Application of Vector Ordinal Optimization to the Transportation Systems with Agent Based Modelling*

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

State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, and Dongguan Research Institute of CASIA, Cloud Computing Center, Chinese Academy of Sciences No. 95 Zhongguancun E. Road, Beijing 100190, China

{zhen.shen & feiyue.wang}@ia.ac.cn

K. Wang

Centre for Military Computational Experiments and Parallel Systems Technology, and College of Mechatronics Engineering and Automation, National University of Defense Technology, Changsha 410073, Hunan Province, China

kai.wang_nudt@hotmail.com

Abstract - **As the computing technology develops, micro-simulation becomes more and more important in the Intelligent Transportation Systems (ITS) research, because it can provide detailed descriptions of the system. However, for a multi-agent systems (MAS) modelling of an ITS, the computation burden is large, as it involves the computation of the state changing of all the agents. Further, if we consider simulation based optimization, which can be simply understood as an intelligent way of running a number of micro-simulations, the computation burden is huge. Moreover, there are multiple objective optimization problems in the ITS. The Vector Ordinal Optimization (VOO) method is a powerful tool for multi-objective optimization. In this paper, we apply VOO to the problem of optimizing the stop times and delay time of an ITS. We test the method on a 4 intersection lattice road network, and on the 18 intersection road network of the Zhongguancun area of Beijing. Compared with the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) method, the VOO method can achieve a speedup of factor of more than 150, with only a little sacrifice of performance.**

Index Terms - **Ordinal Optimization; Vector Ordinal Optimization; Agent-based modelling; Intelligent Transportation Systems**

I. INTRODUCTION

Traffic congestion is a notorious problem for almost all the big cities in the world. The transportation system does not only involve in vehicles, roads, pedestrians, traffic lights, but also infrastructure, logistic, weather and environmental, legal and regulatory, social and economic, ecological and resource factors. To take good use of all related technologies especially the information and communication technologies, the Intelligent Transportation Systems (ITS) [1-10] is initialized, developed and deployed over the past several decades and greatly improve the transportation efficiency and safety, and travel reliability.

The traffic simulation plays an important role in evaluating various traffic and management methods in ITS. 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 and Lattice Boltzman Method (LBM) method. These models are good at describing the overall properties of the traffic flows but lack the flexibility to describe the complex microscopic behaviors such as lane changing and vehicle overtaking. In recent years, as the development of computing technologies, the micro-simulation methods such as Cellular Automata (CA), and Multi-Agent Systems (MAS) [5, 7, 8] become more and more popular. 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 "grow up" in a bottom up fashion [2]. A lot of commercial and academic traffic simulation software such as TRANSIMS, PARAMICS, MATSIM and TransWorld [8] provide micro-simulation modules. And, more and more Metropolitan Planning Organizations (MPOs) have already used micro-simulation to help make decisions.

However, 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. If we take a "simulated based optimization" approach, which can be seen as an intelligent way of running the micro-simulation systems for many times, the computation burden can be very huge.

Ordinal Optimization (OO) [9, 11-14] is an efficient method for the optimization problems in complex systems research. There are two main ideas of OO. The first is ordinal comparison, that is, when comparing the performances of two designs, knowing which is better is much easier than knowing how good one design is than the other. The second is goal softening, that is, for many cases, we only need to obtain good enough designs, rather than obtain the design which is the best for sure. There have been many successful applications of OO

* 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. .

to problems such as re-manufacturing problems, apparel manufacturing scheduling, etc. Vector Ordinal Optimization (VOO) [12-14] is the extension for solving multi-objective problems.

In this paper, we try to solve the traffic lights signaling optimization problem. There are two objectives: stop times and delay 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 compare the results with Non-dominated Sorting Genetic Algorithm II (NSGA-II) [10], which is a famous method for multi-objective optimization. The results show the VOO can achieve a speedup factor of more than 150 with only a little sacrifice of performance. As far as we know, this is the first time to apply VOO to the agent-based modelling of a transportation system. The results show that it is promising to use VOO to solve the multi-objective optimization problems of a large and more real road network.

II. Literature Review

A. Traffic Micro-simulation and Optimization

Although for the MAS model almost everything can be modeled as an agent, people usually begin with the most elementary factors of vehicles, roads and traffic lights. Usually a graph is used to describe the topology of the road network. The nodes and links represent the intersections and roads. There are many car following models, among which GM-type model [15] 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 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.

