High Performance Dynamic Traffic Assignment Based on Variational Inequality

Liang Geyu (HKUST), Gu Yangsong (CSUST)
Mentor: Liu Cheng (ORNL), Wong Kwai (UTK)
gliangad@connect.ust.hk, guyangsong.csust@outlook.com
cheng.n.liu@gmail.com, kwong@utk.edu

August 4, 2018

Abstract

Nowadays, dynamic traffic assignment (DTA) plays an increasingly important role in urban intelligent traffic planning. High efficiency of assignment becomes the ultimate target which many companies and researchers are pursuing for. Fortunately, parallel computing serves us as an effective tool to improve dynamic traffic assignment. DTA, as one kind of traffic assignment, aims at partitioning and then allocating the origin-destination demand to paths in order to minimize the travel cost for every user. In this project, we concentrate on modeling dynamic network loading(DNL) with the fixed-pointed algorithm and solving it in a highly efficient way. And to achieve that goal, the tool of parallel computing we applied is openMp. During this project the DNL process can be classified into two categories, namely ODE and PDE model. As a result, DTA based on ODE model, is successfully solved by parallel computing while PDE model still requires much work to do in the future.

1 Introduction

Dynamic traffic assignment(DTA) is usually viewed as the positive(descriptive) modeling of time varying flows on vehicular networks consistent with established traffic flow. The models of dynamic traffic assignment are determined by departure rates, departure times and route choices over a given space-time interval. They seek to reproduce the traffic flow propagation and dynamic evolution in networks. As a kernel component of dynamic traffic assignment, dynamic network loading (DNL) determines the assignment result directly. In this paper we mainly concentrate on two DNL models which are ordinary differential equations (ODEs) and partial differential equations (PDEs), and the latter is also called Lighthill-Whitham-Richards (LWR) model. The principle that is followed throughout this paper is known as dynamic user equilibrium (DUE) where the unit travel cost, including early and late arrival penalties, is consistent with path and departure time selections by users between certain origin-destination(OD) pair.

1.1 Description of DTA

We're using this section to describe the basic theory of DTA and explain some traffic terms. The entire model is based on a given traffic network, such as the simple network showed in Firgure1(right). In general, a network is consisted of a set of nodes and links while several directed links could be grouped to a path. Each origin-destination(OD) pair may consist one or more paths, and each link is typically associated with some impedance that affects the inflow and outflow. Therefore, the traversal time spent on each path differs from each other. Dynamic user equilibrium(DUE) principle is proposed according to the Wardrop theory. It can also be viewed as a result of Nash equilibrium. Under this principle, users will choose the path that costs the minimum time at an appropriate departure time. So this kind of dynamic user equilibrium(DUE) assignment, which is illustrated in Figure1(left), finds the flow pattern by allocating the OD demands to the network in such a way that no drivers will change routes for achieving better travel choice(Sheffi,1985).

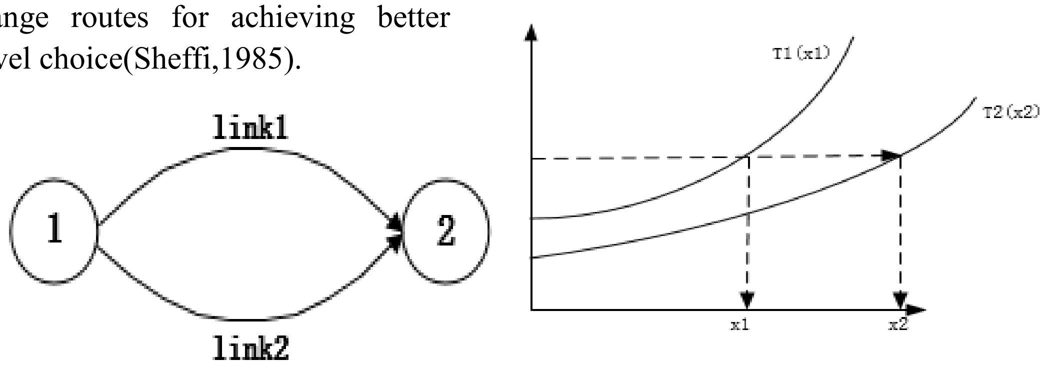

Figure 1: Equilibrium in a simple network: Left: a two-link network with one OD pair. Right: the travel time on both links are equal when the link 1 is assigned for flow x1 while the link 2 is x2.

1.2 A brief review of DTA

Many research teams have made plenty of research in dynamic traffic assignment and achieved great progress in modeling and computation for solving DTA. Dynamic user equilibrium models, as noted by Peeta and Ziliaskopoulos (2001), tend to be comprised of the following four submodels:

(1) A model of path delay;
(2) Flow dynamic;
(3) Flow propagation constraints;
(4) A path/departure-time choice model.

