High Performance Dynamic Traffic Assignment Based on Variatonal Inequality

Students: Liang Geyu(HKUST) Gu Yangsong(CSUST)
Mentors: Cheng Liu (ORNL), Kwai Wong (UTK)

Abstract

In this paper we present a dynamic traffic assignment(DTA) based on continuous-time network loading procedure. A system of differential algebraic equations(DAEs) is proposed for describing traffic flow propagation, travel delay, and route choices. We THEN convert dynamic user equilibrium(DUE) principle to differential variational inequalities(DVI) which will be solved by fixed-point algorithm. To adapt to large scale network and GPU, we program the whole model using C, so that we can solve it efficiently on supercomputers. The solver will be implemented on GPU using CUDA.
There are two kind of DAE systems proposed in this project, one is ODEs, the other is PDEs representing the LWR model. They have their own advantages on describing traffic flow and propagation. The former is mainly based on setting link state function while the later is based on hydrodynamics theory. Given a departure rate of each path associated with OD pairs, it will output the link volume or traversal time.
.

DNL by ODEs

DTA models determine departure rates, departure times and route choices over a given planning horizon.

Dynamic Network Loading(DNL) is a critical procedure to obtain the traffic delay operator, which maps a departure time to a traversal time. DNL constructed by ODEs can be described as the following formulations.

$$\frac{dx^p_{a_i}(t)}{dt} = g^p_{a_{i-1}}(t) - g^p_{a_i}(t) \quad \forall p \in \mathcal{P}, i \in [1, m(p)]$$
$$x^p_{a_i}(0) = x^{p,0}_{a_i} \in \Re^1_+ \quad \forall p \in \mathcal{P}, i \in [1, m(p)]$$
$$h^{\tau,k}_p(t) = g_{a_1}(t + D_{a_1}[x_{a_1}(t)])\left(1 + D'_{a_1}[x_{a_1}(t)]\dot{x}_{a_1}\right)$$
$$g^p_{a_{i-1}}(t) = g^p_{a_i}(t + D_{a_i}[x_{a_i}(t)])\left(1 + D'_{a_i}[x_{a_i}(t)]\dot{x}_{a_i}(t)\right) \quad \forall p \in \mathcal{P}, i \in [2, m(p)]$$

To make DUE principle formula, we introduce the VI formulation, which is widely applied to equilibrium problems in different fields. Here, we use it to formulate the DTA problem.

$$\mathrm{DVI}(\Psi, \Lambda, [t_0, t_f]): \textit{find } h^* \in \Lambda_0 \textit{ such that}$$
$$\begin{cases} \sum_{p \in P} \int_{t_0}^{t_f} \Psi_p(t, h^*)(h - h^*)dt \geq 0 \quad \forall h \in \Lambda \\ \textit{where } \Lambda = h \geq 0 : \frac{dy_{ij}}{dt} = \sum_{p \in P_{ij}} h_p(t), y_{ij}(0) = 0, y_{ij}(t_f) = Q_{ij} \end{cases}$$

To solve DVI problem, we introduce the fixed-point algorithm. The following formulation is the computing process.

$$h^{k+1}_p(t) = \left[h^k_p(t) - \alpha \Psi_p\left(t, h^k\right) + v_{ij}\right]_+ \quad \forall (i,j) \in \mathcal{W}, p \in \mathcal{P}_{ij}$$
$$\sum_{p \in \mathcal{P}_{ij}} \int_{t_0}^{t_f} \left[h^k_p(t) - \alpha \Psi_p(t, h^k) + v_{ij}\right]_+ dt = Q_{ij} \quad \forall (i,j) \in \mathcal{W}$$