For the lane changing model, Gipps model [16] is a well-known one. The whole lane changing process can be divided into three steps in the model: generating lane-changing intentions, choosing a destination lane and judging the feasibility of lane changing, and performing lane changing. According to different lane-changing intentions, the model can be classified into two types: one is discretionary lane changing that aims at travelling with desired speed, and the other is compulsory lane changing that aims at travelling in one of the allowed lanes for the desired turning movement in order to follow the planned path. For one specific vehicle agent travelling in a lane, it can perform discretionary lane changing when it has a long distance from the intersection and it can perform compulsory lane changing when it has a not so long distance from the intersection. When the agent approaches the intersection, it cannot change its lane and has to re-plan its path from the current intersection to its destination if it is not in the lane for the desired turning movement. If the lane-changing intention is generated, the agent has to choose its destination lane and judge the feasibility of lane changing. The agent chooses a lane that brings it closer to its desired speed when it performs discretionary lane changing or chooses a lane that brings it to the next intersection of its planned route when it performs compulsory lane changing. The lane changing is feasible only if the deceleration required for the subject agent to move behind the new leader in the destination lane and the deceleration required for the new follower in the destination lane to allow the subject agent to move into the lane are both acceptable. The acceptable deceleration b_n is given by (2). When the actual required deceleration is larger than b_n, it is physically possible and safe for the vehicle agents to change lanes without an unacceptable risk of collision.

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

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 aggressivity parameter which models the drivers' different driving habits.

The authors have implemented both the two models on workstations with Graphics Processing Unit (GPU) installed, which is a powerful parallel computing tool. Please refer to [5, 8, 9] for more information.

Besides traffic simulation, there are many optimization problems in the ITS research. In [17], GA is used to solve the problem of traffic signal timing optimization with a 34% improvement over the mutually consistent solution of the problem. In [7], 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. OO and VOO

The OO method was invented by Ho et al. in 1992 [11] and now is an efficient method for complex systems research. For simulation based optimization, usually a problem can be formulated as follows,

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

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

generated according to ξ. Usually OO makes the assumption that the observed performance is the sum of true performance and a noise w,

$$\hat{J}(\theta)=J(\theta)+w. \quad (4)$$

This is usually given by a crude model. A crude model can be running the simulation for a few times and taking the average. There are usually two steps of applying OO. The first step is to uniformly sample out at least 1,000 designs from Θ to constitute the sample space N. The second step is to evaluate the designs in N with a crude model and select out some top designs to constitute the selected set S. The size of S is determined by the noise level and the type of the Ordered Performance Curve (OPC), which is obtained by ordering the designs in N according to the crude evaluations [12, 13]. The OPC reflects the internal relations among the designs and describes the problem structure. The OPC can be normalized and sorted into five types. Please see Fig. 1.

OO defines top n% (usually 5%) of Θ as the good enough set, denoted as G_Θ. According to the theory of OO, OO can guarantee the Alignment Probability (AP) to be at least $P_{A,0}$ (usually 95%),

$$AP = P\{|S \cap G_\Theta| \geq k\} \geq P_{A,0}, \quad (5)$$

where k is called the alignment level.

VOO extends OO 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 $J_i(\theta_1) \leq J_i(\theta_2)$, 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.,

$$L_1 \equiv \{\theta \mid \theta \in \Theta, no\ \theta' \in \Theta, \text{ s.t. } \theta' \prec \theta\}. \quad (6)$$

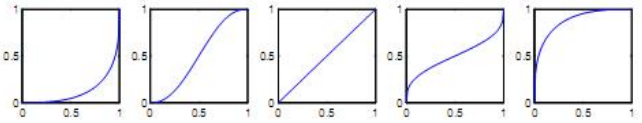

Fig. 1. Five types OPC: Flat, U-Shaped, Neutral, Bell and Steep

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

$$L_{s+1} = \Omega(\Theta \setminus \cup_{i=1,\ldots,s} L_i), s = 1, 2, \ldots \quad (7)$$

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. The ordered performance curve of VOO is called "VOPC" where "V" stands for "vector". It is a one-dimensional function $F(x)$ where x is the layer index, ranging from the 1 to the total number of layers of that problem, and $F(x)$ is the sum of the number of designs in layers from the 1st to the x-th layer. There are usually three kinds of VOPCs. Please see Fig. 2.

