Generic second-order macroscopic traffic node model for general multi-input multi-output road junctions via a dynamic system approach

Matthew A. Wright and Roberto Horowitz

July 31, 2017

Abstract

This paper addresses an open problem in traffic modeling: the second-order macroscopic node problem. A second-order macroscopic traffic model, in contrast to a first-order model, allows for variation of driving behavior across subpopulations of vehicles in the flow. The second-order models are thus more descriptive (e.g., they have been used to model variable mixtures of behaviorally-different traffic, like car/truck traffic, autonomous/human-driven traffic, etc.), but are much more complex. The second-order node problem is a particularly complex problem, as it requires the resolution of discontinuities in traffic density and mixture characteristics, and solving of throughflows for arbitrary numbers of input and output roads to a node (in other words, this is an arbitrary-dimensional Riemann problem with two conserved quantities). We propose a solution to this problem by making use of a recently-introduced dynamic system characterization of the first-order node model problem, which gives insight and intuition as to the continuous-time dynamics implicit in first-order node models. We use this intuition to extend the dynamic system node model to the second-order setting. We also extend the well-known “Generic Class of Node Model” constraints to the second order and present a simple solution algorithm to the second-order node problem. This node model has immediate applications in allowing modeling of behaviorally-complex traffic flows of contemporary interest (like partially-autonomous-vehicle flows) in arbitrary road networks.

1 Introduction

The macroscopic approximation of vehicle traffic has proven a valuable tool for the study of traffic’s nonlinear dynamics, and the design of methods for mitigating and controlling undesirable outcomes like congestion. This macroscopic theory describes the dynamics of vehicles along roads with partial differential equations (PDEs) inspired by fluid flow. The most basic macroscopic formulation is the so-called “kinematic wave” or “Lighthill-Whitham-Richards” (LWR) due to \cite{12, 14}, which describes traffic with a one-dimensional conservation equation,

\[
\frac{\partial \rho}{\partial t} + \frac{\partial (\rho v)}{\partial z} = 0
\]

where \(\rho(z, t)\) is the density of vehicles, \(t\) is time, \(z\) is the lineal direction along the road, and \(v(\rho)\) is the flow speed. The total flow, \(\rho v\), is often expressed in terms of a flux function, \(f(\rho) = \rho v\) (the flux function on a long straight road is often called the fundamental diagram).

The LWR-type formulation (1) is a simple nonlinear model and cannot capture many characteristics of real traffic flows. For example, a flux function \(f(\cdot)\) of \(\rho\) only does not admit the oft-empirically-observed phenomenon of accelerating and decelerating flows tracing a hysteresis loop in the \((\rho, v)\) plane \cite{20}. One extension of the LWR model that can express a richer variety of dynamics is the so-called Aw-Rascle-Zhang (ARZ) \cite{1, 20} family of models. These models fit into the so-called “generic second order”\(^1\) or “extended ARZ” class of traffic models \cite{11}, which can be written as

\[
\begin{align*}
\frac{\partial \rho}{\partial t} + \frac{\partial (\rho v)}{\partial z} &= 0 \\
\frac{\partial w}{\partial t} + v \frac{\partial w}{\partial z} &= 0
\end{align*}
\]

where \(v = V(\rho, w)\)

\(^1\) As seen in (2), the “second-order” model actually consists of two first-order partial differential equations (that is, they only contain first derivatives). In a case of overloaded mathematical terminology, the name “second order” here comes from a system-theoretic view, where a second-order system is one that has two state variables: in this case, \(\rho\) and \(w\) (or, equivalently, \(\rho\) and \(v\)).
where \( w(x, t) \) is a property or invariant that is conserved along trajectories [11].

The property \( w \) in (2) can be described as a characteristic of vehicles that determines their density-velocity relationship. Members of the generic second order model (GSOM) family are differentiated by the choice of \( w \) and its relationship on the \( \rho-v \) behavior. Examples of chosen \( w \)'s include the difference between vehicles' speed and an equilibrium speed [1], driver spacing [20], or the flow's portion of autonomous vehicles [17]. An intuitive way of describing the effect of the property \( w \) in (2) is that it parameterizes a family of flow models, \( f(\rho, w) = \rho V(\rho, w) \), with different flow models for different values of \( w \) [11, 5].

For application of macroscopic traffic simulation, road networks are often modeled as directed graphs. Edges that represent individual roads are called links, and junctions where links meet are called nodes. Typically, the flow model \( f(\cdot) \) on links is called the “link model,” and the flow model at nodes is called the “node model.” Development of accurate link and node models have been areas of much research activity in transportation engineering for many years.

This paper focuses on node models for first- and second-order macroscopic models. The node model resolves the discontinuities in \( \rho \) and/or \( w \) between links and determines a Neumann boundary for nodes with merges, diverges, or both, this Riemann problem becomes multidimensional. Through this, the node model determines how the state of an individual link affects and is affected by its connected links, their own connected links, and so on through the network. As a result, it has recently been recognized that the specific node model used can have a very large role in describing the network-scale congestion dynamics that emerge in complex and large networks (for more on this, see the discussions in, e.g., the introduction sections of [16] and [8]).

In [18], we introduced a novel characterization of node models as dynamic systems. Traditional studies of node models (see, e.g., [16, 6, 2, 15, 8, 19]) usually present the node model as an optimization problem (where the node flows are found by solving this problem) or in algorithmic form (where an explicit set of steps are performed to compute the flows across the node). In contrast, the dynamic system characterization describes the flows across the node as themselves evolving over some period of time (in application, this means that the dynamic system characterization presents time-varying dynamics that are said to occur during the simulation timesteps of the link PDEs). The dynamic system characterization can be thought of as making explicit the time-varying behavior of the flows at nodes of many algorithmic node models: it was shown in [18] that the dynamic system characterization produces the same solutions as the algorithm introduced in [19], which also reduces to the one introduced in [16] as a special case.

The dynamic system characterization has proven useful in imparting an intuition as to what physical processes over time are implicit in these algorithmic node models (see the discussions referring to [18] in [19] for some examples). In this paper, we develop a dynamic system characterization of a second-order node model, and use it to solve the general node problem for second-order models.

This paper has several main contributions. The first is an extension of the dynamic system characterization of first-order node models as introduced in [18] to a simple, closed-form solution algorithm. This represents the completion of an argument began in Section 4.1 of the aforementioned reference. The second contribution is the extension of the dynamic system characterization to the generic second-order models. As we will see, the dynamic system characterization lends itself to an intuitive incorporation of the second PDE in (2) that is not obvious in the traditional, optimization-problem presentation of node models. The third contribution, and the principal contribution of this paper, parallels the first by using the second-order dynamic system node model to derive an intuitive, closed-form algorithm for computing node flows for second-order flow models for general, multi-input multi-output nodes. To the best of our knowledge, this represents the first proposed generic (applicable to multi-input multi-output nodes) node model for second-order traffic flow modeling\(^2\).

The remainder of this paper is organized as follows. Section 2 reviews the first-order node flow problem, the first-order dynamic system characterization introduced in [18], and presents the aforementioned closed-form solution algorithm (contribution one in the above paragraph). Section 3 reviews the link discretization of the GSOM (2) as presented in, e.g., [11, 5], which produces the inputs to our second-order node model, and the standard one-input one-output second-order flow problem and its solution. Section 4 presents the extension of the second-order flow problem to the multi-input multi-output case, the dynamic system characterization to the GSOM family (2) and the solution algorithm for the general node problem (contributions two and three). Finally, Section 5 concludes and notes some open problems.