Merchant and Nemhauser (1978a,b) proposed a flow conservation model based on ordinary differential equations. Ran et al.(1993) modified the MN model by treating both entrance and exit flows as control variables. After that, "first in first out" principle was added to model to make model more practical (Ran et al.1993 and Boyce 1996). Friesz et al.(1993) proposed the link dynamic state function based on exit time function which is easier to obtain compared with MN model. Despite these continuous models, simulation such as cell transmission model (CTM) also shows the great advantages in solving models (Daganzo,1994). In recent years, due to the development of Lighthill-Whitman-Richards (LWR) theory, Friesz et al.(2013) and Han et al. (2014) re-described the flow propagation by hydrodynamic model and solve it by H-J equations. To overcome the discontinuous in cumulative entering flow and exit flow function, Han et al.(2014) proposed a continuous-time link-based kinematic wave model for dynamic traffic networks.

1.3 Organization

The rest of this paper is organized as follows. In section 2, we discuss the dynamic network loading by ordinary differential equations presented in Friesz et al.(2011). By solving the ODE equations, we can obtain the link volume. Then, the fundamental operator of path delay will derivate from link delay. In section 3 we introduce differential variational inequalities (DVI) to formulate DUE principle, and then it is solved by being converted to fixed-point problem. To fasten the speed of solving DTA and enlarge the network scale, we used the openMP, which is a practical tool to enable parallel computing. In section 4, we take small and SiouxFalls network for example to test the algorithm and also the advantages of using parallel computing. In section 5, we will present the LWR model, the difference compared with ODE is the dynamic network loading part. It will be solved by simulation in a discrete way. Link or path travel time could be obtained directly from the result. And the following part of solving fixed-point problem is the same with ODE model.

2 Dynamic network loading by ODEs

Dynamic network loading is a crucial step to find equilibrium. And it should be established in a way which will return the link volume when given travel demand and path departure rates efficiently. ODE model is comparatively easy to implement using computer.

2.1 Dynamic algebraic systems (DAEs)

Let us consider a road network described by a directed graph (A,V), where A,V denotes the set of links and nodes, respectively. Then we begin our work by introducing the following key notations used in the derivation of the DAE system:

G(A,V):	the original network with node set V and link set A;
R	the set of origins in the augmented networks;
S	the set of destination in the augmented networks;
K_{rs}	the set of paths connect O-D pair r-s, $r \in R, s \in S$;
W	the set of OD pair;
$h_p(t)$	the departure rate on path p at t;
$x_a^p(t)$	the flow on arc *a* associate with path p;

$r_a^p(t)$	the first derivative of exit flow function
$g_a^p(t)$	the exit flow function of arc *a* associated with path p
$R_a^p(t)$	the second derivative of exit flow function

The proposed DAE system (Freisz,2011) then reads:

$$\frac{dx_{a_1}^p(t)}{dt} = h_p(t) - g_{a_1}^p(t) \quad \forall p \in P \tag{2.1}$$

$$\frac{dx_{a_i}^p(t)}{dt} = g_{a_{i-1}}^p(t) - g_{a_i}^p(t) \quad \forall p \in P, i \in [2, num(p)] \tag{2.2}$$

$$\frac{dg_{a_i}^p(t)}{dt} = r_{a_i}^p(t) \quad \forall p \in P, i \in [1, num(p)] \tag{2.3}$$

$$\frac{dr_{a_1}^p(t)}{dt} = R_{a_1}^p(x, g, r, h) \quad \forall p \in P \tag{2.4}$$

$$\frac{dr_{a_i}^p(t)}{dt} = R_{a_i}^p(x, g, r) \quad \forall p \in P, i \in [2, num(p)] \tag{2.5}$$

$$x_{a_i}^p((\tau - 1) \cdot \Delta) = x_{a_i}^{p,0} \quad \forall p \in P, i \in [1, num(p)] \tag{2.6}$$

$$g_{a_i}^p((\tau - 1) \cdot \Delta) = 0 \quad \forall p \in P, i \in [1, num(p)] \tag{2.7}$$

$$r_{a_i}^p((\tau - 1) \cdot \Delta) = 0 \quad \forall p \in P, i \in [1, num(p)] \tag{2.8}$$

Where the $R_{a_i}^p(x, g, r)$ was fold by second-order Taylor formulation.

$$R_{a_i}^p(x, g, r) = \frac{2g_{a_{i-1}}^p(t)}{(D_{a_i}[x_{a_i}(t)])^2(1 + D_{a_i}^{'}[x_{a_i}(t)]x_{a_i}^{'}(t))} - \frac{2(g_{a_i}^p(t) + r_{a_i}^p(t))D_{a_i}[x_{a_i}(t)]}{(D_{a_i}[x_{a_i}(t)])^2} \tag{2.9}$$

For initial condition, $g_{a_{i-1}}^p(t) = h_p(t)$ is always satisfied if only if the exit flow belongs to first link.

2.2 Path delay operator

Path delay operator is another crucial ingredient of the DUE model, which is composed of link delay. In general, link delay is a function of the arc volume and by convention it is appropriate to utilize BPR function to describe link travel time. But in this paper, we take a linear function for simplicity.