Algorithm

For each OD pair, given h_0	$\Psi(h^k) = F(D_p)$ $\sum \int [hk - \alpha\Psi(h^k) + v]_+ = Q$
While $\lVert h^{k+1} - h^k \rVert / \lVert h^k \rVert >= \varepsilon$ ODE(h^k) = makeOde(h^k) x = sol. to ODE(h^k) for i = 1:m $\xi_1 = t + D_1[x_1(t)]$ $\xi_{i+1} = \xi_i + D_i[x_i(\xi_i)]$ End for	v = zero of $\sum \int [hk - \alpha\Psi(h^k) + v]_+ - Q$ $h^{k+1} = [hk - \alpha\Psi(h^k) + v]_+$ End while
$D_p = \xi_m$	Cost = $\Psi(h)$

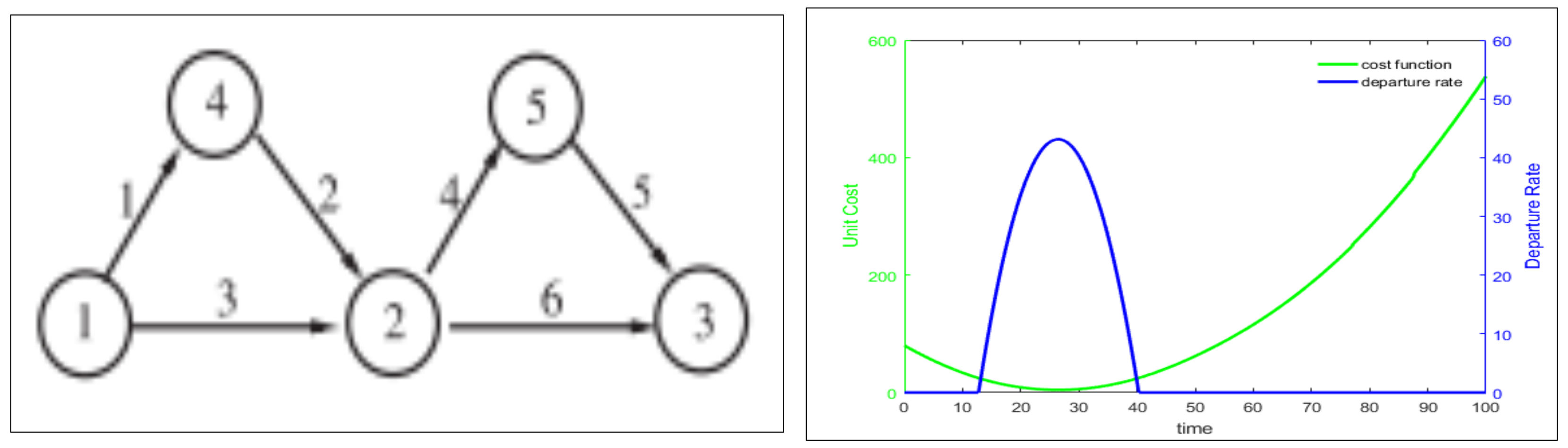

Figure 1 small network

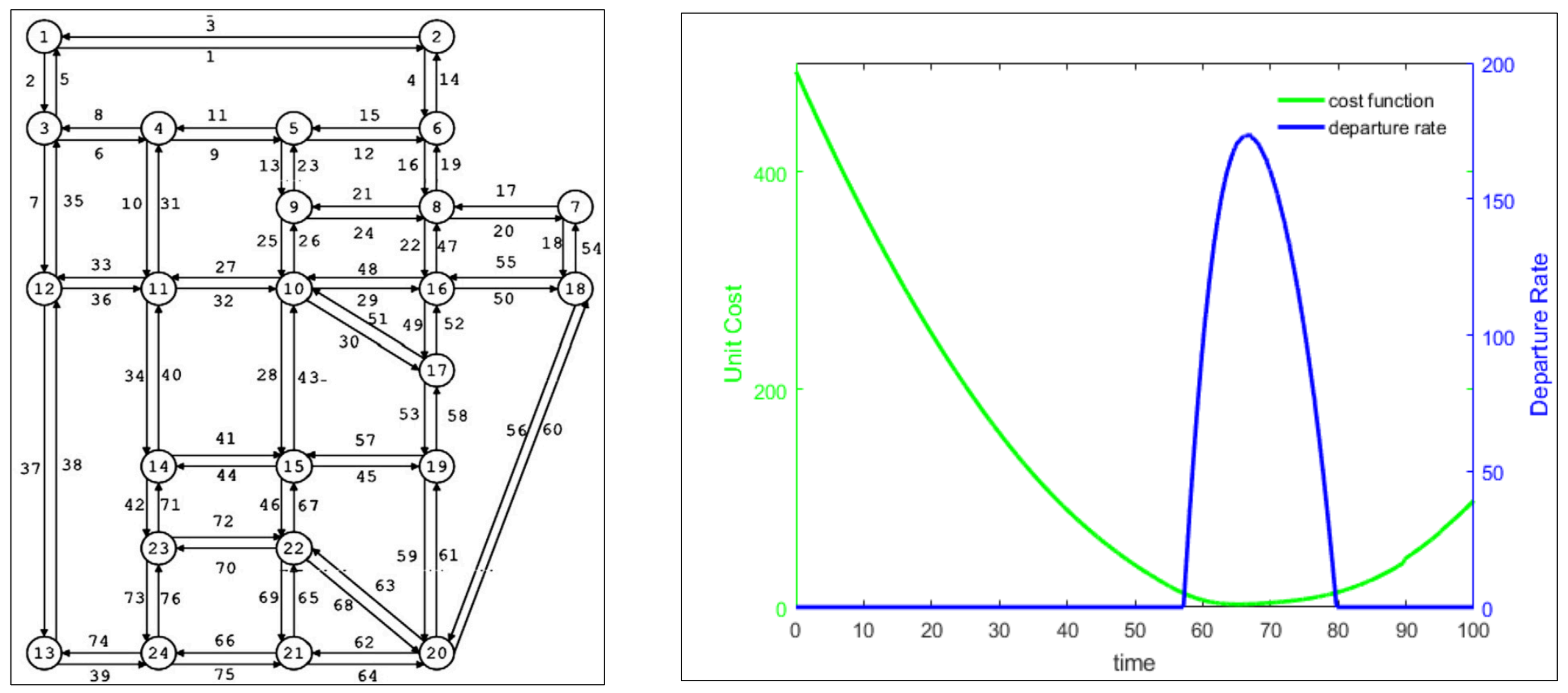

Figure 2 siouxfall network

For the network graphs shown on the left, the calculated assignment results are shown on the right. The small network runs under the tolerance 0.01 for 0.1s while Siouxfall network converges with epsilon=0.0001 for about 120s. GPU acceleration will be used to accelerate computation for larger network problems..

DNL by PDE

In this part, traffic flow is shown described as belows.

$$\begin{cases} \partial_x u(t,x) + \partial_t \phi(u(t,x)) = 0 \quad (x,t) \in [a,b] \times [t_0, t_f] \\ u(t,a) = \bar{u}(t) \end{cases}$$

Convert the PDE to optimization problem by Lax Hopf formulation.

$$\frac{d}{dt}Q^e_p(t) = q^e_p(t), \quad \frac{d}{dt}W^e(t) = w^e(t), \quad \forall p \in \mathcal{P}$$
$$q^{e_i}_p(t) = w^{e_{i-1}}_p(t), \quad i \in [1, m(p)], \quad p \in \mathcal{P}$$
$$W^e(t) = \min_\tau \left\{Q^e(\tau) + L^e \psi^e\left(\frac{t - \tau}{L^e}\right)\right\}, \quad \forall e \in \mathcal{A}$$
$$Q^e(t) = W^e(t + D(t;\ Q^e)), \quad \forall e \in \mathcal{A}$$