---

\(^2\) A note on naming: as we will see in Section 2.1, we build off the so-called “generic class of first-order node models” to develop our second-order node model. Given that the relevant second-order model used (2) is itself called the “generic second order model,” it might be accurate to describe this paper’s results as the “genericization of the generic class of node models to the generic second-order model,” but this description likely loses in comprehensibility what it might gain in accuracy.
2 First-order node model

In this Section, we review the general first-order node problem and a particular node model (and its solution algorithm). This node model will be extended to the second-order node problem in Section 3.

The traffic node problem is defined on a junction of $M$ input links, indexed by $i$, and $N$ output links, indexed by $j$. We further define $C$ classes (sometimes called “commodities”) of vehicle, indexed by $c$. The first-order node problem takes as inputs the incoming links’ per-class demands $S^c_i$, split ratios $\beta^c_{i,j}$ (which define the portion of vehicles of class $c$ in link $i$ that wish to exit to link $j$), and outgoing links’ supplies $R_j$, and gives as outputs the set of flows from $i$ to $j$ for class $c$, $f^c_{i,j}$. We denote as a shorthand the per-class directed demand $S^c_{i,j} \triangleq \beta^c_{i,j} S^c_i$. Nodes are generally infinitesimally small and have no storage, so all the flow that enters the node must exit the node.

The rest of this Section is organized as follows. Section 2.1 defines our first-order node problem as an optimization problem defined by explicit requirements, following the example set by [16]. Section 2.2 reviews the dynamic system of [18] whose executions produce solutions to the node problem. Finally, Section 2.3 uses the dynamic system formulation as a base to develop a node model solution algorithm. This algorithm represents the completion of an argument began in [18].

2.1 “Generic Class of Node Model” requirements

The node problem’s history begins with the original formulation of macroscopic discretized first-order traffic flow models [4]. There have been many developments in the node model theory since, but we reflect only some more recent results.

We can divide the node model literature into pre- and post-[16] epochs. They drew from the literature several earlier-proposed node model requirements to develop a set of conditions for first-order node models they call the ‘generic class of first-order node models’ (GCNM). These set of conditions give an excellent starting point for our discussion of the mathematical technicalities of node models, and have been used as a starting point by many subsequent papers, such as [6, 2, 15, 8, 19]. In the following list, we present the variant of first-order GCNM requirements used in [19], which includes a modification of the first-in-first-out (FIFO) requirement (item 6 below) to [19]’s “partial FIFO” requirement.

1. Applicability to general numbers of input links $M$ and output links $N$. In the case of multi-class flow, this also extends to general numbers of classes $c$.

2. Maximization of the total flow through the node. Mathematically, this may be expressed as $\max \sum_{i,j,c} f^c_{i,j}$. According to [16], this means that “each flow should be actively restricted by one of the constraints, otherwise it would increase until it hits some constraint.” When a node model is formulated as a constrained optimization problem, its solution will automatically satisfy this requirement. However, what this requirement really means is that constraints should be stated correctly and not be overly simplified and, thus, overly restrictive for the sake of convenient problem formulation. See the literature review in [16] for examples of node models that inadvertently do not maximize node throughput by oversimplifying their requirements.

3. Non-negativity of all input-output flows. Mathematically, $f^c_{i,j} \geq 0$ for all $i$, $j$, $c$.

4. Flow conservation: Total flow entering the node must be equal to total flow exiting the node. Mathematically, $\sum_i f^c_{i,j} = \sum_j f^c_{i,j}$ for all $c$.

5. Satisfaction of demand and supply constraints. Mathematically, $\sum_j f^c_{i,j} \leq S^c_i$ and $\sum_i f^c_{i,j} \leq R_j$.

6. Satisfaction of the (partial) first-in-first-out (FIFO) constraint: If a single destination $j'$ for a given $i$ is not able to accept all demand from $i$ to $j'$, then all other flows from $i$ are constrained by the queue of $j'$-destined vehicles that builds up in $i$. The degree to which this queue restricts the other flows $f^c_{i,j}$ is partially defined by the restriction intervals $\eta^c_{i,j'} = [y, z] \subseteq [0, 1]$. This interval means that a queue in the $i,j'$ movement will block the portion of $i,j'$-serving lanes in $i$ with leftmost extent $y$ and rightmost extent $z$ (e.g., if $i, j'$ is a through movement that uses two lanes and $i, j'$ is a right-turn movement that uses the right of those two lanes, then $\eta^c_{i,j'} = [1/2, 1]$). The traditional, full FIFO behavior, where any queue in $i$ blocks all of $i$’s lanes, can be recovered by setting all $\eta^c_{i,j'} = [0, 1]$.

Continuing this example, we will have $\eta^c_{i,j'} = [0, 1]$ since the only lane in $i$ that serves movement $i, j'$ (the right lane) will be blocked by a queue for the through movement, which will queue on both lanes.

To help keep the meaning of $\eta^c_{i,j'}$ clear, we find it helpful to read it as “the restriction interval of $j'$ onto $i$.”
Another item that defines the partial first-in-first-out behavior is the amount of time that a restriction interval \( \eta^j_{ij,j} \) is active. That is, if \( j' \) is a low-supply link with relatively high demands and \( j'' \) is a high-supply link with relatively low demands, it should be the case that \( \eta^j_{ij,j} \) is active on a greater portion of the directed demand \( S^i_{ij,j} \) than \( \eta^j_{ij,j} \). We will see how this effect of time is captured in the dynamic system formulation of Section 2.2.

Finally, we require that we consider the cumulative effect of restriction intervals. Suppose that a movement \( i,j \) has an active restriction from a queue for movement \( i,j' \). Then, say that another downstream link \( j'' \) exhausts its supply, and vehicles begin queueing for the movement \( i,j'' \). Then, the new restriction on \( i,j \) (after this second queue forms) is \( \eta^j_{ij,j} \cup \eta^j_{ij,j} \).

This requirement is stated mathematically as

\[
f_{i,j}^c \leq S_{i,j} - A\left( \bigcup_{j' \neq j} \{ \eta^j_{ij,j} \times \left[ \frac{f_{i,j'}}{S_{i,j}}, S_{i,j} \right] \} \right)
\]

where \( A(\cdot) \) denotes the area of a two-dimensional object, \( \times \) denotes a Cartesian product, \( f_{i,j}^c \triangleq \sum_c f_{i,j} \), and \( S_{i,j} \triangleq \sum_c S_{i,j} \).

The formulation in (3) is complex in order to state it as an optimization constraint, and not as a consequence of the time-varying queue formation intuition outlined in the third paragraph of this item. A major contribution of the dynamic system approach to node modeling is the explicit encoding of this more intuitive description.

See [19, Sections 3.2-3.3] or [18, Section 3.2] for a much more in-depth discussion of this requirement.

### 7. Satisfaction of the invariance principle

If the flow from some input link \( i \) is restricted by the available output supply, this input link enters a congested regime. This creates a queue in this input link and causes its demand \( S_i \) to jump to capacity \( F_i \) in an infinitesimal time, and therefore, a node model should yield solutions that are invariant to replacing \( S_i \) with \( F_i \) when flow from input link \( i \) is supply-constrained [9].

### 8. Supply restrictions on a flow from any given input link are imposed on class components of this flow proportionally to their per-class demands

Mathematically, \( f_{i,j}^c / (\sum_c f_{i,j}^c) = \beta_{ij}^c S_i / (\sum_c \beta_{ij}^c S_i^c) \).