Given $h_p(t)$, arc volume will be solved from ODE, and the link travel time (delay) can be written as:

$$D(x_a) = \alpha x_a + \beta \tag{2.10}$$

$$\alpha = 1/\text{capacity} \tag{2.11}$$

$$\beta = 10^*\text{len/speed} \tag{2.12}$$

Where the fundamental diagram $D(\bullet)$ couples some arc features, namely capacity, length and speed. To each link, these variables are constants.

Therefore, we can derive the path travel time as the following form:

$$D_p(t,h) = \sum_{i=1}^{num(p)} [\tau_{a_i}^p(t) - \tau_{a_{i-1}}^p(t)] = \tau_{a_{num(p)}}^p(t) - t \tag{2.13}$$

$$\tau_{a_1}^p(t) = t + D_{a_1}[x_{a_1}(t)] \tag{2.14}$$

$$\tau_{a_i}^p(t) = \tau_{a_{i-1}}^p(t) + D_{a_i}[x_{a_i}(\tau_{a_{i-1}}^p(t))] \tag{2.15}$$

Where $\tau_{a_i}^p(t)$ is the time of flow exit link a_i which enters the network at t.

A common early or late arrival penalty function is added to stress the importance of arriving on time in real life, and we consider the effective path delay operators of the following form:

$$\Psi_p(t,h) = D_p(t,h) + 0.5 \times (t + D_p(t,h) - T_A)^2 \tag{2.16}$$

where T_A is the target arrival time.

From the derivation effective path delay, we can see that it is a progress of mapping the departure rate to path delay. In other words, given $h_p(t)$ to each path, then we will obtain the coordinate path delay.

3 Implementation

3.1 Mathematics implementation

DVI implement makes DUE principle formulate so that the DTA problem can be solved more conveniently. Than we rewrite the DUE problem according to the following theorem. We refer to Friesz(2011,2013) for detailed proofs.

Theorem 1. *Infinite-dimensional inequality formulation of DUE. The simultaneous departure-time-and-path-choice dynamic user equilibrium of Definition 1 is equivalent to the following differential* *variational inequality problem on* Λ *:*

$$\mathrm{DVI}(\Psi,\Lambda,[t_0,t_f]): \textit{find } h^* \in \Lambda_0 \textit{ such that}$$

$$\begin{cases} \sum_{p\in \mathrm{P}} \int_{t_0}^{t_f} \Psi_p(t,h^*)(h-h^*)dt \ge 0 \quad \forall h \in \Lambda \\ where \;\; \Lambda = h \ge 0 : \dfrac{dy_{ij}}{dt} = \sum_{p\in P_{ij}} h_p(t), y_{ij}(0)=0, y_{ij}(t_f)=Q_{ij} \end{cases} \tag{3.1}$$

Where $y_{ij}(0)=0, y_{ij}(t_f)=Q_{ij}$ is the boundary conditions. The solution of h is the final result that we want to achieve.

There are many ways to solve DVI, here we convert it to fixed-point problem:

$$h^* = P_\Lambda[h^* - \alpha\Psi_p(t,h^*)] \tag{3.2}$$

And for each iteration, we will get a v value by solving the following flow conversation equation:

$$\sum_{p\in P_{ij}} \int_{t_0}^{t_f} [h_p^k(t) - \alpha\Psi(t,h_p^k) + v_{ij}]_+ = Q_{ij} \tag{3.3}$$

Then the next input h^{k+1} can be denoted as:

$$\begin{aligned} h_p^{k+1} &= [h_p^k(t) - \alpha\Psi(t,h_p^k) + v_{ij}]_+ \\ Q_{23} &= 200 \end{aligned} \tag{3.4}$$

Where the diagram $[]_+$ is equal to $\max\{[],0\}$.If the difference between h_p^{k+1} and $h_p^k(t)$ is less than a pre-given tolerance, the iteration will be terminated.

3.2 Algorithm

Here is a piece of pseudo code for the algorithm described above and this is also the blue print of how our real code is implemented.

Algorithm 1 Computing ODE and fixed point iteration

```
Initialization: path, timespan, arc, h0, Q(demand), epsilon (tolerance) et.
for all i = 1 : num (OD pairs)
  while condition is true            ⅄ Difference between hk and hk+1 is larger than tolerance
    for j = 1 : num (paths)
      for k = 1 : num (links)
        solve ODE to get link volume                  ⅄ equations 2.1-2.8
```

end for

end for ⅄ end solving ODE

get link delay $D(x_{a_i}, h_p) \leftarrow x_{a_i}$

get effective path delay $Phi \leftarrow D(t,h) + F(t,h)$ ⅄equation 2.16

solve equation 3.3 get a v_{ij}

update hk+1 according to equation 3.4 ⅄each v map to a hk

end while

end for

Output: Phi, hk

3.3 High performance computation implementation

To enable computation of inputs with large graph size or large number of od pairs, we decided to integrate the parallel computing technology into our code. The ultimate goal of this implementation is using CUDA or other library concerning GPU. Here we only implement an early-stage version which is assisted by OpenMP library. OpenMP (Open Multi-Processing) is an application programming interface (API) that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran. In our case OpenMP or parallel computing can be used in the process of solving ODE (equation 2.1 to 2.8) and solving v for each od pair (equation 3.3), as these two processes happen to be the dominating cost resource in the final running time, the advantage of using parallel computing is prominent when the graph is enlarged and case is complicated to a certain degree, as we'll show in the next section.

