technology*, vol. 51, no. 3, pp. 498-510, May 2002.

[7] L. Li, D. Wen, N. Zheng, and L. Shen, "Cognitive cars: A new frontier for ADAS research," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, pp. 395-407, 2012.

[8] Z. Shen, K. Wang and F. Zhu, "Agent-based traffic simulation and traffic signal timing optimization with GPU," *Proceedings of the 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC),* pp. 145-150, Washington D. C., 2011.

[9] K. Wang and Z. Shen, "Artificial societies and GPU-based cloud computing for intelligent transportation management," *IEEE Intelligent Systems*, vol. 26, pp. 22-28, 2011.

[10] K. Wang and Z. Shen, "A GPU-Based Parallel Genetic Algorithm for Generating Daily Activity Plans," *IEEE Transactions on Intelligent Transportation Systems,* vol. 13, pp. 1474-1480, 2012.

[11] K. Wang and Z. Shen, "A GPU based traffic parallel simulation module of artificial transportation systems," *2012 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI'12)*, pp. 160-165, Jul. 8-10, Suzhou, China, 2012.

[12] K. Wang and Z. Shen, "GPU based ordinal optimization for traffic signal coordination," *2012 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI'12),* pp. 166-171, Jul. 8-10, Suzhou, China, 2012.

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

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

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

[16] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, issue 2, pp. 182-197, 2002.

[17] D. E. Knuth, "A generalization of Dijkstra's algorithm," *Information Processing Letters*, vol. 6, pp. 1-5, 1977.

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

978-1-4799-2914-613/$31.00 ©2013 IEEE