This assumes that the classes are mixed isotropically. This means that all vehicles attempting to take movement \( i,j \) will be queued in roughly random order, and not, for example, having all vehicles of commodity \( c = 1 \) queued in front of all vehicles of \( c = 2 \), in which case the \( c = 2 \) vehicles would be disproportionately affected by spillback.

We feel this is a reasonable assumption for situations where the demand at the node is dependent mainly on the vehicles near the end of the link (e.g., in a small cell at the end).

In addition to the above numbered requirements, two other elements are needed to define a node model. The first is a rule for the portioning of output link supplies \( R_j \) among the input links. Following [7], in [16] it was proposed to allocate supply for incoming flows proportionally to input link capacities, which we will denote \( F_i \). In this paper, we allocate supply proportionally to the links’ “priorities” \( p_i \) (in the spirit of [4, 13, 6, 19]). In the dynamic system view, priorities represent the relative rate at which vehicles exit each link \( i \) to claim downstream space (one reasonable formulation might be to follow the capacity-proportional example, \( p_i = F_i \), if, as in [16], it assumed that vehicles exit a link at rate \( F_i \)).

The second necessary element is a redistribution of “leftover supply.” Following the initial partitioning of supplies \( R_j \), if one or more of the supply-receiving input links does not fill its allocated supply, some rule must redistribute the difference to other input links who may still fill it. This second element is meant to model the selfish behavior of drivers to take any space available, and ties in closely with requirement 2 above. [16] referred to these two elements collectively as a “supply constraint interaction rule” (SCIR). For some discussion of choices of SCIRs in recent papers, see [19, Section 2.1].

In this paper, we consider a SCIR of the form [19]

\[
\forall i \text{ s.t. } \sum_j f_{i,j} < \sum_c S_i^c, \quad W_i \neq \emptyset \quad (4a)
\]

---

5 This amount-of-time requirement is encoded in the right part of the Cartesian product in (3). This amount-of-time requirement appears as a component of a Cartesian product in the two-dimensional rectangles that appear in [19, Section 3.2]. However, it is much more intuitive to understand this as an explicit temporal property as it appears in the dynamic system characterization, so we will not discuss [19, Section 3.2]’s derivation here.
\( \forall i \text{ s.t. } W_i \neq \emptyset, f_{i,j} \geq \frac{p_{i,j}}{\sum_{j'=1}^{J} p_{i,j'}} R_j, \forall j \in W_i \quad (4b) \)

where \( W_i = \{ j^* : \sum_c \beta_{i,j^*}^c S_i^c > 0, p_{v,j',f_{i,j'}} \geq p_{i,j}, f_{v,j'} \forall i' \neq i \} \)

The set \( W_i \) denotes all output links that restrict the flow from \( i \). The conditions for membership in \( W_i \) can be read as “there is some nonzero demand for the movement \( i, j^* (\sum_c \beta_{i,j^*}^c S_i^c > 0) \), and \( i \) claims at least its priority-proportional allocation of \( j^* \)'s supply \( (p_{v,j',f_{i,j'}} \geq p_{i,j}, f_{v,j'} \forall i' \neq i)$.” Note that if a link \( j^* \) is in \( W_i \cap W_j \), then \( p_{v,j',f_{i,j'}} = p_{i,j}, f_{v,j'} \) by construction.

Constraint (4a) says that if a link \( i \) is not able to fill its demand, then there is at least one output link in \( W_i \) that restricts \( i \), and that \( i \)'s movements claim at least as much as their oriented-priority-proportional allocation of supply. Constraint (4b) captures the reallocation of “leftover” supply, which states that a link \( i \) that cannot fulfill all of its demand to the links in \( W_i \) will continue to send vehicles after links \( i' : j \notin W_i \) have fulfilled their demands to the \( j \in W_i \).

This concludes the setup of our generic node model problem. A solution will have flows that are constrained by at least one of the constraints outlined above. An algorithm to solve this problem and proof of optimality is given in [19].

### 2.1.1 Other first-order node model requirements

Note that the list of first-order node requirements presented in Section 2.1 (which is the particular node problem of interest for the remainder of this paper) is not an exhaustive list of all “reasonable” node model requirements. Since the statement of the GCNM requirements in [16], several authors have proposed extensions or modifications (as we have in the “partial FIFO” relaxation). Beyond what we have covered here, one of the most discussed are nodal supply constraints. These supply constraints, as their name suggests, describe supply limitations at the node rather than in one of the output links. They are meant to describe restrictions on traffic that occur due to interference between flows in the junction (rather than vehicles being blocked in the input link), or the exhaustion of some “shared resource” such as green light time at a signalized intersection. Each movement through the node may or may not consume an amount of a node supply proportional to its throughput.

The node supply constraints in the GCNM framework were originally proposed by in [16]. In [2] it was noted that these node supplies may lead to non-unique solutions. Very recently, [8] revisited the node supply constraints (mostly in the context of distribution of green time) to address [2]’s critique of non-uniqueness of solutions and proposed a generalization of the flow-maximization objective that still enforces that drivers will take any available space they can.

We do not explicitly include the node supply constraints in the dynamic system node models and resulting solution algorithms in this paper. The path towards their inclusion in the first- and second-order cases is straightforward butnotationally cumbersome and somewhat beyond this paper’s scope of fusing the GCNM and second-order link models.

### 2.2 Review of first-order node dynamic system

This Section reviews the node dynamic system characterization of node models presented in [18]. This dynamic system is a hybrid system, which means that it contains both continuous and discrete states (also called discrete modes). Here, the continuous states evolve in time according to differential equations, the differential equations themselves change between discrete states, and the discrete state transitions are activated when conditions on the continuous states are satisfied.

- Let there be \( N \cdot M \cdot C \) time-varying continuous states \( x_{i,j}^c(t) \), each representing the number of vehicles of class \( c \) that have taken movement \( i, j \) through the node. The continuous state space is denoted \( X \).
- Let \( J \) be the set of all output links \( j \). Let there be \( 2^M \) discrete states \( q_{\nu}, \nu \in 2^J \) (recall \( 2^J \) refers to the power set of \( J \)), the index \( \nu \) representing the set of downstream links that have become congested. A downstream link \( j \) is said to “become congested” at time \( t \) if \( \sum_i \sum_c x_{i,j}^c(t) = R_j \). The discrete state space is denoted \( Q \).
• $\text{Init} \subseteq Q \times X$ defines the set of permissible initial states of the system at $t = 0$.

• $\text{Dom}: Q \rightarrow X$ denotes the domain of a discrete state, which is the space of permissible continuous states while the discrete state is active.

• $\Phi: Q \times X \rightarrow Q \times X$ is a reset relation, which defines the transitions between discrete states and the conditions for those transitions.

• The hybrid system execution begins at time $t = 0$.

• Each link is given a “time limit” $T_i \triangleq (\sum_c S^c_{i,j})/p_i$. This is necessary to ensure that $x_{i,j}^c < S_{i,j}^c$ when $i, j$ has a partial FIFO constraint active (which appears in the dynamic system as a flow rate attenuation).