4 Numerical example

4.1 Small network

We consider the small network summarized in Figure 2, with six-arc and five-node. There are two OD pairs $W = \{(1,3),(2,3)\}$ among which, the following six paths are employed:

$$p_1 = \{3,6\},\ p_2 = \{1,2,6\},\ p_3 = \{1,2,4,5\},\ p_4 = \{3,4,5\},\ p_5 = \{6\},\ p_6 = \{4,5\}$$

We assume a fixed demand for OD pair(1,3) and (2,3) such that $Q_{13} = 400$ and $Q_{23} = 200$, the target arrival time Ta is 75.0 and 50.0 out of a 0.0 to 100.0 time span, the result are shown in **Figure 2** and **Figure 3.**

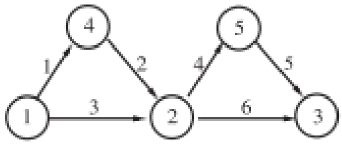

Figure 2

Arc	Jam density (vehicles/km)	Free flow speed (km/5min)	Length (km)
1	800	6	4
2	800	6	8
3	800	8	4
4	800	8	10
5	1000	8	8
6	600	6	10

Figure 3

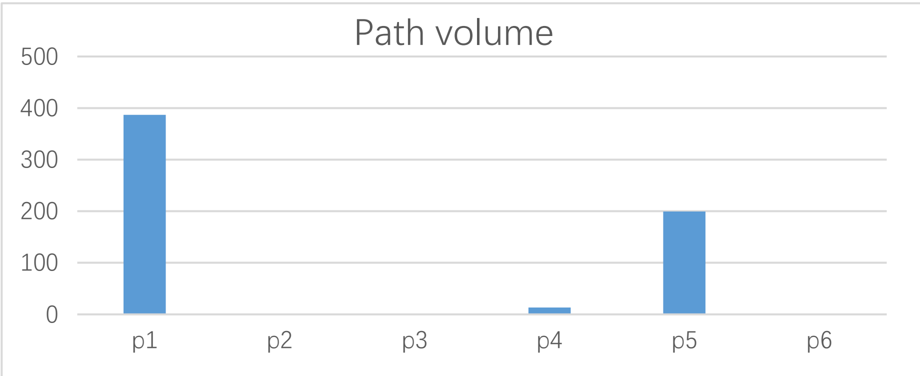

Figure 4

Figure 4 shows the accumulated path volume throughout the whole time span, which indicates the people's preference of paths.

Here we take a closer look at the first path's departure rate and unit cost function, which is shown in **Figure 5**.

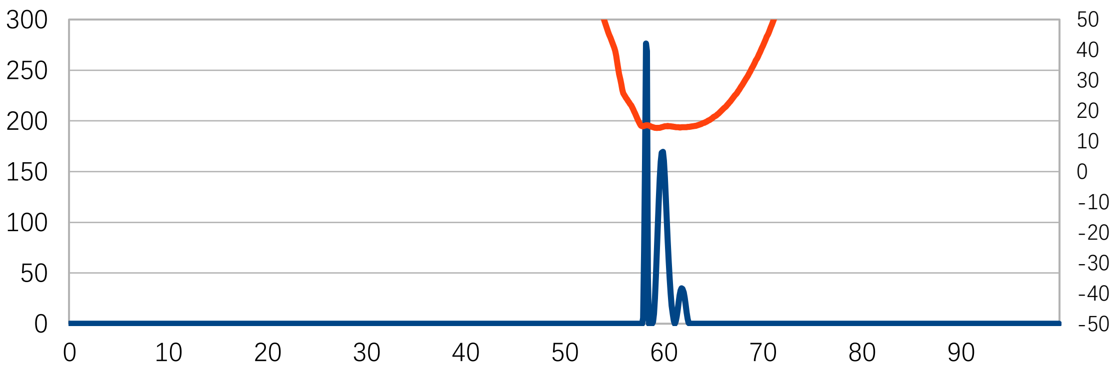

Figure 5

This graph exhibits the typical characters of DUE thus proves the accuracy of our algorithm. And it also provides some inspiration to traffic planning: sometimes entering the network batch by batch will help improve the travel efficiency.

4.2 Sioux Falls network

Sioux Falls network (**Figure 6**) is much larger than small network since there are 24 nodes, 76 links and thousands paths. Here we implement three different sets of data, which respectively contains 10 od pairs, 23 od pairs(all the od pair whose origin is node 1), and 552 od pairs (all the possible od pairs in this graph).