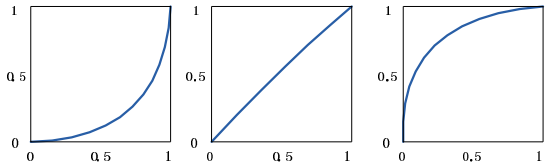

Fig. 2. Three types of VOPC: flat, neutral and steep

For VOO, the top g layers are defined as good enough set G_Θ and we need to determine the selected set S, where is constituted by top s layers. The alignment probability is defined as,

$$AP = P\left\{\left|\cup_{i=1}^{s} \hat{L}_i \cap \cup_{i=1}^{g} L_i\right| \geq k\right\} \geq P_{A,0}, \quad (8)$$

where $\hat{L}_i$ is the estimated i-th layer, obtained by using the crude model.

III. Problem Description and Application of VOO

Except the multi-objective, the formulation is the same with the one presented in [5, 8, 9], which are papers of the authors of this paper. For readers' convenience, we give a concise description here.

A. Road Network Modelling

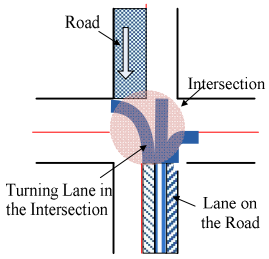

Fig. 3. An intersection

For every road there are usually several lanes. Please see Fig. 3. 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 the vehicle follow a given path when turning.

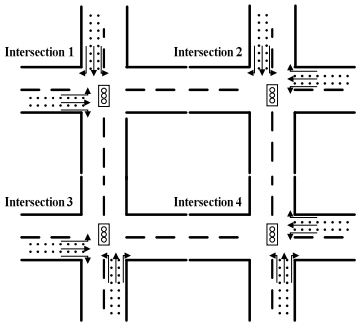

Fig. 4. A 4 intersection road network

We index the intersections in the road network arbitrarily from 1 to I. The road network used in this paper is shown in Fig. 4. We use the GM-type car-following model and the

Gipps lane changing model, as we described in Section II.

For every intersection, there are several phases of the traffic lights, shown in Fig. 5. 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. Here we use the well-known Dijkstra's algorithm [18] for routing the vehicles. The algorithm uses the free travel time as the weights for roads if the vehicle travels only once, while it uses the average travel time in the past 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 phases of the intersections versus the reference intersection is denoted by a vector $\vec{\psi} = [\psi_1, \psi_2, \cdots, \psi_I]^T$. Obviously $\psi_1 = 0$, as we use the 1-st intersection itself as the reference intersection. We denote $\vec{\theta}_m = [\theta_{1m}, \theta_{2m}, \cdots, \theta_{Im}]^T$ (m = 1, 2, ..., M) with θ_{im} (i = 1, 2, …, I) as the green time of the m-th phase of the i-th intersection. The problem can be formulated as follows,

$$\begin{aligned} &\min F = [f_1(c, \vec{\psi}, \vec{\theta}_1, \vec{\theta}_2, \cdots, \vec{\theta}_M), f_2(c, \vec{\psi}, \vec{\theta}_1, \vec{\theta}_2, \cdots, \vec{\theta}_M)]^T \\ &s.t. \quad c_{\min} \le c \le c_{\max} \\ &\qquad 0 \le \vec{\psi} \le c e_I \\ &\qquad \vec{\theta}_m \ge \theta_{\min,m} e_N, m = 1, 2, \cdots, M, \end{aligned} \tag{9}$$

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.

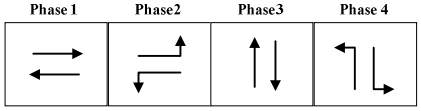

Fig. 5. Phase sequence of the traffic lights

D. Encoding and Decoding of Designs

In order to apply VOO and 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 that is 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,

$$\phi_m = \sum_{k=0}^{7} 2^k b_{mk}, m = 1, 2, \cdots, M + 2, \tag{10}$$

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.

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	$\begin{cases} p_{im} = p_{\min} + \phi_{m+2} \frac{p_{\max} - p_{\min}}{255}, m = 1, 2, ..., M \\ \theta_{im} = \theta_{\min,m} + \frac{p_{im}}{\sum_{j=1}^{M} p_{ij}} (c - \sum_{j=1}^{M} \theta_{\min,i}), m = 1, 2, ..., M \end{cases}$

E. Algorithm Flow

The application of VOO is described as follows.

Step 1: Initialization. Set the size of sample space N (usually 1,000), the number of good enough layers g, the number of replications T_c used in the crude model and T_a in the more accurate model, the alignment level k, the alignment probability level $P_{A,0}$ (usually 0.95). For steep type VOPC $g \in [1,10]$, for neutral type $g \in [1,20]$, for flat type $g \in [1,44]$. For all types of VOPCs, $k \in [1,100]$.

Step 2: Take out uniform samples from Θ to obtain N and evaluate designs in N by the crude model, and then obtain the layers, and the VOPC type.