Our hybrid system $(Q, X, \text{Init}, \dot{x}_{i,j}^c, \text{Dom}, \Phi)$ is

\begin{align*}
Q &= \{q_\nu\}, \nu \in 2^J \quad (6a) \\
X &= \mathbb{R}^{M \cdot N \cdot C} \\
\text{Init} &= Q \times \{x_{i,j}^c(t = 0) = 0 \quad \forall i, j, c\} \\
\dot{x}_{i,j}^c &= \begin{cases} 
p_{ij} \sum_c S^c_{i,j} & \text{if } x_{i,j}^c(t) < S^c_{i,j} \\
0 & \text{otherwise}
\end{cases} \quad (6d) \\
\text{Dom}(q_\nu) &= \begin{cases}
x: \sum_i \sum_c x_{i,j}^c = R_j \quad \forall j \in \nu \text{ and } \\
\sum_i \sum_c x_{i,j}^c \leq R_j \quad \forall j \notin \nu
\end{cases} \\
\Phi(q_\nu, x) &= (q_\nu', x) \text{ if } \sum_i \sum_c x_{i,j}^c = R_j, \quad (6f)
\end{align*}

where $\nu' = \nu \cup j^*$. 

When $\dot{x}_{i,j}^c = 0$ for all $i, j, c$, the execution is complete and $f_{i,j}^c = x_{i,j}^c$. □

It was shown in [18] that the hybrid system (6) produces the same solutions as [19]’s algorithm. In the following Section, we show how to quickly compute executions of the hybrid system, which, since it is based on the continuous-time dynamics of (6), presents a more intuitive algorithm than the one in [19].

### 2.3 Execution of the first-order node dynamic system as a simple algorithm

Evaluating continuous-time or hybrid systems typically involves forward integration of the differential equation(s) with fixed or varying step sizes. However, in the case of (6), evaluation can be performed in a much simpler manner. This is due to the particular dynamics of the system - since the continuous-time dynamics and the condition for discrete mode switching are very simple, the time that the next mode switch will occur can be found in closed form. Equations (6e) and (6f) say that a mode switch where link $j$ enters $\nu$ will occur when

\[ \sum_i \sum_c x_{i,j}^c = R_j. \]  

When $\dot{x}_{i,j}^c = 0$ for all $i, j, c$, the execution is complete and $f_{i,j}^c = x_{i,j}^c$. □

Say we are currently at time $t_0$. Combining (7) with (6d), we can find the time that the mode switch occurs, which we denote $t_j$.

\[ R_j = \sum_i \sum_c x_{i,j}^c(t_0) + \int_{t_0}^{t_j} \sum_i \sum_c \dot{x}_{i,j}^c dt. \]  

\[ (8) \]
Solving the integral in (8),

\[
\int_{t_0}^{t_j} \sum_i \sum_c \hat{x}_{i,j}^c dt = \int_{t_0}^{t_j} \sum_i \sum_c p_{i,j} \frac{S_{i,j}^c}{\sum_c S_{i,j}^c} \left(1 - \bigcup_{j' \in v, \exists x_{i,j'}^c < S_{i,j'}} \eta_{j',j}^i\right) dt
\]

\[= (t_j - t_0) \sum_i p_{i,j} \left(1 - \bigcup_{j' \in v, \exists x_{i,j'}^c < S_{i,j'}} \eta_{j',j}^i\right). \tag{9}\]

Then, plugging (9) into (8),

\[t_j = t_0 + \frac{R_j - \sum_i \sum_c x_{i,j}^c(t_0)}{\sum_i p_{i,j} \left(1 - \bigcup_{j' \in v, \exists x_{i,j'}^c < S_{i,j'}} \eta_{j',j}^i\right)} \tag{10}\]

This value can be computed for each output link \(j\). Then, the \(j\) with the smallest \(t_j\) will be the first link to fill and join \(\nu\). We had used \(j^*\) for this output link, so let \(t_{j^*} \triangleq \min_j t_j\). However, one of the input links may have its time limit \(T_i\) expire. This would also change the dynamics, as it stops sending vehicles at that time.

Therefore, evaluation of the system trajectory beginning from \(t_0\) can be done by (i) evaluating (10) for each output link, (ii) identifying \(t_{j^*}\), and (iii) checking whether any of the time limits \(T_i\) occur before \(t_{j^*}\). This is an event-triggered simulation: it is only necessary to determine when the next event will occur. The equations for \(\hat{x}_{i,j}^c\) over \([t_0, \min\{T_i\}, t_{j^*}]\) can then be evaluated in closed form under \(q_\nu\).

Note that the \(\hat{x}_{i,j}^c\)'s for an \(i\) may change to zero from nonzero without a change in the discrete state \(q_\nu\), if the conditional of \(x_{i,j}^c(t) < S_{i,j}^c\) in (6d) is broken. This can be understood as the \(i\) running out of vehicles that it is able to send. This may happen if \(p_i > S_i\) for that \(i\), and some (partial) FIFO constraint becomes active on \(i\). In the following algorithm, we introduce a new set, \(\mu\), that was not present in the dynamic system definition and contains the \(i\)'s that either exhaust their supply or have their time limits expire (i.e., those \(i\)'s whose \(\hat{x}_{i,j}^c\) become zero without \(j\) necessarily entering \(\nu\)).

These steps are summarized in Algorithm 1. This algorithm represents the completion of an argument began in [18].

### 3 Review of second-order flow modeling

#### 3.1 Introduction

The formulation of the GSOM seen in (2) has been called the “advective form” [5]. In this form, the property \(w\) is advected with the vehicles at speed \(v\). That is, it is constant along trajectories. This form makes the statement that the property \(w\) is a property of vehicles and is easy to understand conceptually.

However, to apply a discretization, it is useful to consider the total property \(pw\), and rewrite (2) in “conservative form” [11, 5],

\[
\frac{\partial \rho}{\partial t} + \frac{\partial (\rho v)}{\partial x} = 0, \\
\frac{\partial (\rho w)}{\partial t} + \frac{\partial (\rho w v)}{\partial x} = 0 \tag{11}\]

where \(v = V(\rho, w)\).

We will review the relevant finite-volume discretization using the Godunov scheme of (11) [11] in the next Section. For a deeper analysis on the physical properties of (11), see, e.g., [11].

We make one note on constraints imposed on the form of \(v(\rho, w)\) in (11). It has been stated [11, (19)] that, to apply the Godunov discretization to (11), one is restricted to choices of \(V(\rho, w)\) for which there is a unique \(\rho\) for every \((v, w)\) and a unique \(w\) for every \((v, \rho)\). That is, \(V(\rho, w)\) must be invertible in both its arguments.
Algorithm 1 First-order node model solution algorithm

Input: $S_i^c, \beta_{i,j}^c, R_j, p_i, \eta_{j'}^i$

Output: $f_{i,j}^c$

$t_i \leftarrow (\sum_c S_{i,j}^c)/p_i$

for all $j$

for all $c$

$S_{i,j}^c \leftarrow \beta_{i,j}^c S_i^c$

$f_{i,j}^c \leftarrow 0$

end for

$p_{i,j} \leftarrow p_i \cdot (\sum_c S_{i,j}^c)/(\sum_c S_i^c)$

end for

$\mu \leftarrow 0$

for all $i$, $\forall j, c$, $S_{i,j}^c = 0$

$\mu \leftarrow \mu \cup i$

$\dot{x}_{i,j}^c \leftarrow 0 \forall j, c$

end for

$\nu \leftarrow \emptyset$

for all $j$, $R_j = 0$

$\nu \leftarrow \nu \cup j$

end for

return $T_i, S_{i,j}^c, f_{i,j}^c, p_{i,j}, \mu, \dot{x}_{i,j}^c, \nu$

Algorithm 2 Setup and initialization

for all $i$

$t_i \leftarrow (\sum_c S_{i,j}^c)/p_i$

for all $j$

for all $c$

$S_{i,j}^c \leftarrow \beta_{i,j}^c S_i^c$

$f_{i,j}^c \leftarrow 0$

end for

$p_{i,j} \leftarrow p_i \cdot (\sum_c S_{i,j}^c)/(\sum_c S_i^c)$

end for

$\mu \leftarrow 0$

for all $i$, $\forall j, c$, $S_{i,j}^c = 0$

$\mu \leftarrow \mu \cup i$

$\dot{x}_{i,j}^c \leftarrow 0 \forall j, c$

end for

$\nu \leftarrow \emptyset$

for all $j$, $R_j = 0$

$\nu \leftarrow \nu \cup j$

end for

return $T_i, S_{i,j}^c, f_{i,j}^c, p_{i,j}, \mu, \dot{x}_{i,j}^c, \nu$
Algorithm 3 Computing the time to integrate forward, first-order case