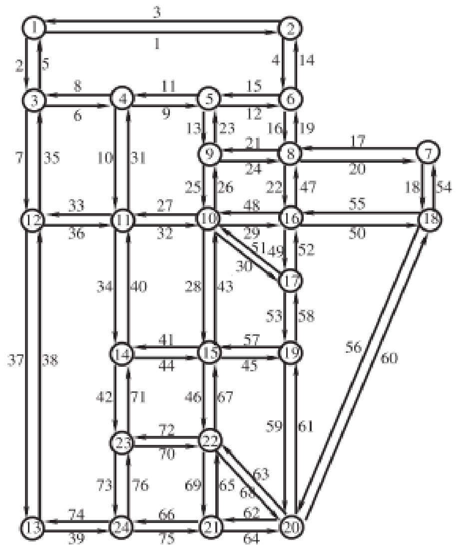

Figure 6

While the efficiency on each input will be discussed in the next subsection, here we'd like to show some randomly taken result. These results all behave as expected and prove the correctness of our algorithm.

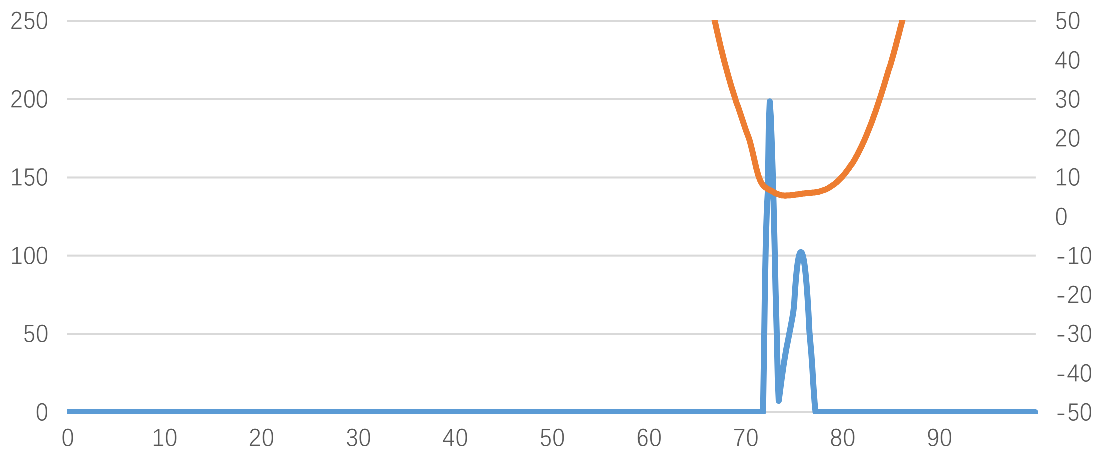

Figure 7

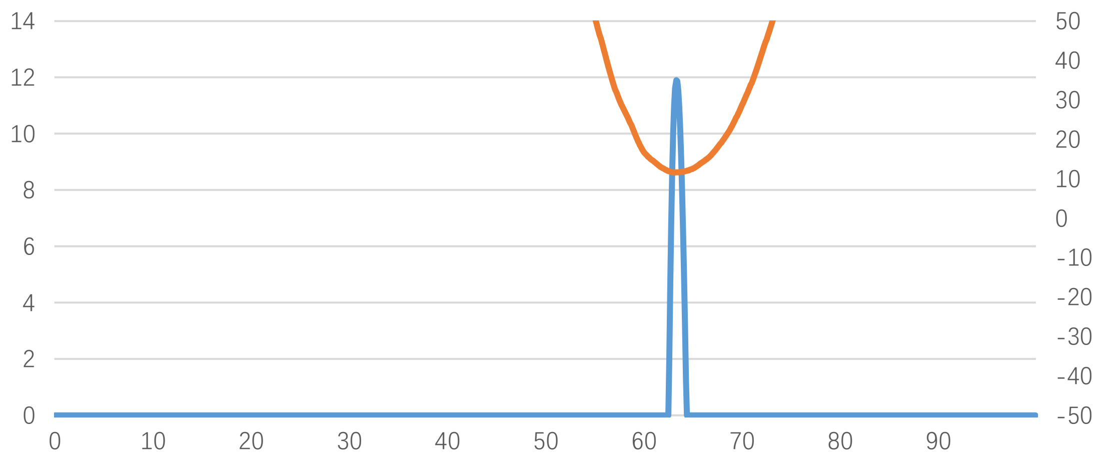

Figure 8

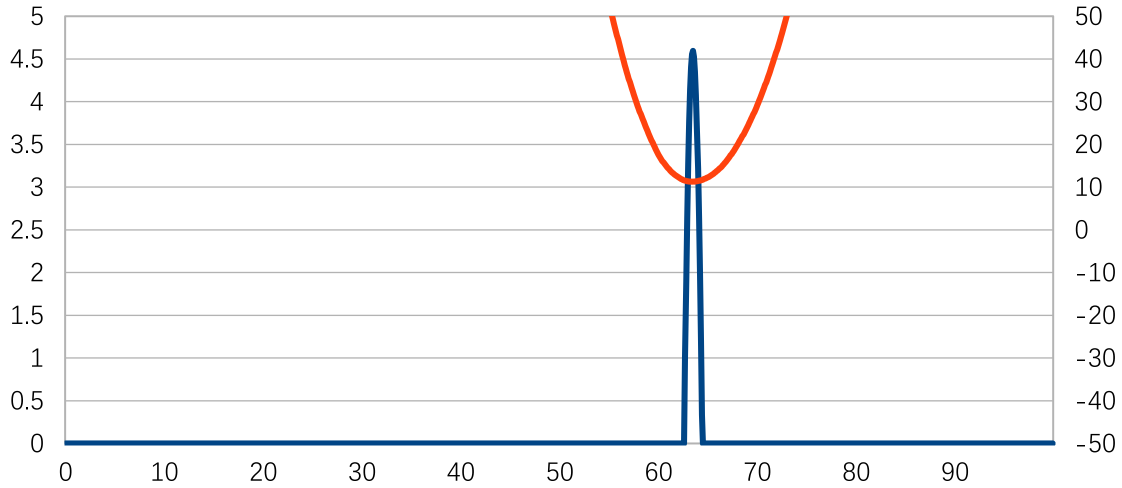

Figure 9

Figure 7: Departure rate and cost function of path No.30 in Siuoxfalls(10 pairs)
Figure 8: Departure rate and cost function of path No.40 in Siuoxfalls(23 pairs)
Figure 9: Departure rate and cost function of path No.6209 in Siuoxfalls(552 pairs)

4.3 OpenMP efficiency