Step 3: Obtain the size of the estimated selected set $\hat{s}_0$, adjust g, k according to the empirical formulas to,

$$g' = \max\{1, \left\lfloor \frac{100}{\hat{s}_0} \times g \right\rfloor\} \tag{11}$$

$$k' = \max\{1, \frac{10000}{|N|} \times k\}, \tag{12}$$

where $\lfloor \bullet \rfloor$ is the floor operator.

Step 4: According to the formula,

$$s' = Z(k', g') = e^{z_1} k'^{z_2} g'^{z_3} + Z_4, \quad (13)$$

we obtain s'. The parameters are given by Table III [12, 13].

Table III. Parameters for VOO

	Flat	Neutral	Steep
Z_1	5.2099	0.9379	0.3885
Z_2	0.9220	0.8445	0.8536
Z_3	-1.9542	-0.8890	-0.8847
Z_4	1.9662	0.5946	0.5414

Finally the size of selected size S is given by,

$$s = \left\lceil \frac{\hat{s}_0}{100} \times s' \right\rceil, \quad (14)$$

where $\lceil \bullet \rceil$ is the ceiling operator.

Step 5: Evaluate the designs in S with T_a replications of simulations, which can be viewed as the more accurate model.

IV. Experiments

We test VOO on two cases. Case 1 is the 4 intersection road network given in Fig. 4. Case 2 is the 18 intersection road network of Zhongcuancun area of Beijing. Please see Fig. 6.

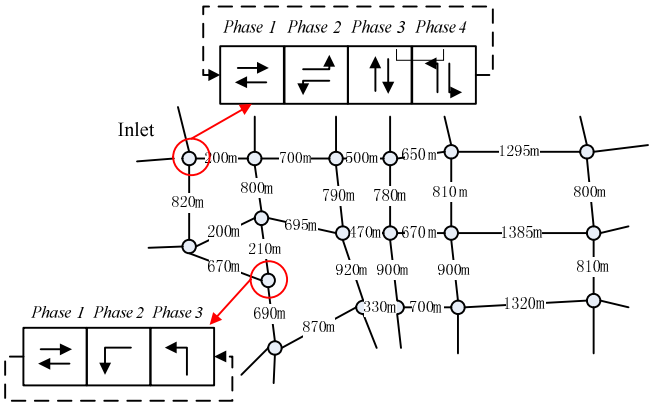

Fig. 6. Zhongguancun area road network

For the 4 intersection lattice road network, the distance between neighbouring 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). We use T_c = 1 replication for the crude model and T_a = 10 replications as the more accurate model. We obtain the VOPC as shown in Fig. 7. It is the neutral type. All designs are divided into $\hat{s}_0$ = 58 layers. We set the good enough layer as g = 1, i.e., the Pareto frontier. And we set the alignment level k = 1. After adjustment according to (11) and (12) we obtain g' = 1 and k' = 10. According to (13) and Table III, we obtain s' = 17.72. According to (14) finally we obtain s = 11.

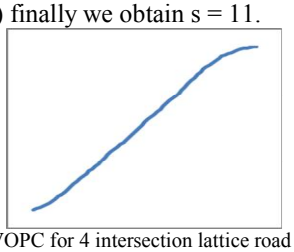

Fig. 7. VOPC for 4 intersection lattice road network

We show the results obtained by VOO together with results given by NSGA-II, for which we use 1,000 generations. Please see Fig. 8. The results of VOO are worse than NSGA-II. However, VOO takes only 307 s except the time for time needed for a person to ascertain VOPC type, while NSGA-II takes 51,780 s. The speedup factor is 168.66.

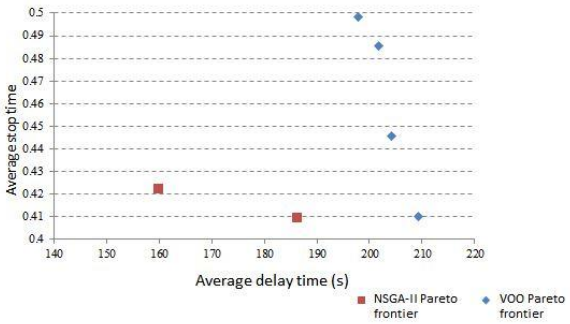

Fig. 8. VOO and NSGA-II for 4 intersection road network

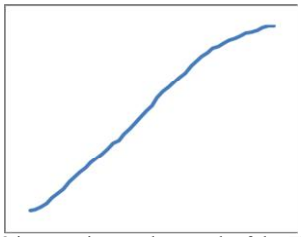

Fig. 9. VOPC for 18 intersection road network of the Zhongguancun area

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. We obtain $\hat{s}_0$ = 44 layers. We set g = 1 and k = 1 and we obtain g' = 2 and k' = 10. Following the

same steps, we obtain $s' = 10.24$ and $s = 5$. The results are shown in Fig. 10 together with the results of NSGA-II. The time for VOO and NSGA-II are respectively 613 s and 140,620 s. The speedup factor is 229.40.