for all $j \notin \nu$ do
  $t_j \leftarrow t + (R_j - \sum_i \sum_c f_{c,i,j}^i/x_{i,j}^i)$ \{Compute the filling time for every output link\}
end for

for all $i, j: i \notin \mu$ and $j \notin \nu$ do
  $t_{i,j} \leftarrow (S_{c,i,j}^e - f_{i,j}^e)/x_{i,j}^e$ \{for any $c$ \{by construction (Section 2.1, item 8), all $c$’s fulfill their demands at the same time\}.
end for

$dt \leftarrow \min\{\{t_{i,j}\}_{i \notin \mu, j \notin \nu}, \{t_j\}_{j \notin \nu}, \{T_i\}_{i \in \mu}\}$
return $dt$

3.2 Godunov discretization of the GSOM

The Godunov discretization of the first-order (LWR) model (1), first introduced as the Cell Transmission Model [3] is well-known. The Godunov scheme discretizes a conservation law into small finite-volume cells. Each cell has a constant value of the conserved quantity, and inter-cell fluxes are computed by solving Riemann problems at each boundary. The Godunov scheme is a first-order method, so it is useful for simulating solutions to PDEs with no second- or higher-order derivatives like the LWR formulation. In the CTM, the Riemann problem is stated in the form of the demand and supply functions.

Since (11) is also a conservation law with no second- or higher-order derivatives, the Godunov scheme is applicable as well [11]. However, due to the second PDE for $\rho_w$, an intermediate state arises in the Riemann problem and its solution [1, 20, 10, 11]. This intermediate state has not always had a clear physical meaning, and this lack of clarity likely inhibited the extension of the Godunov discretization to the multi-input multi-output node case. In our following outline of the discretized one-input one-output flow problem, we make use of a physical interpretation of the intermediate state due to [5].

A final note: in the first-order node model, we were able to ignore the first-order demand and supply functions that generated the supplies $R_j$ and per-class demands $S_{c}^{i}$. That is, we were agnostic to the method by which they were computed (and to the input and output link densities), as they did not change during evaluation of the node problem. As we will see shortly, this is not the case for the second-order flow problem (due to the intermediate state and its interactions with the downstream link). Therefore, our explanation below makes use of the second-order demand and supply functions $S(\rho, w)$ and $R(\rho, w)$, respectively.

3.2.1 Preliminaries

In this paper, we say that each vehicle class $c$ has its own property value $w_c$. The net (averaged over vehicle classes) property of a link $\ell$, denoted $w_\ell$, is

$$w_\ell = \sum_c w_c^e \rho_\ell^e \tag{12}$$

where $\rho_\ell = \sum_c \rho_\ell^e$ is the total density of link $\ell$.

In the second-order model, the fundamental diagram of a link is a function of both net density and net property as defined above. This carries over to the demand and supply functions in the Godunov discretization [11, 5]. That means that the supply and demand are defined at the link level with the net quantities $\rho_\ell$ and $w_\ell$. For an input link $i$,

$$S_i = S(\rho_i, w_i) = \begin{cases} \rho_i v_i & \text{if } \rho_i \leq \rho_c(w_i) \\ F(w_i) & \text{if } \rho_i > \rho_c(w_i) \end{cases} \tag{13}$$

where $\rho_c(w_i)$ is the critical density for property value $w_i$ and $F(w_i)$ is the capacity for property value $w_i$.

The demand from (13) is split among the classes and movements proportional to their densities and split ratios,

$$S^e_i = S_i \frac{\rho_i^e}{\rho_i} \quad \text{and} \quad S^e_{i,j} = \beta_{i,j}^e S^e_i.$$
3.2.2 Computing supply

Solving for an output link’s supply is a much more complicated problem. We will begin our discussion with a review of the one-input-one-output case [5, Sections 3.3, 3.4].

The supply $R$ of the output link in this one-to-one case, where $i$ is the input link and $j$ is the output link, is

$$R_j = R(\rho_M, w_i) = \begin{cases} F(w_i) & \text{if } \rho_M \leq \rho_c(w_i) \\ \rho_M v_M & \text{if } \rho_M > \rho_c(w_i). \end{cases} \quad (14)$$

We see that the supply of the downstream link is actually a function of the upstream link’s vehicles’ property, and the density and speed of some “middle” state, $M$. The middle state is given by [5, (16)]

$$w_M = w_i \quad (15a)$$

$$v_M = \begin{cases} V(0, w_i) & \text{if } V(0, w_i) < v_j \\ v_j & \text{otherwise} \end{cases} \quad (15b)$$

$$\rho_M \text{ s.t. } v_M = V(\rho_M, w_M) \quad (15c)$$

where $v_j = V(\rho_j, w_j)$ is the velocity of the downstream link’s vehicles and $V(\cdot)$ is the velocity function as given by the fundamental diagram.

In [5], the intuition behind the meaning of the middle state is given as follows: the middle state vehicle is actually those that are leaving the upstream link $i$ and entering the downstream link $j$. As they leave $i$ and enter $j$, they clearly carry their own property (15a), but their velocity is upper-bounded by the velocity at which that the downstream vehicles exit link $j$ and free up the space that the $i$-to-$j$ vehicles enter (15b). The middle density, $\rho_M$ (and therefore the downstream supply $R$), is then determined by both the upstream vehicles’ characteristics (i.e., $w_i$) and the downstream link’s flow characteristics (through $v_j$). In other words, the number of vehicles that can fit into whatever space is freed up in the downstream link is a function of the drivers’ willingness to pack together (defined by $w_i$). Since the meaning of supply $R_j$ is “the number of vehicles that $j$ can accept,” this means that $R_j$ is dependent on $w_i$ (15c).

Note that (15b) is also the equation by which congestion spills back from $j$ to $i$: if $j$ is highly congested, then $v_j$ will be low. This then makes $\rho_M$ large in (15c), which in turn leads to a small $R_j$ in (14).

Now that we have reviewed the 1-to-1 case, we can consider how to generalize this to a multi-input-multi-output node when we determine supply for several links.