We used the lab machine with 12 cores and OpneMP will have 24 parallel processes when invoked. We first apply openMP on small graph. Since there is only 2 od pairs in this graph, the improvement of speed is not obvious, as shown in **Figure 10**.

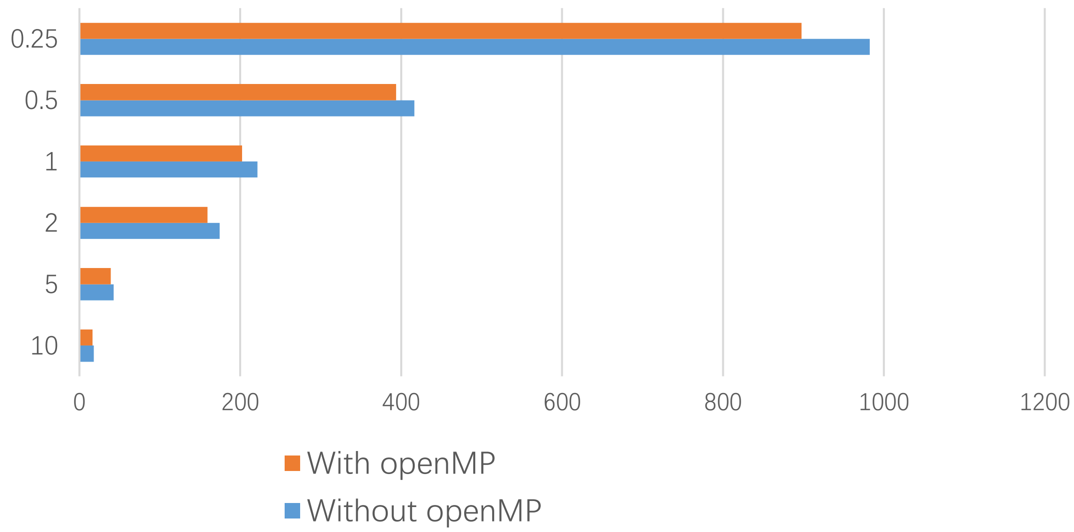

Figure 10, Epsilon vs Running Time on small graph

However, when we start to run data input which is much larger like siouxfalls, the advantage of parallel computing starts to show. The following table lists the average time per iteration for different input. Since we are considering the average time then the epsilon or iteration number is irrelevant.

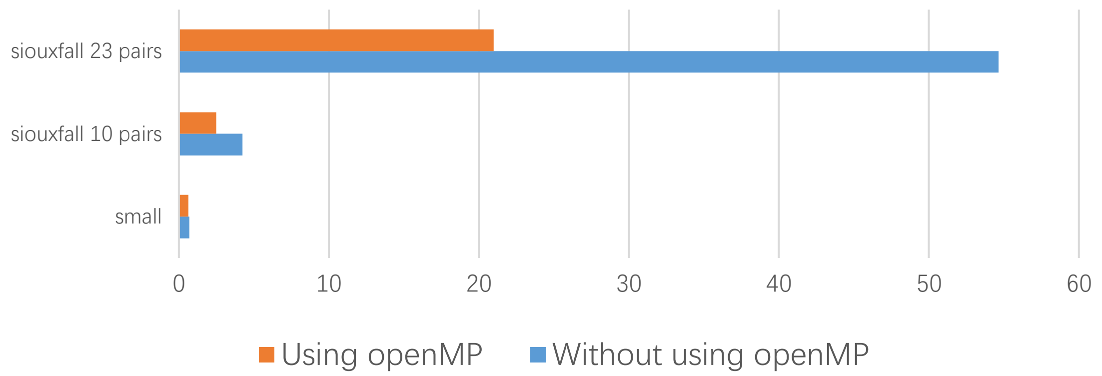

Figure 11

In conclusion, we find that the algorithm with OpenMP implemented also outperform the one without OpenMP, and the gap between them become larger with a larger input.