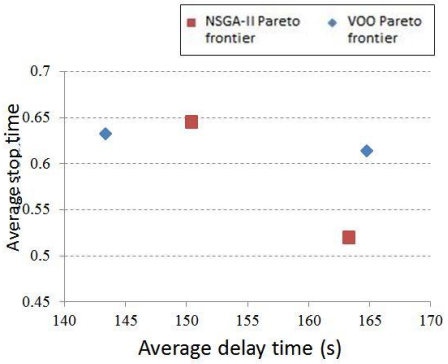

Fig. 10. VOO and NSGA-II for 18 intersection of Zhongguancun area

We see from the comparison that NSGA-II are better than VOO for the small problem, as NSGA-II is more like enumerating for the small case. However, for a larger problem, as the computation burden goes heavier, NSGA-II does not have much advantage over VOO, and the speedup factor of VOO over NSGA-II is larger. We could expect that when the problem scale goes up, the computation for NSGA-II becomes more formidable, and the usage of VOO can be more effective.

Of course, we admit that the comparison between VOO and NSGA-II are not totally fair, because the convergence rate becomes slower as NSGA-II runs and it can stop before the final generation is achieved. And we should test the methods on more cases. But we argue that the VOO is of good usage as it can give designs with similar performances at a much lower computation cost.

V. Conclusions

In this paper we applied the Vector Ordinal Optimization method to solve a two-objective optimization problem. The two objectives are the stop times and the average delay time. We test the method on a 4 intersection lattice road network, and the 18 intersection Zhongguancun area road network. We compare the results with the ones given by the Non-dominated Sorting Genetic Algorithm-II method. It shows that the VOO achieves a speedup factor of above 150 with a little sacrifice of the performance.

We choose GA for the comparison because GA is a very popular algorithm for optimization. In the future, we will try to compare different algorithms to verify the effects of VOO. Also, we will test on more cases and try to apply the VOO method to solving larger and more practical problems, and provide guidance for the transportation management and control.

Acknowledgment

The authors thank Dr. Shuangshuang Li, Prof. Fenghua Zhu, and Mr. Hang Gao of Chinese Academy of Sciences for helpful discussion.

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 trnsporttion sytems," *Complex Systems and Complexity Science*, vol. 1, no. 2, pp. 52-59, 2004 (in Chinese).

[3] 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.

[4] 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.

[5] Z. Shen, K. Wang and F. Zhu, "Agent-based traffic simulation and traffic signal timing optimization with GPU," *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC'11)*, pp. 145-150, 2011, Washington D. C., USA.

[6] 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.

[7] 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.

[8] 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, 2012, Suzhou, China.

[9] 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, 2012, Suzhou, China.

[10] Z. Shen, K. Wang and F. Y. Wang, "Application of NSGA-II to the agent based modelling of Intelligent Transportation Systems," *2013 IEEE International Conference on Intelligent Transportation Systems*, in print.

[11] Y. C Ho, R. S. Sreenivas and P. Vakili, "Ordinal Optimization of Discrete Event Dynamic Systems," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 2, pp. 61-88, 1992.

[12] Y. C. Ho, Q. C Zhao and Q. S Jia, *Ordinal Optimization: soft optimization for hard problems*, Springer, 2007.

[13] Q. C. Zhao, Y. Ho and Q. Jia, "Vector Ordinal Optimization," *Journal of Optimization Theory and Applications*, vol. 125, pp. 259-274, 2005.

[14] D. Li, L. Lee and Y. C. Ho, "Vector Ordinal Optimization-a new heuristic approach and its application to computer network routing design problems," *International Journal of Operations and Quantitative Management*, vol. 5, pp. 211-230, 1999.

[15] D. C. Gazis, R. Herman, and R. W. Rothery, "Nonlinear follow-the-leader models of traffic flow," *Operational Research*, vol. 9, pp. 545-567, 1961.

[16] P. G. Gipps, "A model for the structure of lane-changing decisions," *Transportation Research Part B: Methodological*, vol. 20, no. 5, pp. 403-414, 1986.

[17] 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, 2004.

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