4 Second-order node model

4.1 Multi-input-multi-output case

We saw that the reasoning behind the dependence of $R_j$ on $w_i$ was that the spacing tendencies of $i$’s vehicles determine the number of vehicles that can fit in $j$. Therefore, in generalizing to a multi-input-multi-output node, it makes sense to define a link $j$’s “middle state” as being dependent on the vehicles actually entering link $j$. That is, if $w_{j-}$, the $w$ just upstream of $j$, is the “middle state” of link $j$, then we say

$$w_{j-} = \frac{\sum_i \sum_c w_i \hat{x}_{i,c,j}^x}{\sum_i \sum_c \hat{x}_{i,c,j}^x}. \quad (16a)$$

The “$j$-upstream middle state” velocity and density, $v_{j-}$ and $\rho_{j-}$, are then

$$v_{j-} = \begin{cases} V(0, w_{j-}) & \text{if } V(0, w_{j-}) < v_j \\ v_j & \text{otherwise} \end{cases} \quad (16b)$$

$$\rho_{j-} \text{ s.t. } v_{j-} = V(\rho_{j-}, w_{j-}) \quad (16c)$$

and the supply $R_j$ is

$$R_j = R(\rho_{j-}, w_{j-}) = \begin{cases} F(w_{j-}) & \text{if } \rho_{j-} \leq \rho_c(w_{j-}) \\ \rho_{j-} v_{j-} & \text{if } \rho_{j-} > \rho_c(w_{j-}). \end{cases} \quad (17)$$

Note that in (16a), we defined $w_{j-}$ as a function of $\hat{x}_{i,j}$. Recall from the first-order node model that the $\hat{x}_{i,j}$’s can change as (i) upstream links $i$ exhaust their demand or (ii) downstream links $j$ run out of supply. These two
events correspond to discrete state changes in our hybrid system. This, of course, carries over to the second-order node model. This means that the \( j^- \) quantities, and thus the supply \( R_j \), change as \( \dot{x}^c_{i,j} \)'s change. Therefore, at each discrete state transition, we need to determine the new supply for each output link \( j \) for the new mixture of vehicles that will be entering \( j \) in the next discrete state.

We will explain how this is done through the following example. Suppose that at time \( t_0 \), we compute some \( w_{j^-}, v_{j^-}, \rho_{j^-}, \) and \( R_j \) with (16)-(17). Then, at time \( t_1 \), one of the \( \dot{x}^c_{i,j} \) for that \( j \) changes. At that point, we recompute \( \rho_j \) and \( w_j \),

\[
\begin{align*}
\rho^c_j(t_1) &= \rho^c_j(t_0) + \frac{1}{L_j} \sum_i x^c_{i,j}(t_1) \\
\rho_j(t_1) &= \sum_c \rho^c_j(t_1) \\
w_j(t_1) &= \frac{\sum_c w^c \rho^c_j(t_1)}{\rho_j(t_1)}
\end{align*}
\]  

where \( L_j \) is the length of \( j \). Then, we recompute all the “middle state” variables and \( R_j \) using (16)-(17). Critically, note that in this recomputation, the new \( v_j \) at \( t_1 \) is \( v_j(t_1) = V(\rho_j(t_1), w_j(t_1)) \). This means that \( v_j^- (t_1) \) will also be different than \( v_j^- (t_0) \). This will carry through to create a \( R_j(t_1) \) that is different from \( R_j(t_0) \), and takes into account both the vehicles that have moved into \( j \) between \( t_0 \) and \( t_1 \), and the difference in properties \( w_j^- (t_0) \) and \( w_j^- (t_1) \).

Note that if \( w_j^- (t_1) \) leads to significantly tighter packing (i.e., smaller inter-vehicle spacing) than \( w_j^- (t_0) \), it is conceivable that we will have \( R_j(t_1) > R_j(t_0) \) (especially if \( \rho_j(t_0) \) is not that much smaller than \( \rho_j(t_1) \)).

Of course, the description above assumes isotropic mixing of all vehicle classes in the link \( j \) (recall we stated this assumption for input links \( i \) in item (8) of the first-order GCNM requirements in Section 2.1).

Unlike supply, demand does not need to be recomputed since we assume the mixture of vehicles demanding each movement remains the same (due to our isotropic mixture assumption)

In summary, we state the second-order generalization of the GCNM requirements as the same as the first-order requirements stated in Section 2.1, with the addition of a constraint enforcing the conservasion of property via the second PDE: \( \sum_{i,j} f^c_{i,j} w^c = \sum_{i,j} f^c_{i,j} w^c \), and the modification of the supply constraint such that the supply is computed from the second-order fundamental diagram using the property of its incoming flows. This second point, where the supply constraint is also dependent on the flow solution, only worsens the nonconvexity of the node problem. Indeed, we are drifting away from a setting where the optimization-problem makes the most sense, and it may be more helpful to understanding to consider the physical dynamics encoded by the solution methods.

In any case, we now have all the ingredients necessary to extend our first-order hybrid system node model to the second-order formulation.

### 4.2 Dynamic system definition

We state the second-order node dynamic system as an extension to the first-order one presented in 2.2. Most of the symbols remain the same. However, we make a few changes:

- Let \( \mu \in 2^I \) (where \( I \) is the set of all input links \( i \)) be the set of all exhausted input links (this set was introduced in the first-order algorithm in Section 2.3). This is necessary to state the recalculations of supply according to the steps in Section 4.1 when a link exhausts its demand and the net property of a \( j^- \) changes.

- Paralleling \( j^* \), let \( i^* \) denote an exhausted input link. An input link is said to be exhausted at time \( t \) if \( S^c_{i,j} - \dot{x}^c_{i,j}(t) = 0 \ \forall j, c \). Note that the formula for the time of demand exhaustion remains the same as in the first-order case, \( T_i = (\sum_c S^c_i)/p_i \). If this happens, we then have a “fresh” counter of vehicles that have entered it.

- To accommodate recomputing of supply using (18), we will add more continuous states: the \( N \cdot M \cdot C \) quantities \( \dot{x}^c_{i,j} \), which will denote the flow of movement \( i, j \) for class \( c \) for movement \( c \) since the last time that supplies have been recalculated, and the \( M \cdot C \) per-class densities of the output links, \( \rho^c_j \). This is necessary because, following (16)-(18), the new supplies \( R_j \) for the new \( w_j^- \) will also take into account the vehicles \( \dot{x}^c_{i,j} \), that have already made the movement, so when determining when a link \( j \) is filled with its new supply we will need a

- We assume we have the initial \( \rho^c_j(0) \) for all \( j, c \).
Our hybrid system \((Q, X, \text{Init}, \dot{x}_{i,j}, \dot{x}_{i,j}^c, \rho_{j}^c, \text{Dom}, \Phi)\) is

\[
\begin{align*}
Q &= \{q_{\mu,\nu}\}, \ \mu \in 2^J, \ \nu \in 2^J \\
X &= \mathbb{R}^{M \cdot N \cdot C} \times \mathbb{R}^{M \cdot N \cdot C} \times \mathbb{R}^{M \cdot C} \\
\text{Init} &= Q \times \left\{ \begin{array}{l}
\dot{x}_{i,j}^c(t=0) = 0 \quad \forall i, j, c; \\
\dot{x}_{i,j}^c(t=0) = 0 \quad \forall i, j, c; \\
\rho_{j}^c(t=0) = \rho_{j}^c(0) \quad \forall j, c.
\end{array} \right.
\end{align*}
\]

\[
\dot{x}_{i,j}^c = \begin{cases} 
\sum_{c} \dot{x}_{i,j}^c & \text{if } i \not\in \nu \\
0 & \text{otherwise}
\end{cases}
\]

\[
\dot{\rho}_{j}^c = \sum_i \dot{x}_{i,j}^c
\]

\[
\text{Dom}(q_{\mu,\nu}) = \left\{ \begin{array}{l}
x : t \geq T_i \forall j, c : x_{i,j}^c = S_{i,j}^c \quad \forall i \in \mu, \\
t \leq T_i, \ \exists j, c : x_{i,j}^c \leq S_{i,j}^c \quad \forall i \not\in \mu, \\
\sum_i \sum_c \dot{x}_{i,j}^c = R_{j}^{\mu,\nu} \quad \forall j \in \nu, \\
\sum_i \sum_c \dot{x}_{i,j}^c \leq R_{j}^{\mu,\nu} \quad \forall j \not\in \nu
\end{array} \right\}
\]

\[
\Phi(q_{\mu,\nu}, x) = \begin{cases} 
(q_{\mu',\nu'}, x') & \text{if } \sum_i \sum_c \dot{x}_{i,j}^c = R_{j}^{\mu',\nu'} \\
(q_{\mu',\nu}, x') & \text{if } t = T_i \\
\quad \quad \quad \forall j, c x_{i,j}^c = S_{i,j}^c & \text{where } \mu' = \mu \cup i^*, \\
\end{cases}
\]

where \(x' = \{(x_{i,j}^c), \{\dot{x}_{i,j}^c\}, \{\rho_{j}^c\} = (\{x_{i,j}^c\}, \{0\}, \rho_{j}^c)\) and \(R_{j}^{\mu,\nu}\) from (16), (17), with \(w_j = \sum_c w_c \rho_{j}^c / \sum \rho_{j}^c\)