5 Dynamic network loading by PDEs

The PDEs model here actually refers to the LWR model which originates from hydrodynamic model(Friesz,2013). This is a microscopic model that can capture several network traffic phenomena:

(1). Queues and delay;

(2). Density-velocity relationship;

(3). First-in-first-out(FIFO); and

(4). Route information.

5.1 Partial dynamic algebraic systems (PDAEs)

The analytical solution to the PDE allows us to rewrite the network loading procedure as a system of differential algebraic equations (DAEs) instead of partial differential algebraic equations (PDAEs)(Friesz,2013). For convenience, we denote the following diagrams and variables.

$f(\cdot)$	The flow function of density
$\rho(\cdot)$	The density function of t, x
$\mu_a^p(\cdot)$	The proportion of link volumes in path p
$N_{down}^a(t)$	The cumulative exiting flow on link a
$N_{up}^a(t)$	The cumulative entering flow entering link a

$q_{in}^{a}(t)$	The flow rate entering link
$q_{out}^{a}(t)$	The flow rate exiting link
$D_a(t)$	The demand of flow rate out of link a
$S_a(t)$	The supply of flow rate into link a
k_a	The speed of forward flow propagation
w_a	The speed of backward flow propagation

The initial flow propagation can be described as the following form:

$$\frac{\partial \rho(t,x)}{\partial t}+\frac{\partial f(\rho(t,x))}{\partial x}=0 \tag{5.1}$$

Because of the relation between $f(\bullet)$ and $\rho(\bullet)$, the dynamic flow evolution formula (PDE) actually is a Nonlinear PDE, it's difficult to solve it directly. In other words, it can be solved by approximate ways (Friesz,2013) or discretization methods (Han,2012).

The proposed DAE system shown in 5.2-5.7 is given by Han(2012)
Part 1: link model

$$D_a(t)=\begin{cases} q_{in}^{a}(t-\frac{L_a}{k_a}) & if\ N_{up}^{a}(t-\frac{L_a}{k_a})=N_{down}^{a}(t) \\ C_a & if\ N_{up}^{a}(t-\frac{L_a}{k_a})>N_{down}^{a}(t) \end{cases} \tag{5.2}$$

$$S_a(t)=\begin{cases} q_{out}^{a}(t-\frac{L_a}{w_a}) & if\ N_{up}^{a}(t)=N_{down}^{a}(t-\frac{L_a}{w_a})+\rho_{jam}^{a}L_a \\ C_a & if\ N_{up}^{a}(t)<N_{down}^{a}(t-\frac{L_a}{w_a})+\rho_{jam}^{a}L_a \end{cases} \tag{5.3}$$

Part 2 Junction model

$$\alpha_{ij}^{J}(t)=\sum_{p\ni a,b}\mu_a^p(t,L_a) \tag{5.4}$$

$$q_{out,i}=\min\{D_i(t),\frac{S_j(t)}{\alpha_i}\}\ j\in I^o \tag{5,5}$$

$$q_{in,j}=\sum_{i\in I^v}\alpha_{ij}\bullet q_{out,i}(t) \tag{5.6}$$

Part 3 Link travel time

$$N_{down}^{a}(t)=N_{up}^{a}(\tau_a(t)) \tag{5.7}$$

Finally, the link travel time is equal to the difference between cumulative entering volume count and exiting volume count.

5.2 Computation implementations

According to the formulations proposed by Han et al.(2012,2014), the algorithm are mainly composed of two parts, which are link model and junction model. The pseudo code listed below are mainly composed of two parts, one is solving PDE and outputting link delay, the other is solving fixed point iteration and outputting best path cost function and departure rate function.

Algorithm 2: Computing dynamic network loading based on LWR model by C

Initialization: path, timespan, arc, h0, Q(demand), epsilon (tolerance) et.

for all i = 1 : num (OD pair)

    While condition is true ⅄ Difference between hk and hk+1 is larger than tolerance

      for t =1 : num (timesteps)

        for i = 1 : num (links)

Solve D Get link demand ⅄ equation 5.2

Solve S Get link supply ⅄ equation 5.3

for j = 1 : num (linkin)

for k = 1 : num (linkout)

get turning ratio $\alpha_{ij}^{J}(t)$ ⅄ equation 5.4

end for

end for

end for

Calculate entering flow $q_{in,j}$ ⅄ equation 5.5

Calculate exiting flow $q_{out,i}$ ⅄ equation 5.6

end for

get effective path delay $Phi \leftarrow D(t,h) + F(t,h)$

get a v_{ij} in each iteration ⅄ equation 3.3

update hk+1 according to equation 3.4 ⅄ each v map to a hk

end while

end for

Output: Phi, hk

5.3 Dynamic network loading

The calculation results can be seen in **Figure 12 and 13.** The Vertical difference between two horizontal lines denotes the length of link as is shown in left picture. In **picture 13**, Slant line illustrates the cumulative inflow and outflow, and the link travel time can be derivate from the horizontal difference of each two slant lines.