Example and Results

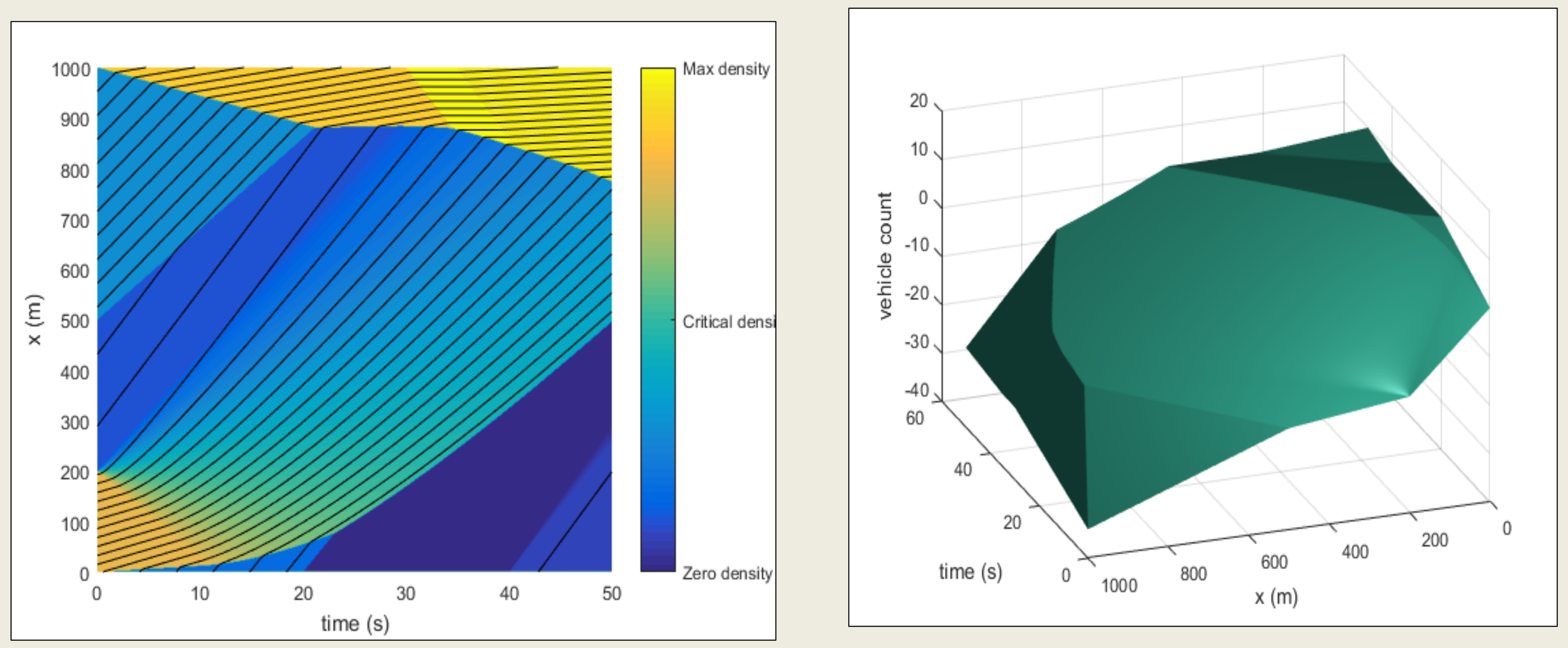

Figure3 Flow propagation based on LWR

Future Work

- ✓ For ODEs model, we will verify the convergence of the algorithm theoretically and then implement them on a parallel computing platform.
- ✓ For PDE model, we will continue to explore the physical model and build the programming model on GPU.
- ✓ We will continue to improve the models in describing some traffic phenomena.

Acknowledgements

This project was sponsored by the National Science Foundation through Research Experience for Undergraduates (REU) award, with additional support from the Joint Institute of Computational Sciences at University of Tennessee Knoxville. This project used allocations from the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by the National Science Foundation. In addition, the computing work was also performed on technical workstations donated by the BP High Performance Computing Team.