When \(\dot{x}_{i,j}^c = 0\) for all \(i, j, c\), the execution is complete and \(f_{ij}^{\mu,\nu} = x_{ij}^c\).

Unsurprisingly, the second-order dynamic system is more complicated than the first-order one. The reader will note that the discrete dynamics, as discussed before, are triggered by links \(j^*\) filling and links \(i^*\) emptying. The filling of a \(j^*\) and its entering into \(\nu\) remains the same as the first-order system. The emptying or time-expiry of input links, rather than being encoded in the continuous dynamics as was done in the first-order system’s (6d), is now in the discrete dynamics in (19g),(19h). While it was possible to reduce the number of discrete states in the first-order system by including \(i\)-emptying in the continuous dynamics, in the second order system, any change in the continuous dynamics changes the output links’ \(w_j^-\), so all continuous dynamics changes must trigger a recomputation of \(R_{j}\), which, in (19), we do when \(\mu\) or \(\nu\) change.

Thankfully, although the second-order system seems much more complex than the first-order system, the second-order solution algorithm is not that much more complicated than the solution method of the first-order system. We will see why in the next Section.

### 4.3 Solution algorithm

Note that, just as in the first-order system, the second order system has constant continuous dynamics in each discrete state. This means that, just as in the first-order case, we can easily compute the time that the next discrete state transition occurs. Like in Section 2.3, this is the smallest of the \(t_j\)’s and \(T_i\)’s. As we said, the input link “time limits” remain the same as before, \(T_i = (\sum c S_{i,j}^c) / \rho_i\). The time that an output link runs out of supply and is filled under the discrete state \(q_{\mu,\nu}\), if \(t_0\) is the time that the discrete state switched to \(q_{\mu,\nu}\) and \(j\)’s supply was recomputed,
is similar to (10),

\[
t_j = t_0 + \sum_i \sum_j \sum_c x_{i,j}^c = t_0 + \frac{R_{j}^{\eta_{i,j},\nu}}{\sum_{i \notin \mu} p_{i,j} \left( 1 - \left| \bigcup_{j' \in \nu, \exists c: x_{i,j}' < S_{i,j}' c} \right| \right)}
\]  

(20)

but differs in two key ways. First, the term for supply is the recomputed \( R_{j}^{\eta_{i,j},\nu} \) from (16), (17) (this also accounts for why the numerator in (20) does not have a subtracted quantity as in (10), as that subtraction of already-filled supply is accounted for in the recomputed supply. Second is that the denominator is summed over \( i \notin \mu \) rather than all \( i \), as the set \( \mu \) is not in the definition of the first-order dynamic system as stated in Section 2.2.

We now state solution algorithm for the second-order dynamic system. It follows the same logic as the first-order case: identifying the next \( T_i \) or \( t_j \) to occur, finding the constant continuous-time dynamics that the system will evolve under until that time, integrating forward in time, a new step of recomputing supply, and repeating.

Algorithm 4 Second-order node model solution algorithm

Input: \( S_{c,i}, \rho_{c,j}^e, \beta_{c,i,j}, p_{i}, \eta_{i,j}, R_j \) \( \{ R_j \text{ is only the initial } R_j \text{ and will be re-computed} \} \)
Output: \( f_{c,i,j}^e \)

\( T_i, S_{c,i,j}^e, f_{c,i,j}^e, \rho_{c,j}^e, w_{c,j}^e, \beta_{c,i,j}, p_{i,j}, \eta_{i,j}^e, R_j \leftarrow \) (Algorithm 5)

\( t \leftarrow 0 \) \{Begin main loop\}

while \( \exists i,j,c: \dot{x}_{i,j}^c \neq 0 \) do

for all \( i, j, c: i \notin \mu \) do

\( \dot{x}_{i,j}^c \leftarrow p_{i,j} \frac{S_{c,i,j}^e}{S_{c,i,j}^e - f_{c,i,j}^e} \left( 1 - \left| \bigcup_{j' \in \nu, \exists c: x_{i,j}' < S_{i,j}' c} \right| \right) \) \{Compute continuous-time dynamics\}

end for

dt \leftarrow \) (Algorithm 6)

for all \( i, j, c \) do

\( f_{c,i,j}^e \leftarrow f_{c,i,j}^e + \dot{x}_{i,j}^c \cdot dt \)

\( x_{i,j}^c \leftarrow x_{i,j}^c \cdot dt \)

end for

\( \rho_{j}, w_{j} \leftarrow \) (Algorithm 8)

for all \( i: \forall j, c, S_{c,i,j}^e - f_{c,i,j}^e = 0 \) do

\( \mu \leftarrow \mu \cup i \) \{Account for all emptied input links\}

\( \dot{x}_{i,j}^c \leftarrow 0 \forall j, c \)

end for

for all \( j: t + dt = t_j \) do

\( \nu \leftarrow \nu \cup j \) \{Account for all filled output links\}

end for

\( t \leftarrow t + dt \)

end while

return \( f_{c,i,j}^e \)

Algorithm 5 Setup and initialization, second-order case

\( T_i, S_{c,i,j}^e, f_{c,i,j}^e, \rho_{c,j}^e, \beta_{c,i,j}, p_{i}, \eta_{i,j}, R_j \leftarrow \) (Algorithm 2)

for all \( i \) do

\( w_i = \sum_{c} \frac{w_{c,j}^e}{p_{i}} \)

end for

for all \( i, j, c \) do

\( \dot{x}_{i,j}^c \leftarrow 0 \)

end for

return \( T_i, S_{c,i,j}^e, f_{c,i,j}^e, \rho_{c,j}^e, \beta_{c,i,j}, p_{i}, \eta_{i,j}^e, R_j \)
Algorithm 6 Computing the time to integrate forward, second-order case

for all $j$ do
    $R_j \leftarrow$ (Algorithm 7)
end for

for all $j \notin \nu$ do
    $t_j \leftarrow t + R_j / \dot{x}^c_{i,j}$ \{Compute the filling time for every output link\}
end for

for all $i, j : i \notin \mu$ and $j \notin \nu$ do
    $t_{i,j} \leftarrow (S^c_{i,j} - f^c_{i,j}) / \dot{x}^c_{i,j}$ for any $c$
end for

dt $\leftarrow \min\{t_{i,j} \mid i \notin \mu, j \notin \nu, \{T_i\}_{i \notin \mu}\}$
return $dt$

Algorithm 7 Second-order computation of supply

\[ w_{j-} = \frac{\sum_i \sum_c w_i x^c_{i,j}}{\sum_i \sum_c x^c_{i,j}} \]
if $V(0, w_{j-}) < v_j$ then
\[ v_{j-} = V(0, w_{j-}) \]
else
\[ v_{j-} = v_j \]
end if

\[ \rho_{j-} \, \text{s.t.} \, v_{j-} = V(\rho_{j-}, w_{j-}) \]
if $\rho_{j-} \leq \rho_c(w_{j-})$ then
\[ R_j = R(\rho_{j-}, w_{j-}) \]
else
\[ R_j = F(w_{j-}) \]
end if
return $R_j$

Algorithm 8 Recomputing the downstream links’ density and property

\[ \rho^c_j \leftarrow \rho^c_j + \frac{1}{L_j} \sum_i \dot{x}^c_{ij} \]
\[ \rho_j \leftarrow \sum_c \rho^c_j \]
\[ w_j \leftarrow \sum_c w^c \rho^c_j \]
return $\rho_j, w_j$
4.4 Second-order extension of the GCNM requirements?