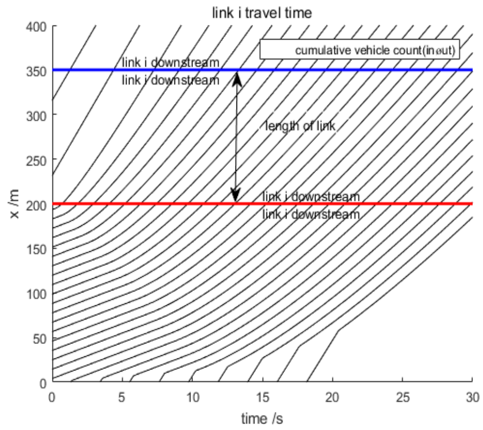

Figure 12 Flow propagation

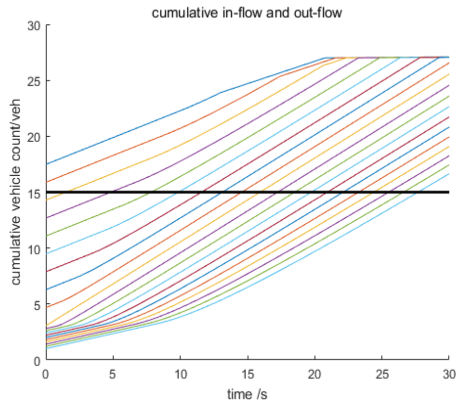

Figure 13 Cumulative vehicle count

6 Conclusion and remarks

In this paper, we have presented the whole progress of dynamic traffic assignment including dynamic network loading and fix point iteration. In the part of dynamic network loading , we introduced ODE model and PDE model which were respectively proposed by Friesz and Han. The purpose of this project is aimed at solving the dynamic traffic assignment using parallel computing and then apply it to large scale graph to satisfy strict time requirements in real traffic assignment.

For ODE model, we apply OpenMP in our code, which is one kind of parallel computing. We succeed and obtain good assignment results. The parallel computing significantly fastened the solving speed as the same result compared with normal computation. In this way, it's possible to apply this technology to computing big graphs and urban network planning.

For PDE model, it remains many works to do. We haven't finish the entire code, but thanks to the Han (2014), we proposed the pseudo code, and the next step is realizing it by C. Finally, if possible, parallel computing will be also used to certify the high performance in dynamic traffic assignment.

References

Daganzo, C., 1994. The cell transmission model. Part I: a simple dynamic representation of highway traffic. Transportation Research Part B 28 (4), 269–287.

Friesz, T., Han, K., Neto, P., Meimand, A. and Yao, T. (2013). Dynamic user equilibrium based on a hydrodynamic model. Transportation Research Part B: Methodological, 47, pp.102-126.

Friesz, T., Kim, T., Kwon, C. and Rigdon, M. (2011). Approximate network loading and dual-time-scale dynamic user equilibrium. Transportation Research Part B: Methodological, 45(1), pp.176-207.

Friesz, T. (2014). Dynamic optimization and differential games. [Place of publication not identified]: Springer-Verlag New York.

Han, K., Piccoli, B., Friesz, T. L., & Yao, T. 2012. A Continuous-time Link-based Kinematic Wave Model for Dynamic Traffic Networks. Arxiv e-prints, Aug.

Han, K., Friesz, T.L., Yao, T., 2013a. A partial differential equation formulation of Vickrey's bottleneck model, part I: Methodology and theoretical analysis. Transportation Research Part B 49, 55–74.
Han, K., Friesz, T.L., Yao, T., 2013b. A partial differential equation formulation of Vickrey's bottleneck model, part II: Numerical analysis and computation. Transportation Research Part B 49, 75–93.
Han, K., Piccoli, B. and Szeto, W. (2015). Continuous-time link-based kinematic wave model: formulation, solution existence, and well-posedness. Transportmetrica B: Transport Dynamics, 4(3), pp.187-222.
Han, K., Piccoli, B. and Friesz, T. (2016). Continuity of the path delay operator for dynamic network loading with spillback. Transportation Research Part B: Methodological, 92, pp.211-233.
Han, K., Friesz, T.L., Yao, T., 2014. Vehicle spillback on dynamic traffic networks and what it means for dynamic traffic assignment models. 5th International Symposium on Dynamic Traffic Assignment. Salerno, Italy, 17-19 June 2014.
Lighthill, M. and Whitham, G. (1955). On kinematic waves. [London]: [Royal Society].
Programming Methods
Merchant, D., Nemhauser, G., 1978a. A model and an algorithm for the dynamic traffic assignment problems. Transportation Science 12 (3), 183–199.
Peeta, S., Ziliaskopoulos, A., 2001. Foundations of dynamic traffic assignment: the past, the present and the future. Networks and Spatial Economics 1 (3),233–265.
Ran, B., Boyce, D., 1996. Modeling Dynamic Transportation Networks: An Intelligent Transportation System Oriented Approach. Springer-Verlag, New York.
Yosef, S., URBAN TRANSPORTATION NETWORKS (1985): Equilibrium Analysis with Mathematical, 15-25.