In solving the second-order node problem, the fact that the supply must be continually recalculated can be interpreted as indicating that the use of the supply and demand quantities is not as natural as in the first-order case. We see that demand $S(\rho, w)$ and supply $R(\rho, w)$ alone (i.e., not including the $\rho$ and $w$) is not enough to solve the node problem in the second-order case: the link $\rho$ and $w$ quantities are required. This is not unnatural: the node problem is, after all, a Riemann problem to resolve discontinuities in $\rho$ and $w$. In the first-order case, the node problem is often stated in terms of supply and demand instead of the actual conserved quantity $\rho$ because i) they have a more intuitive physical meaning, and ii) since link densities are not needed beyond their use in $S(\rho)$ and $R(\rho)$ for the first-order problem, beginning with $S(\rho)$ and $R(\rho)$ simplifies the problem by removing one step. However, we have seen that using the beginning-with-supply-and-demand framework in the second-order case does not simplify the problem along the lines of ii, as we still need to make use of $\rho$ and $w$. Therefore, in the future it may make more sense to state the second-order node problem as taking inputs of $\rho$ and $w$ for all links, rather than its inputs being $S(\cdot)$ and $R(\cdot)$. That would remove the unintuitive nature of needing to “recompute supply.”

5 Conclusion

This paper presented a generalization of the widely-used “Generic Class of Node Model” macroscopic traffic junction models to the so-called “General Second Order Model” flow model. This paper’s results allow the extension of macroscopic modeling of variable-behavior flows (based on different mixtures of driving behavior) to complex general networks. Many of these flows and networks had been only able to be modeled by microscopic models that consider the behavioral variability on a per-car level, but macroscopic models that can capture the aggregate features as a more granular model can greatly increase the scale of problems that we are able to study. As stated before, the second-order flow models have been used to represent flows of great contemporary interest, such as mixtures of human-driven and autonomous vehicles [17]. Researchers and practitioners will need to use every tool available to understand and predict the system-level changes that will arise from the traffic demand changing not just in size, but in characteristics.

Some immediate avenues for future refinement of second-order macroscopic models presented themselves during this paper. As mentioned in Section 2.1.1, we do not address node supply constraints in this paper’s node models. However, the immediate application of a general, multi-input-multi-output second-order node model, macroscopic simulation of mixed-human-driven-and-autonomous traffic on complex networks, is of particular concern in scheduling problems involving green light timing. Future work, then, should incorporate the node supply constraints into the general second-order node problem so that they may be used in signal optimization and the still-developing potential that connected and automated vehicles bring to traffic control.

References

[1] A. Aw and M. Rascle. Resurrection of ‘second order’ models of traffic flow. *SIAM Journal on Applied Mathematics*, 63(1):916–938, 2000.

[2] R. Corthout, G. Flötteröd, F. Viti, and C. M. Tampère. Non-unique flows in macroscopic first-order intersection models. *Transportation Research Part B: Methodological*, 46(3):343–359, Mar. 2012.

[3] C. Daganzo. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological*, 28(4):269–287, 1994.

[4] C. Daganzo. The cell transmission model, Part II: Network traffic. *Transportation Research Part B: Methodological*, 29(2):79–93, 1995.

[5] S. Fan, Y. Sun, B. Piccoli, B. Seibold, and D. B. Work. A Collapsed Generalized Aw-Rascle-Zhang Model and Its Model Accuracy. *arXiv preprint arXiv:1702.03624*, 2017.

[6] G. Flötteröd and J. Rohde. Operational macroscopic modeling of complex urban road intersections. *Transportation Research Part B: Methodological*, 45(6):903–922, July 2011.

[7] G. Gentile, L. Meschini, and N. Papola. Spillback congestion in dynamic traffic assignment: a macroscopic flow model with time-varying bottlenecks. *Transportation Research Part B: Methodological*, 41(10):1114–1138, 2007.
[8] S. E. Jabari. Node modeling for congested urban road networks. *Transportation Research Part B: Methodological*, 91:229–249, Sept. 2016.

[9] J. Lebacque and M. Khoshyaran. First-order macroscopic traffic flow models: intersection modeling, network modeling. In *The 16th International Symposium on Transportation and Traffic Theory (ISTTT)*, pages 365–386, 2005.

[10] J.-P. Lebacque, S. Mammar, and H. Haj-Salem. The Aw–Rascle and Zhang’s model: Vacuum problems, existence and regularity of the solutions of the Riemann problem. *Transportation Research Part B: Methodological*, 41(7):710–721, Aug. 2007.

[11] J.-P. Lebacque, S. Mammar, and H. H. Salem. Generic second order traffic flow modelling. In *Transportation and Traffic Theory 2007. Papers Selected for Presentation at ISTTT17*, pages 755–776, 2007.

[12] M. Lighthill and G. Whitham. On kinematic waves I: Flow movement in long rivers. II: A theory of traffic flow on long crowded roads. *Proc. Royal Society of London, Part A*, 229(1178):281–345, 1955.

[13] D. Ni and J. D. Leonard. A simplified kinematic wave model at a merge bottleneck. *Applied Mathematical Modelling*, 29(11):1054–1072, Nov. 2005.

[14] P. Richards. Shock waves on the highway. *Operations Research*, 4(1):42–51, 1956.

[15] E.-S. Smits, M. C. Bliemer, A. J. Pel, and B. van Arem. A family of macroscopic node models. *Transportation Research Part B: Methodological*, 74:20–39, Apr. 2015.

[16] C. M. J. Tampère, R. Corthout, D. Cattrysse, and L. H. Immers. A generic class of first order node models for dynamic macroscopic simulation of traffic flows. *Transportation Research, Part B*, 45(1):289–309, 2011.

[17] R. Wang, Y. Li, and D. B. Work. Comparing traffic state estimators for mixed human and automated traffic flows. *Transportation Research Part C: Emerging Technologies*, 78:95–110, May 2017.

[18] M. Wright, R. Horowitz, and A. A. Kurzhanskiy. A dynamic system characterization of road network node models. In *Proceedings of the 10th IFAC Symposium on Nonlinear Control Systems*, volume 49, pages 1054–1059, August 2016.

[19] M. A. Wright, G. Gomes, R. Horowitz, and A. A. Kurzhanskiy. On node and route choice models for high-dimensional road networks. *Submitted to Transportation Research Part B*, 2016.

[20] H. M. Zhang. A non-equilibrium traffic model devoid of gas-like behavior. *Transportation Research Part B: Methodological*, 36(3):275–290, 2002.